Decamouflage: A Framework to Detect Image-Scaling Attacks on CNN

Bedeuro Kim1,2, Alsharif Abuaduba2,3, Yansong Gao2,4, Yifeng Zheng2,5, Muhammad Ejaz Ahmed2, Surya Nepal2,3, Hyoungshick Kim1,2
1Department of Electrical and Computer Engineering, Sungkyunkwan University, 2CSIRO’s Data61, 3Cybersecurity CRC, 4Nanjing University of Science and Technology, 5Harbin Institute of Technology
kimbr@skku.edu, sharif.abuaduba@data61.csiro.au, yansong.gao@data61.csiro.au, yifeng.zheng@hit.edu.cn, ejaz.ahmed@data61.csiro.au, surya.nepal@data61.csiro.au, hyoung@skku.edu

I. INTRODUCTION

Deep learning models have shown impressive success in solving various tasks [1], [2], [3], [4]. One representative domain is the computer vision that is eventually the impetus for the current deep learning wave [1]. The convolutional neural network (CNN) models are widely used in the vision domain because of its superior performance [1], [5], [2]. However, it has been shown that deep learning models are vulnerable to various adversarial attacks. Hence, significant research efforts have been directed to defeat the mainstream of adversarial attacks such as adversarial samples [6], [7], backdooring [8], [9], and inference [10], [11].

Xiao et al. [12] introduced a new attack called image-scaling attack (also referred to as camouflage attack) that potentially affects all applications using scaling algorithms as an essential pre-processing step, where the attacker’s goal is to create attack images presenting a different meaning to humans before and after a scaling operation. This attack would be a serious security concern for computer vision applications. Below we first give a concise example of the image-scaling attack and exemplify its severe consequences.

Image-scaling attack example. Input of CNN models typically takes fixed-size images such as 224 × 224 (representing the height, width) so as to reduce the complexity of computations [2]. However, the size of raw input images can be varied or become much larger (e.g., 800 × 600) than this fixed-size. Therefore, the resizing or downscaling process is a must before feeding such larger images into an underlying CNN model. Xiao et al. [12] revealed that the image-scaling process is vulnerable to the image-scaling attack, where an attacker intentionally creates an attack image that is visually similar to a base image for humans but recognized as a target image by the CNN model after image-scaling function (e.g., resizing or downscaling) is applied to the attack image. Figure 1 illustrates an example of image-scaling attacks. The ‘wolf’ image is disguised delicately into the ‘sheep’ image as base image to form an attack image. When the attack image is down-sampled/resized, the ‘sheep’ pixels are discarded, and the ‘wolf’ image is finally presented. General, image-scaling attack abuses an inconsistent understanding of the same image between humans and machines.

Fig. 1: Example of image-scaling attacks presenting a deceiving effect. The left image shows what human sees before the scaling operation and the right image shows what the CNN model sees after the scaling operation.

The strength of the image-scaling attack is its independence on CNN models and data — it requires no knowledge of training data and the model because it mainly exploits the image-scaling function used for pre-processing. For image-scaling attacks, only knowledge about the used image-scaling function is required. It is noted that the attacker can relatively easily obtain this information because a small number of well-known image-scaling functions (e.g., nearest-neighbor, bilinear, and bicubic interpolation methods) are commonly used for real-world services, and a small number of input sizes (e.g., 224 × 224 and 32 × 32) are used for representative CNN models [12], as summarized in Table I. Furthermore, the pa-
parameters for the image-scaling function can be exposed to the public in some services. Nonetheless, even when the parameter information is not provided explicitly, it is feasible to infer the function parameter information used in a target service with API queries under a limited trials by an attacker [12].

<table>
<thead>
<tr>
<th>Model</th>
<th>Size (pixels * pixels)</th>
</tr>
</thead>
<tbody>
<tr>
<td>LeNet-5</td>
<td>32 * 32</td>
</tr>
<tr>
<td>VGG, ResNet, GoogleNet, MobileNet</td>
<td>224 * 224</td>
</tr>
<tr>
<td>AlexNet</td>
<td>227 * 227</td>
</tr>
<tr>
<td>Inception V3/V4</td>
<td>299 * 299</td>
</tr>
<tr>
<td>DAVE-2 Self-Driving</td>
<td>200 * 66</td>
</tr>
</tbody>
</table>

The image-scaling attacks can target various surfaces. First, as an evasive attack, the attack images crafted via image-scaling attacks can achieve the attack effect similar to adversarial examples with an advantage of agnostic to underlying CNN models. This attack has successfully mounted on a number of commercial cloud-based computer vision services that deploy the state-of-the-art machine learning models including Microsoft Azure1, Baidu2, and Tencent3. Second, the attack image can be exploited for data poisoning to insert a backdoor into any model trained over the poisonous data (see Section VI).

Unlike other adversarial attacks where corresponding countermeasures have been well investigated, only one study suggested defense mechanisms against image-scaling attacks. Quiring et al. [13] first analyzed the root cause of image-scaling attacks and proposed two defense mechanisms, (1) use of robust scaling algorithms and (2) image reconstruction, to prevent image-scaling attacks by delicately exploiting the relationship between the downsampling frequency and the convolution kernel used for smoothing pixels. The proposed defense mechanism sanitizes those pixels, which renders the image-scaling attack technique unable to inject target pixels with the required quality. However, their defense approaches have the following downsides. First, the use of robust scaling algorithms is likely to cause backward compatibility problems with existing scaling algorithms in OpenCV and TensorFlow. Second, as Quiring et al. [13] mentioned, small artifacts from an attack image can remain even after applying their suggested scaling algorithms, as the manipulated pixels are not cleansed and still contribute to the scaling. Third, the image reconstruction method removes the set of pixels in the attack images and reconstructs those pixels with image filters. This approach would significantly decrease the attack chance, but it can degrade the quality of input normal images for CNN models—noting attack images are rare in comparison with normal images.

From the security deterrence perspective, prevention is blinded [13], which is unable to tell whether an attack eventually occurs, thus infeasible to track the provenance of the launched attack. Blinded prevention is unable to provide deterrence, since there is no price or risk faced by an attacker. In contrast, detection addresses such concern where the attacker can be captured and under penalty risks whenever they want to launch the attack. Therefore, detection is desirable in the real-world critical system—help identify attack sources. In addition, detection is complementary with the prevention. Once the input is regarded as an adversarial image, the attack effect removal [13] can be consequentially applied to remove the attacking effect. In this context, the pixel sanitizing operation is solely applied to those attack images to restore the correct classification, thus minimizing the performance degradation resulted from the pixel sanitizing on any normal inputs. In this process, both detection and prevention are realized.

Therefore, to minimize image quality degradation on normal inputs with prevention mechanisms and most importantly provide deterrence on the attack, we focused on detecting attack images regarding the image-scaling attack, including one novel angle e.g., treating the image-scaling attack as a kind of steganography for information hiding. We aim to develop a defense mechanism to detect attack images without any modifications to input images for CNN models. Also, we develop Decamouflage as an independent module compatible with any existing scaling algorithms—alike a plug-in protector. Furthermore, Decamouflage is designed for detecting attack images crafted via image-scaling attacks even under black-box settings where there is no prior information about the attack algorithm.

Our key contributions are summarized as follows:

- **Decamouflage** is the first practical solution to detect image-scaling attacks. We develop three different detection methods (scaling, filtering, and steganalysis). Our source code is released at https://github.com/kimbedeuro/Decamouflage4.

- We identify three fundamental metrics (mean squared errors (MSE), structural similarity index (SSIM), and centered spectrum points (CSP)) that can be used to distinguish benign images from attack images generated by image-scaling attacks.

- We empirically validate the feasibility of Decamouflage for both the white-box setting (with the knowledge of the attacker’s algorithm) and the black-box setting (without the knowledge of the attacker’s algorithm). We demonstrate that Decamouflage can be effective in both settings with experimental results.

- We evaluate the detection performance of Decamouflage using an unseen testing dataset to show its practicality. We used the “NeurIPS 2017 Adversarial Attacks and Defences Competition Track” image dataset [14] to find the optimal thresholds for Decamouflage and used the “Caltech 256” image dataset [15] for testing. To implement image-scaling attacks, we use the code released in the original work by Xiao et al. [12]. The experimental results demonstrate that Decamouflage achieves detection

---

1https://azure.microsoft.com/en-us/services/cognitive-services/computer-vision/?v=18.05
2https://ai.baidu.com/tech/imagerecognition/fine_grained
3https://ai.qq.com/product/visionimgidy.shtml
4The artifacts including source code will be released upon the publication.
accuracy of 99.9% with a false acceptance rate of 0.0% and a false rejection rate of 0.1% in the white-box setting, and detection accuracy of 98.5% with a false acceptance rate of 0.3% and a false rejection rate of 2.7% in the black-box setting. In addition, the running time overhead of Decamouflage is less than or equal to 137 milliseconds on average evaluated with a personal PC with an Intel Core i5-7500 CPU (3.41GHz) and 8GB memory, indicating that Decamouflage can be deployed for online detection.

- We show the robustness of Decamouflage against image-scaling attacks with varying image sizes and the visual constraint parameter. The scaling detection method using MSE perfectly detects all attacks in both white-box and black-box settings.

II. BACKGROUND

In this section, we provide the prior knowledge for the image-scaling attack and its enabled insidious backdoor attack.

A. Image-Scaling Attack

The pre-processing steps for input images in a typical deep learning pipeline is an essential stage. Recently, Xiao et al. [12] demonstrated a practical adversarial attack targeting the scaling functions used by widely used deep learning frameworks. The attack exploited the fact that deep learning-based models accept only fixed-size input images.

![Image-Scaling Attack Diagram](image)

Fig. 2: Overall process of an image-scaling attack. An adversary creates an attack image \( A \) (tampered sheep image) such that it looks like \( O \) (original image) to humans, but it is recognized as \( T \) (targeted wolf image) by CNN models after applying image-scaling operations. Here \( X \approx Y \) represents that \( X \) looks similar to \( Y \).

One detailed example is illustrated in Figure 2, where a wolf is disguised into a sheep image. The human sees sheep, but the model sees a wolf once the tampered sheep image \( A \) undergoes the downsampling step. More precisely, the adversary slightly alters an original image \( O \) so that the obtained attack image \( A = O + \Delta \) resembles a target image \( T \) once downscaled. The attack mechanism can be demonstrated as the following quadratic optimization problem:

\[
\min \|\Delta\|^2 \quad \text{s.t.} \quad \|\text{scale}(O + \Delta) - T\|_{\infty} \leq \epsilon
\]

(1)

where \( \epsilon \) is the constraint parameter (within [0,1]) representing the maximum visual similarity between the original image \( O \) and the obtained attack image \( A \).

Each pixel value of \( A \) needs to be maintained within the fixed range (e.g., [0,255] for 8-bit images). This problem can be solved with Quadratic Programming (QP) [8]. The successful attack criteria are that the obtained image \( A \) should be visually similar to the original image \( O \), but the downscaled output \( D \) should be recognized as the target image \( T \) after scaling. In other words, the attack has to satisfy two properties:

- The resultant attack image \( A \) should be visually indistinguishable from the original image \( O \) (\( A \approx O \)).
- The output image \( D \) downscaled from the attack image \( A \) should be recognized as the target image \( T \) by CNN models (\( T \approx D \)).

III. POTENTIAL DETECTION METHODS: KEY INSIGHTS

To proactively defeat the image-scaling attack, one would first identify potential methods from different angles. Therefore, the first research question (RQ) is as below.

RQ. 1: What are the potential methods to reveal the target image embedded by the image-scaling attack?

This work identifies three efficient methods and visualizes their ability to detect that attack. Here we provide a general concept for each method. We exchangeably use the terms original image and benign image in the rest of this paper.

A. Method 1: Scaling Detection

We first explore the potential of reverse-engineering the attack process. In the attack process, the attack image \( A \) is downscaled to the output image \( D \) to be recognized as \( T \) for CNN models. Therefore, we seek to upscale the output image \( D \) to the upscaled image \( S \) in the reverse engineering process. Based on the reverse engineering process, we design a detection method as follows. Given an input image \( I \) (which can potentially be an attack image) for a CNN model, we apply the downsampling and upsampling operations in sequence to obtain the image \( S \) and measure the similarity between \( I \) and \( S \). Our intuition is that if the input image \( I \) is a benign image (i.e., the original image \( O \)), \( S \) will remain similar to \( I \); otherwise, \( S \) would be significantly different from \( I \) (see Figure 3).

![Scaling Detection Diagram](image)

Fig. 3: Overview of the scaling detection method. We obtained the upscaled image \( S \) from the downscaled image \( D \) and then measured the image similarity between \( S \) and the input image \( I \). If the input image \( I \) is a benign image (i.e., original image \( O \)), \( S \) will remain similar to \( I \); otherwise, \( S \) would be significantly different from \( I \).
Xiao et al. [12] suggested the color histogram as an image similarity metric for detecting attack images without conducting experiments. However, we found that the color histogram is not a valid metric. Our observation is consistent with the results in [16]. Therefore, it is non-trivial to find a proper metric to distinguish the case of attack images from benign images. We will discuss this issue in Section IV.

B. Method 2: Filtering Detection

The image-scaling attack relies on embedding the target image pixels within the original image pixels to avoid human visual inspection by abusing image-scaling functions. Therefore, if we use image filters to remove noises, the embedded target image pixels can be removed or disrupted because the embedded target image pixels would be significantly different from the original image pixels. Figure 4 shows the results of an attack image after applying the minimum filter [17], the median filter, and the maximum filter, respectively. We can see that the minimum filter can potentially expose the target image. Thus, we use the minimum filter for this method.

Based on this observation, we suggest another detection method. Given an input image $I$ for a CNN model, we apply an image filter to obtain the image $F$ and measure the similarity between $I$ and $F$. Our intuition is that if the input image $I$ is a benign image, $F$ will remain similar to $I$; otherwise, $F$ would be significantly different from $I$. For this purpose, we take the minimum filter for illustration and elaborated descriptions.

The minimum filter is used with fixed window size. Figure 4 illustrates how the minimum filter works on an image. The image filtering process is done by dividing the image $M \times N$ into smaller 2D blocks $x_i \times y_j$ where $b$ is the number of blocks and $x, y$ are the filter size. For example, as shown in Figure 5, when we use the minimum filter, only the smallest pixel value among a neighborhood of the block $x_i \times y_j$ is selected. In this paper, we used Pillow [18], which is a popularly used imaging library. We empirically chose the $3 \times 3$ size for the minimum filter because $3 \times 3$ is the minimum size that Pillow can support and also preserves an acceptable detection rate.

We will discuss how to measure the image similarity between $I$ and $F$ and determine whether a given image is an attack image in Section IV.

C. Method 3: Steganalysis Detection

The image-scaling attack’s key idea is to embed the target image as cluttered pixels so that they are less recognized by human eye perceptuality. Consequently, we treat the perturbed pixels as information that the attacker tries to hide in this method, which is similar to steganography [19]. Steganography is a technique of hiding information in digital media such as images to avoid secret data detection by unintended recipients. Therefore, we may constructively employ steganalysis mechanisms to expose the hidden perturbed pixels embedded by the image-scaling attack based on the similarity between the image-scaling attack and steganography.

We explore the frequency domain based steganalysis mechanism to find out the perturbed pixels within the attack image. Fourier Transform (FT) is an operation that transforms data from the time (or spatial) domain into the frequency domain [20]. Because an image consists of discrete pixels rather than continuous patterns, we use the Discrete Fourier Transformation (DFT) [21]. We first transform the input (potential attack) image $A$ into the 2-dimensional space, namely spectrum image. For a square image of size $N \times N$, the 2-dimensional DFT is given by:

$$F(k, l) = \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} f(i,j) e^{-i2\pi\left(\frac{ki}{N} + \frac{lj}{N}\right)}$$

(2)

where $f(i,j)$ is the spatial domain images, and the exponential term is the corresponding basis function to each $F(k,l)$ point in the DFT space. The basis functions are sine and cosine waves with increasing frequencies as depicted below:

$$\left[ \cos\left(2\pi\left(\frac{ki}{N} + \frac{lj}{N}\right)\right) - i \cdot \sin\left(2\pi\left(\frac{ki}{N} + \frac{lj}{N}\right)\right) \right]$$

(3)

The resultant DFT spectrum contains the low and high-frequency coefficients. The low frequencies capture the image’s core features, whereas the high frequency reflects the less significant regions within an image. Direct visualization of both frequencies shows that a broad dark region in the middle represents the high frequency, while low frequency appears as a whiter cluttered area on the edges. This visualization can not provide us with an automated quantification to distinguish attack images from benign images. Therefore, we apply logarithmic with a shift to flip the whiter frequency to centralize the low frequencies called centered spectrum as given by:
where $\Theta$ is the predetermined shift for $F(k, l)$ low-frequency point. We empirically use the $20 \times F(x, y)$ which is typically used to generate the centered spectrum.

If we apply the DFT operation on a benign image, a benign image has one centered spectrum point. However, as shown in Figure 6, attack images overall exhibit multiple centered spectra as opposed to one centered spectrum point observed in benign images because the cohesion of the original image pixels is broken due to the arbitrary perturbation to embed the target image pixels.

Based on this observation, we suggest the frequency domain based steganalysis detection method. Given an input image $I$ (which can be an attack image) for a CNN model, we convert it into a fourier spectrum to obtain the image $B$ and count the centered spectrum points in $B$. We will discuss how to count the number of the centered spectrum points and determine whether a given image is an attack image in Section IV.

![Fig. 6: Centered spectrum points on a benign image and an attack image.](image)

**Summary:** As an answer to RQ. 1, we suggest that three detection methods (scaling, filtering, and steganalysis) can potentially expose attack images generated by image-scaling attacks. Each method is designed based on a different insight/angle to detect image-scaling attacks. The scaling detection and filtering detection methods are designed to detect the image-scaling attacks in the spatial domain, while the steganalysis method is designed to detect the image-scaling attacks in the frequency domain.

**IV. Decamouflage System Design**

In this section, we provide the Decamouflage framework exploiting the above-identified detection methods to answer the RQ. 2:

**RQ. 2: How can we develop an automated process to detect image-scaling attacks using the identified methods?**

We first define the threat models that we focused on in this paper. Next, we introduce three key metrics to find image-scaling attacks in an automated manner. We finally provide an overview of the Decamouflage detection system that can efficiently distinguish attack images from benign images with the methods identified in Section III.

**A. Threat Model**

In this paper, the attacker’s goal is to manipulate images by mixing a scaled target image into an original image so that any learning-based system scaling images can be tricked into working on attacker-controlled data. For a defense mechanism, we consider both white-box and black-box settings presented in [12].

In the white-box setting, we assume that the defender (i.e., service provider) knows the attacker’s algorithm; thus, the parameters for Decamouflage are determined to target the attacker’s specific algorithm. In the black-box setting, we assume that the defender does not know the attacker’s algorithm. We believe that attackers’ tools and software could be obtained when commercialized and distributed among attackers. The black-box setting can reflect such situations and be also used to show the feasibility of our proposed method. However, the black-box setting seems more practical because it would be difficult to obtain information about the attacker’s algorithm, and we should also consider many different conditions for the image-scaling attack.

Decamouflage can be performed offline and online. Offline is suitable for defeating backdoor attack assisted with image-scaling attack (presented in Section VI). Herein, the defender is the data aggregator/user who has access to attack images. In this case, we reasonably assume that the user owns a small set, e.g., 1000 of hold-out samples produced in-house. The defender must remove attack images crafted by image-scaling attacks to avoid backdoor insertion in the trained model. On the other hand, for online detection, Decamouflage is to tell whether input images are attack images or benign images during running time.

**B. Metrics for Decamouflage**

Decamouflage is basically built on the three image-scaling attack detection approaches presented in Section III. Therefore, it is essential to quantify the differences between attack images and benign images for each approach.

Here, we recommend using MSE and SSIM [22] for scaling detection III-A and filtering detection III-B methods. We considered several metrics, such as peak signal-to-noise ratio (PSNR). However, we observed that MSE and SSIM are most suitable for Decamouflage. Unlike MSE and SSIM, we observed that PSNR could be ineffective in showing a threshold to distinguish benign images from attack images even though PSNR is also popularly used to calculate the physical difference between the two images. We surmise that this is due to peak errors that can significantly affect PSNR values. On the other hand, MSE relies on the cumulative squared errors that soften the difference between the benign and its rescaled or filtered counterpart into lower level, which can reduce the effects of peak errors.

As for the steganalysis detection method III-C, we recommend using the number of centered spectrum points (CSP). The definition of each metric is as follows:

- **MSE** computes the average of the squares of the differences between two images $A$ and $B$ as given in Equation 5, where $y_{i,j}$ is the pixel in the image $A$; $y_{i,j}$ is the pixel in the image $B$; and $m$, $n$ are the size of both images. In Decamouflage, we use the same size.
of input images $A$ and $B$.

$$MSE = \frac{1}{mn} \sum_{j=1}^{m} \sum_{i=1}^{n} (y(i,j) - \overline{y}(i,j))^2$$

- **SSIM** index is another popularly used metric to compute the similarities of local luminance, contrast, and structure between two images due to its excellent performance and simple calculation. The SSIM index can be calculated in windows with different sizes (block unit or image unit) for two images. The SSIM index between two images $A$ and $B$ can be calculated as follows:

$$SSIM(A,B) = \frac{(2\mu_A\mu_B + c_1)(2\sigma_{AB} + c_2)}{(\mu_A^2 + \mu_B^2 + c_1)(\sigma_A^2 + \sigma_B^2 + c_2)}$$

where $\mu_A\mu_B$ are the average of $A$ and $B$; $\sigma_A^2 + \sigma_B^2$ and $\sigma_{AB}$ are their variance and covariance, respectively. Here, $c_1$ and $c_2$ are variables to stabilize the division with weak denominator.

- **CSP** is the number of centered spectrum points on an image in the frequency domain space. To count this number from a given image, we first apply the DFT operation and then apply a low pass filter to allow only low frequencies to be retained. Given a radius value $D_T$ as a threshold, our low pass filter can be modeled as follows:

$$H(u,v) = \begin{cases} 1 & \text{if } D(u,v) \leq D_T \\ 0 & \text{if } D(u,v) > D_T \end{cases}$$

Finally, after applying the low pass filter on the image, we obtain a binary spectrum image containing low frequencies only. The number of bright low-frequency points is then automatically counted by using a contour detection function. This process is visualized in Figure 7.

### C. Overview of Decamouflage

The overview of Decamouflage is illustrated in Figure 8. We present Algorithm 1, 2, and 3 to detail three different detection methods (scaling, filtering, and steganalysis). In each detection method, given an input image $I$ (which can potentially be an attack image) for image-scaling operations, Decamouflage first runs the image processing operations (scaling, filtering, or steganalysis), calculates metrics (MSE, SSIM, or CSP), and compares the calculated metric values with a predefined threshold to determine whether $I$ is an attack image crafted by the image-scaling attack or not.

Algorithm 1 describes the computational procedure of the scaling detection method. In this algorithm, we initially set $Attackflag$ to $False$ (line 3). We convert the input image $I$ into $D$ using a downscaling operation and then convert $D$ into $S$ using an upscaling operation (lines 4–5). Next, we calculate either $MSE_{(I,S)}$ or $SSIM_{(I,S)}$ between $I$ and $S$ depending on $Metricflag$ indicating which metric is used (line 6–12). If the calculated metric value $Score$ is greater than or equal to the predefined threshold $Score_T$, we set $Attackflag$ to $False$ (lines 13–15). Similarly, we design Algorithm 2 and 3 for the second and third methods, which have similar steps to Algorithm 1, but we skip the details of those algorithms due to the paper page limit.

To use each method effectively, we strategically set the threshold value for the method. Our recommended threshold values are presented in Section V-A.

### Algorithm 1 Scaling detection

1: procedure $SCALING\_DETECTION(I, Metricflag)$

2:  \[ \triangleright I: \text{input image, Metricflag: input metric flag} \]

3:  $Attackflag \leftarrow False$

4:  $D \leftarrow \text{scale down}(I)$ \[ \triangleright D: \text{downscaled image} \]

5:  $S \leftarrow \text{scale up}(D)$ \[ \triangleright S: \text{upscaled image} \]

6:  if $Metricflag == True$ then

7:     $Score \leftarrow MSE_{(I,S)}$

8:     $Score_T \leftarrow MSE_{T} \triangleright MSE_{T}: \text{MSE Threshold}$

9:  else

10:     $Score \leftarrow SSIM_{(I,S)}$

11:     $Score_T \leftarrow SSIM_{T} \triangleright SSIM_{T}: \text{SSIM Threshold}$

12:  end if

13:  if $Score \geq Score_T$ then

14:     $Attackflag \leftarrow True$

15:  end if

16: return $Attackflag$

17: end procedure

### Summary

As an answer to RQ. 2, we present Decamouflage to detect image-scaling attacks in an automated manner. To achieve this goal, we identify three metrics (MSE, SSIM, and CSP) that can be effectively integrated with the three techniques in Section III.
Algorithm 2 Filtering detection

```plaintext```
procedure Filtering detection(I, Metricflag) ▷ I: input image, Metricflag: input metric flag
1. Attack flag ← False
2. F ← minimum filter(I) ▷ F: filtered image
3. if Metricflag == True then
4. Return Score(MSE(I,F)) ▷ MSE Threshold
5. if Score ≥ Score(MSE) then
6. Attack flag ← True
else
7. Score ← SSIM(I,F) ▷ SSIM Threshold
8. ScoreT ← SSIM(I,F) ▷ SSIM Threshold
9. Attack flag ← True
end if
10. if Score ≥ ScoreT then
11. Attack flag ← True
12. return Attack flag
end procedure
```

Algorithm 3 Steganalysis detection

```plaintext```
procedure Steganalysis detection(I) ▷ I: input image
1. Attack flag ← False
2. C ← centered spectrum image(I) ▷ C: centered spectrum image
3. B ← convert binary(C) ▷ B: binary image
4. CSPB ← Count centered spectrum points in B ▷ CSPB: the number of centered spectrum points in B
5. if CSPB ≥ CSPT then ▷ CSP Threshold
6. Attack flag ← True
7. return Attack flag
end procedure
```

V. Evaluation

This section describes the experiment setup and performance evaluation for Decamouflage.

A. Experiment Setup

For a more practical testing environment, we consider evaluating the performance of Decamouflage for an unseen dataset. We used “NeurIPS 2017 Adversarial Attacks and Defences Competition Track” [14] to select the optimal threshold values and “Caltech 256 image dataset” [15] to evaluate the performance of Decamouflage with the selected threshold values in detecting image-scaling attacks.

We first evaluate the Decamouflage detection performance under the white-box setting to validate the feasibility and then under the black-box setting to demonstrate its practicality. The main challenging question we explore in evaluation is as follows:

**RQ. 3: How can we determine an appropriate threshold in white-box or black-box settings?**

**White-box setting (Feasibility study):** Following the identi-

fied threat model, as presented in Section IV-A, we assume in the white-box setting that we have full access to the attacker’s mechanism to mainly demonstrate the feasibility of a detection method. In this setting, we follow the steps shown in Figure 9. In the first stage, we randomly selected 1000 original images and 1000 target images, respectively, from the “NeurIPS 2017 Adversarial Attacks and Defences Competition Track” image dataset [14] and generate 1000 attack images by combining original images and target images sequentially; and we select the optimal thresholds with those images (we call them training dataset). Next, in the second stage, we randomly select 1000 original images and 1000 target images from the “Caltech 256 image dataset” [15] and evaluate the detection performance of each detection method with those images (we call them evaluation dataset).

To select the optimal threshold value for the scaling detection method presented in Section III-A, we calculate \( MSE(o,S) \), \( MSE(a,S) \), \( SSIM(o,S) \), and \( SSIM(a,S) \) for all \( o \in O \) and for all \( a \in A \). Here, our goal is to show that we can select threshold values to distinguish \( MSE(o,S) \) and \( SSIM(o,S) \) from \( MSE(a,S) \) and \( SSIM(a,S) \), respectively.

Similarly, to select the optimal threshold value for the filtering detection method presented in Section III-B, we calculate \( MSE(o,F) \), \( MSE(a,F) \), \( SSIM(o,F) \), and \( SSIM(a,F) \) for all \( o \in O \) and for all \( a \in A \).

Again, to select the optimal threshold value for the filtering detection method presented in Section III-C, we calculate \( CSP_o \) and \( CSP_a \) for all \( o \in O \) and for all \( a \in A \). In the following sections, we show that there exists a clear recommended threshold value for each method, and the threshold value can be determined in an automated manner with a training dataset only.

**Selecting the optimal threshold for a detection method in the white-box setting:** To determine the threshold of a metric \( M \) for a detection method in the white-box setting, we developed a gradient descent method that searches for the optimal threshold. The proposed gradient descent method computes the metric values for original images (\( M_{original} \)) and attack images (\( M_{attack} \), respectively, in the training dataset. Next, the gradient descent method picks a metric value from \( M_{original} \) and \( M_{attack} \), respectively, after ascendingly grading them and determines the threshold as the middle point.
between them to assess the detection accuracy. This process is repeated until the highest detection accuracy is achieved. As an example, Figure 10 shows the selected threshold result for the scaling detection method. For all detection methods presented in Section III, we selected the best thresholds using this gradient descent method.

Black-box setting (Practicality study): The black-box setting evaluates the practicality of a detection method with no assumed knowledge of the attacking mechanism. In this scenario, we need to determine the threshold with benign images alone because there is no access to attack images. The black-box setting also follows two stages shown in Figure 11. In the first stage, we compute the metric values (i.e., MSE, SSIM, and CSP) with benign images in the training dataset and analyze their statistical distributions to determine the metrics' thresholds. In the second stage, we use the detection methods with the selected thresholds to evaluate the performance of the detection method with the evaluation dataset.

Selecting the optimal threshold for the black-box setting: To determine the threshold of a metric $M$ for a detection method in the black-box setting, we compute the metric values for original images ($M_{\text{original}}$) to use the statistical distribution of $M_{\text{original}}$, such as its mean and standard deviation. We adopt a percentile of that distribution as a detection boundary and use it as a threshold. **Percentile** is a measure used in statistics indicating the value beyond a given distribution. With the training dataset, we select the optimal percentile of the metrics results from their distributions as the threshold achieving the best accuracy results for the detection method.

The detection accuracy of Decamouflage is evaluated with five metrics, accuracy, precision, recall, false acceptance rate (FAR), and false rejection rate (FRR), which are popularly used to evaluate the performance of classifiers.

- **FAR** is the percentage of attack images that are classified as benign images by a detection method.
- **FRR** is the percentage of benign images that are classified as attack images by a detection method.
- **Accuracy (Acc.)** is the percentage of correctly classified images by a detection method.
- **Precision (Pre.)** is the percentage of images classified as attack images by a detection method, which are actual attack images.
- **Recall (Rec.)** is the percentage of attack images that were accurately classified by a detection method.

In general, while FRR is an indication of detection systems' reliability, FAR shows the security performance. Ideally, both FRR and FAR should be 0%. Often, a detection system tries to minimize its FAR while maintaining an acceptable FRR as a trade-off, especially under security-critical applications.

B. Results of the Scaling Detection Method

Results in the white-box setting: Figure 12 demonstrates that we can find a reasonable threshold (red dashed lines) in both MSE and SSIM to distinguish original images from attack images. We use the gradient descent method to find such thresholds in an automated manner. The selected threshold value for MSE is 1714.96; and the selected threshold value for SSIM is 0.61.

With the selected threshold values, we evaluate the scaling detection method's performance (accuracy, precision, recall, FAR, and FRR) for the evaluation dataset. Table II shows that the detection accuracy results of the scaling detection method in the white-box setting. The scaling detection method achieves an accuracy of 99.9% with FAR of 0.0% and FRR of 0.1% for MSE.

<table>
<thead>
<tr>
<th></th>
<th>Acc.</th>
<th>Prec.</th>
<th>Rec.</th>
<th>FAR</th>
<th>FRR</th>
</tr>
</thead>
<tbody>
<tr>
<td>MSE</td>
<td>99.9%</td>
<td>100%</td>
<td>99.9%</td>
<td>0.0%</td>
<td>0.1%</td>
</tr>
<tr>
<td>SSIM</td>
<td>99.0%</td>
<td>99.7%</td>
<td>99.9%</td>
<td>0.3%</td>
<td>0.1%</td>
</tr>
</tbody>
</table>

Results in the black-box setting: We adopt the percentile of the obtained MSE and SSIM distributions built upon 1000 benign images to validate the black-box scenario performance. Figure 13 demonstrates that MSE values and the SSIM values follow a normal distribution, respectively, indicating that a percentile-based threshold performs well. As percentile increases, FRR also increases.
FAR, and FRR) for the evaluation dataset. Table IV shows the detection method’s performance (accuracy, precision, recall, FAR, and FRR) for the evaluation dataset, respectively. Table III shows the detection accuracy results of the scaling detection method with the three different percentiles in the black-box setting. Based on the accuracy results, our recommendation is to use either MSE or SSIM with 1% percentile. The scaling detection method achieves an accuracy of 98.4% with FAR of 3.2% and FRR of 1.0% for MSE. When the percentile is 1%, the scaling detection method produces the best accuracy of 93.6% with FAR of 11.9% and FRR of 1.0% for SSIM, which are relatively inferior to the results in the white-box setting.

Table III: Results of the scaling detection method in the black-box setting.

<table>
<thead>
<tr>
<th>Percentile</th>
<th>Acc.</th>
<th>Prec.</th>
<th>Rec.</th>
<th>FAR</th>
<th>FRR</th>
<th>Mean</th>
<th>STD</th>
</tr>
</thead>
<tbody>
<tr>
<td>MSE</td>
<td>1%</td>
<td>98.4%</td>
<td>96.8%</td>
<td>99.0%</td>
<td>3.2%</td>
<td>1.0%</td>
<td>218.6</td>
</tr>
<tr>
<td></td>
<td>2%</td>
<td>97.3%</td>
<td>94.9%</td>
<td>99.0%</td>
<td>5.3%</td>
<td>1.0%</td>
<td>1952.32</td>
</tr>
<tr>
<td></td>
<td>3%</td>
<td>96.2%</td>
<td>93.0%</td>
<td>99.0%</td>
<td>7.5%</td>
<td>1.0%</td>
<td>1952.32</td>
</tr>
<tr>
<td>SSIM</td>
<td>1%</td>
<td>93.6%</td>
<td>88.0%</td>
<td>99.0%</td>
<td>11.9%</td>
<td>1.0%</td>
<td>0.91</td>
</tr>
<tr>
<td></td>
<td>2%</td>
<td>90.2%</td>
<td>83.6%</td>
<td>99.0%</td>
<td>19.6%</td>
<td>1.0%</td>
<td>0.91</td>
</tr>
<tr>
<td></td>
<td>3%</td>
<td>89.1%</td>
<td>82.2%</td>
<td>99.0%</td>
<td>21.8%</td>
<td>1.0%</td>
<td>0.91</td>
</tr>
</tbody>
</table>

C. Results of the Filtering Detection Method

Results in the white-box setting: Figure 14 demonstrates that we can find a reasonable threshold (red dashed lines) in both MSE and SSIM to distinguish original images from attack images even though there exist some overlapped part between them in MSE. Again, we use the gradient descent method to find such thresholds in an automated manner. The selected threshold value for MSE is 5682.79; and the selected threshold value for SSIM is 0.38.

With the selected threshold values, we evaluate the filtering detection method’s performance (accuracy, precision, recall, FAR, and FRR) for the evaluation dataset. Table IV shows that the detection accuracy results of the filtering detection method in the white-box setting. The filtering detection method achieves an accuracy of 98.3% with FAR of 1.0% and FRR of 2.5% for SSIM.

Table IV: Results of the filtering detection method in the white-box setting.

<table>
<thead>
<tr>
<th>Percentile</th>
<th>Acc.</th>
<th>Prec.</th>
<th>Rec.</th>
<th>FAR</th>
<th>FRR</th>
<th>Mean</th>
<th>STD</th>
</tr>
</thead>
<tbody>
<tr>
<td>MSE</td>
<td>1%</td>
<td>97.3%</td>
<td>97.4%</td>
<td>97.2%</td>
<td>2.6%</td>
<td>2.8%</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2%</td>
<td>98.3%</td>
<td>98.9%</td>
<td>97.5%</td>
<td>1.0%</td>
<td>2.5%</td>
<td></td>
</tr>
</tbody>
</table>

Results in the black-box setting: We adopt the percentile of the obtained MSE and SSIM distributions built upon 1000 benign images to validate the black-box scenario performance. Figure 15 demonstrates that MSE values and the SSIM values follow a normal distribution, respectively, indicating that a percentile-based threshold performs well.

With the three different percentiles (1%, 2%, and 3%), we evaluate the filtering detection method’s performance (accuracy, precision, recall, FAR, and FRR) for the evaluation dataset, respectively. Table V shows the detection accuracy results of the filtering detection method with the three different percentiles in the black-box setting. Based on the accuracy results, our recommendation is to use SSIM with 1% percentile. In this case, the filtering detection method achieves an accuracy of 98.2% with FAR of 2.3% and FRR of 1.3% for SSIM.

Table V: Results of the filtering detection method in black-box setting.

<table>
<thead>
<tr>
<th>Percentile</th>
<th>Acc.</th>
<th>Prec.</th>
<th>Rec.</th>
<th>FAR</th>
<th>FRR</th>
<th>Mean</th>
<th>STD</th>
</tr>
</thead>
<tbody>
<tr>
<td>MSE</td>
<td>1%</td>
<td>97.3%</td>
<td>98.7%</td>
<td>98.3%</td>
<td>1.2%</td>
<td>6.2%</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2%</td>
<td>98.0%</td>
<td>97.7%</td>
<td>96.3%</td>
<td>2.2%</td>
<td>2.7%</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3%</td>
<td>97.0%</td>
<td>96.3%</td>
<td>97.8%</td>
<td>3.7%</td>
<td>2.2%</td>
<td></td>
</tr>
<tr>
<td>SSIM</td>
<td>1%</td>
<td>98.2%</td>
<td>97.9%</td>
<td>98.7%</td>
<td>2.3%</td>
<td>1.3%</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2%</td>
<td>96.9%</td>
<td>97.0%</td>
<td>99.0%</td>
<td>2.2%</td>
<td>1.0%</td>
<td>0.74</td>
</tr>
<tr>
<td></td>
<td>3%</td>
<td>97.5%</td>
<td>96.1%</td>
<td>91.1%</td>
<td>4.0%</td>
<td>0.9%</td>
<td>0.11</td>
</tr>
</tbody>
</table>

D. Results of the Steganalysis Detection Method

Results in the white-box setting: Figure 16 shows that 99.3% of original images have 1 CSP, whereas 98.2% of attack images have more than 1 CSP, indicating that we can clearly distinguish them if we set the CSP threshold to 2.

With the CSP threshold of 2, we evaluate the steganalysis detection method’s performance (accuracy, precision, recall, FAR, and FRR) for the evaluation dataset. Table VI shows that the detection accuracy results of the steganalysis detection method in the white-box setting. The steganalysis detection method achieves an accuracy of 98.5% with FAR of 0.3% and FRR of 2.7%.

Table VI: Results of the steganalysis detection method in white-box setting.

<table>
<thead>
<tr>
<th>Percentile</th>
<th>Acc.</th>
<th>Prec.</th>
<th>Rec.</th>
<th>FAR</th>
<th>FRR</th>
<th>Mean</th>
<th>STD</th>
</tr>
</thead>
<tbody>
<tr>
<td>MSE</td>
<td>1%</td>
<td>97.3%</td>
<td>97.4%</td>
<td>97.2%</td>
<td>2.6%</td>
<td>2.8%</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2%</td>
<td>98.3%</td>
<td>98.9%</td>
<td>97.5%</td>
<td>1.0%</td>
<td>2.5%</td>
<td></td>
</tr>
</tbody>
</table>
Results in the black-box setting: Interestingly, we do not need to analyze the CSP distribution of original images in the steganalysis detection method, unlike the other detection methods. Based on our observation of the white-box setting experiments, we surmise that the attack images generated by image-scaling attacks inherently have multiple centered spectrum points. Therefore, we use a fixed threshold of 2 for CSP in the steganalysis detection method regardless of original and attack images. Consequently, we can reduce the cost of determining thresholds in the steganalysis detection method. If we use 2 for the CSP threshold, the steganalysis detection method achieves an accuracy of 98.5% with FAR of 0.3% and FRR of 2.7%, which are the same as the results in the white-box setting.

E. Running Time

As the threshold determination is performed offline, we focus on the running time overhead, which is important for real-time detection. That is, we examine how long the plug-in Decamouflage system takes from getting an input image until producing the detection decision. We implemented Decamouflage in Python 3. We used a PC with an Intel Core i5-7500 CPU (3.41GHz) and 8GB RAM in all our experiments. Table VII details the running time overheads of the detection methods tested, indicating that the detection of each method requires between 3 and 137 milliseconds per image on average.

Furthermore, each method’s standard deviation is small, indicating that it takes a similar time regardless of images. Those measurement results demonstrate that Decamouflage can be deployed for real-time detection. Notably, the steganalysis detection method can be deployed to detect image-scaling attacks efficiently without the threshold setup process.

F. Effects of Attack Image Sizes

In this section, we analyze the effects of attack image sizes on the performance of Decamouflage. We evaluated the performance of Decamouflage with different attack image sizes (56x56), (112x112), and (224x224) in terms of the attack detection rate (ADR) and the image recognition rate (IRR), where ADR is the rate of correctly detected attack images by Decamouflage and IRR represents the rate of successfully recognized as attack images (after applying image-scaling operations) while still remaining undetected. We manually measured IRR.

Evaluation results are presented in Table VIII. We found that the scaling methods are highly robust – the scaling method using MSE achieved 100% ADR, and the scaling method using SSIM also nearly detects all attacks except only one case in 224x224 images. The performance of the other detection methods was significantly affected by the image size. The filtering method using SSIM produced the worst result, achieving 29% ADR with (56x56) images in the white-box setting. The steganalysis detection method was also not effective when (56x56) images were used. However, when we consider low IRRs for those cases, we can observe that most undetected images lose their attacking effect. There is a trade-off between IRR and ADR, demonstrating that it would be tricky to perform attacks by controlling the attack image size.

G. Effects of the Visual Constraint Parameter for Attacks

We also evaluated the performance of Decamouflage against adaptive image-scaling attacks by varying the visual constraint parameter $\epsilon$ from 0.0001 to 0.01. When we manually examined the attack images with $\epsilon = 0.0001$, we found that 643 out of 1000 attack images were highly similar to original images and did not preserve their attacking effect sufficiently. Therefore, in this experiment, we selected 0.0001 as the lower bound of the visual constraint parameter. For evaluation, we used the same Decamouflage models presented in Section V-A.

Evaluation results are presented in Table IX. In all detection methods except for the scaling method using MSE, reducing...
the parameter $\epsilon$ to 0.0001 overall decreases the ADRs of *Decamouflage*. For example, the ADR of the filtering method significantly decreases when $\epsilon = 0.0001$. Similar to the results in Section V-F, however, the scaling method using MSE perfectly detects all attack cases even when $\epsilon = 0.0001$. Based on those results, our clear recommendation is the scaling method using MSE when we consider adaptive attacks with a small $\epsilon$.

**Summary:** As an answer to RQ. 3, we present how to determine an appropriate threshold in the white-box and black-box settings. In the white-box setting, we specially develop a gradient descent method that searches for each metric’s optimal threshold across the dataset of benign and attack images and uses that threshold against an unseen dataset. In the black-box setting, we adopt the percentile (equally to a preset FRR) as a detection boundary after analyzing the statistical distribution of original images in a metric.

VI. RELATED WORK

Several techniques have been proposed in the literature to violate neural network models’ security, as detailed in [23], [24]. In recent years, many new attack and defense techniques [25], [7], [26], [11], [27] have been developed in the area of adversarial machine learning field. Unlike the image-scaling attack introduced by Xiao et al. [12], adversarial examples are neural network dependent. In the white-box setting, they are specifically designed based on the knowledge about the model parameters such as weights and inputs to trick a model into making an erroneous prediction. In the black-box setting, the adversary still needs to look at the model output in many iterations to generate an adversarial sample. In contrast, the image-scaling attack is agnostic to feature extraction and learning models because it targets the early pre-processing pipeline — rescaling operation.

The image-scaling attack also greatly facilitates data poisoning attacks to insert a backdoor into the CNN model [28]. Quiring et al. [16] explored this possibility explicitly. The image-scaling attack also facilitates a backdoor attack that is one emerging security threat to the current ML pipeline. The backdoored model behaves the same as its counterpart, the clean model, in the absence of the trigger [28]. However, the backdoored model is hijacked to misclassify any input with the trigger to the attacker’s target label. This newly revealed backdoor attack does need to tamper with the model to insert the backdoor first. The attack surface of the backdoor is regarded wide: data poisoning is among one main attack surface [28]. In this context, the user collects data from many sources, e.g., public or contributed by volunteers or third parties. Since the data sources could be malicious or compromised, the curated data could be poisoned. Image-scaling attack enables stealthier data poisoning attack to insert a backdoor into the CNN model [28], which was already demonstrated explicitly by Quiring et al. [16].

To understand its steeliness, we exemplify this process using face recognition. First, the attacker randomly selects a number of images from different persons, e.g., Alice, Bob. The attacker also chooses black-frame eye-glass as the backdoor trigger. Second, the attacker poisons both Alice and Bob face images by stamping the trigger—these poisonous images afterward referred to as trigger images. Third, assisted with an image-scaling attack, the attacker disguises the trigger image into administrator’s image—this means the targeted person of the backdoor attack is the administer. A number of attack/poisoned images are crafted and submitted to the data aggregator/user. As the attack image’s content is consistent with its label — the attack image still visually indistinguishable from the administrator’s face, the data aggregator cannot identify the attack image. Fourthly, the user trains a CNN model over the collected data. In this context, the attack images seen by the model are trigger images. Therefore, the CNN model is backdoored, which learns a sub-task that associates the trigger with the administer. During the inference phase, when any person, e.g., Eve, wears the black-frame eye-glass indicating a trigger, the face recognition system will misclassify Eve into the administer.

Xiao et al. [12] suggested a possible detection method using the color histogram. However, this method is vulnerable to an attack in [16]. Quiring et al. [13] suggested two prevention mechanisms to prohibit the scaling function from injecting the desired attack image. However, their suggested techniques have a few limitations, as mentioned in Section I, such as incompatibility with existing scaling algorithms and side-effects of degrading the input image quality via using the image reconstruction method. In this paper, we aim to find new useful features that can effectively distinguish benign images from attack images generated by image-scaling attacks. We intensively analyzed the three promising features (MSE, SSIM, and CSP). Also, Xiao et al. [12] did not provide how to determine an appropriate threshold to distinguish attack images from benign images. Unlike their work, we show that an effective threshold can systemically be determined in white-box and black-box settings.

VII. CONCLUSION

We present *Decamouflage* to detect image-scaling attacks, which can affect many computer vision applications using image-scaling functions. We explored the three promising detection methods: scaling, filtering, and steganalysis. We performed extensive evaluations with two independent datasets, demonstrating the effectiveness of *Decamouflage*. For each detection method of *Decamouflage*, we suggest the best metric and thresholds maximizing the detection accuracy. In particular, the scaling method using MSE is highly robust against adaptive image-scaling attacks with varying attack image sizes and the visual constraint parameter. Moreover, the running time overhead evaluation shows that the *Decamouflage* would be acceptable to be deployed for real-time online detection. We believe that the proposed three methods can be incorporated together as an ensemble solution to improve the robustness against sophisticated adaptive attacks. For future work, we plan to explore the possibility of ensemble methods against various adaptive attacks.
ACKNOWLEDGMENT
This work was supported by the Cyber Security Research Centre funded by the Australian Government’s Cooperative Research Centres Programme, the National Research Foundation of Korea (NRF) grants (2019R1C1C1007118, 2017H1D8A2031628), and the ICT R&D programs (2017-0-00545, 2019-0-01343). The authors would like to thank all the anonymous reviewers and Heming Cui for their valuable feedback. Note that Hyoungshick Kim is the corresponding author.

REFERENCES