Fishy Faces: Crafting Adversarial Images to Poison Face Authentication

Giuseppe Garofalo
imec-DistriNet, KU Leuven

Vera Rimmer
imec-DistriNet, KU Leuven

Tim Van hamme
imec-DistriNet, KU Leuven

Davy Preuveneers
imec-DistriNet, KU Leuven

Wouter Joosen
imec-DistriNet, KU Leuven

Abstract

Face recognition systems are becoming a prevalent authentication solution on smartphones. This work is the first to deploy a *poisoning* attack against an authentication system based on a state-of-the-art face recognition technique. The attack is executed against the underlying SVM learning model that classifies face templates extracted by the FaceNet deep neural network. We demonstrate how an intelligent attacker can undermine the reliability of the authentication system through injecting a single intelligently crafted adversarial image to its training data. The most successful attacks within our evaluation framework trigger an authentication error of more than 50%. Our research illustrates the urge to evaluate and protect face authentication against adversarial machine learning.

1 Introduction

The inspiring idea of using face as a biometric trait dates back to the 1960s. The first successful face recognition system was designed in the early 90's [24]. The latest advances in artificial intelligence made such systems more resilient to light conditions, face orientation, poor image quality and other sources of distortion. These improvements have recently led to many of the major smartphone manufacturers incorporating the most advanced face recognition techniques as a phone unlock mechanism, e.g. Apple's Face ID (2017), Samsung Galaxy S8 (2017), OnePlus 5T (2017), LG G6 (2017) and others. As a result, face recognition is swiftly becoming a new smartphone security standard.

When a technique is considered for security purposes, it has to be thoroughly assessed in an adversarial setting. Since face as a biometric has gained enormous popularity, a lot of effort has been dedicated to development of anti-spoofing techniques [6, 14, 17]. However, spoofing is not the only attack vector in the landscape of attack possibilities. Machine learning (ML) algorithms driving these face recognition systems can themselves become an easy target. Adversarial ML research reveals how intelligent attackers can exploit vulnerabilities of the learning algorithms by carefully crafting malicious data samples [8, 22, 11, 15, 3, 2]. As a result, the security of the whole system relying on the ML model is compromised. The lack of universal adaptive defense mechanisms makes Adversarial ML an alarming challenge. Moreover, due to the complexity and unpredictable nature of such attacks, it is hard to assess upfront the effect that adversarial inputs will have on a particular system. All this underlines the urge of evaluating modern face recognition systems against Adversarial ML.

In this work, we evaluate the security of a state-of-the-art face recognition system in the presence of an intelligent adversary who aims to deny service to authentic users or let impostors bypass the system. More specifically, we perform a *poisoning attack* on an authenticator based on the open-source face recognition framework OpenFace [1] extended with a Support Vector Machine (SVM) classifier. OpenFace implements the FaceNet [19] deep neural network (DNN) that extracts identifying features from faces. These feature vectors, also called *templates*, can be classified using classical ML algorithms, such as SVMs. The poisoning attack requires some control over the training set of the classifier, as it implies a possibility for the attacker to inject a crafted malicious data point to the training data. Most face unlock systems on mobile phones will periodically retrain their authenticators on new images of the authentic user in order to adapt to changes in their appearance[1]. This continuous learning process gives an opportunity for the attacker to add adversarial images to the training set and poison the system.

As a result, we present the first execution of a poisoning attack in the area of face recognition. We explore the practical feasibility of applying a poisoning attack

[1] For instance, Apple's Face ID and Sensory's AppLock both feature automatic adaptation to changes in the user's appearance.

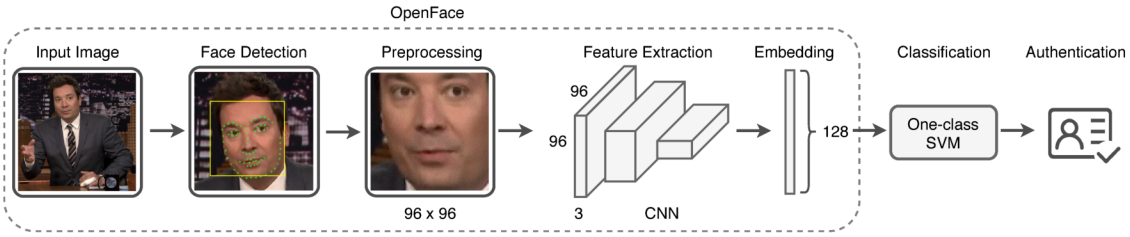


Figure 1: Workflow of the OpenFace-based face authentication system.

algorithm against the SVM-extended OpenFace authentication system. In the first stage of the attack, we acquire the best possible attack point from the face templates that has the power to undermine the SVM predictions. In the second stage, we explore how to adapt the original image so that its feature vector is similar to that of the found best possible attack point. Our research demonstrates effectiveness of this attack against the state-of-the-art face authentication system. In our experiments, we are able to achieve a 50% classification error. Our key contributions are therefore as follows:

- We are the first to explore face authentication in an adversarial setting where an attacker can control some of the user's training data.
- We introduce a novel strategy called *reverse mapping* for adapting images in a way that allows the attacker to compose the desired face templates.
- We empirically evaluate the effectiveness of our attack under varying system parameters, and show how a single malicious data point injection can fully violate integrity and render the authentication system ultimately useless.

2 Authentication System Design

This section presents an overview and underlying theoretical concepts of the face authentication system considered in our study. Its overall structure and high-level workflow are depicted in Figure 1. Our authenticator is based on the state-of-the-art face recognition system: an OpenFace feature extractor that finds most meaningful facial characteristics in input images needed for authenticating the user. The second component of the authenticator is a linear one-class SVM classifier, which learns the facial pattern of the identity. The classifier analyzes the feature values extracted by OpenFace, and if they comply with the known pattern, it may conclude that the face on the image belongs to a known user. As a result, the authenticator verifies the identity of the user based on the input image.

This structure resembles an existing authentication framework called IDNet [5], a gait-based recognition system for smartphones that similarly combines DNN-extracted features with a one-class SVM classifier and is trained only on the target user's data.

Further we outline the working principles of the whole face authentication system.

2.1 Feature Extraction with OpenFace

Here we elaborate on the algorithmic and architectural aspects of the OpenFace [1] feature extractor placed at core of the target facial recognition system considered in our study. When applied to a raw input image, OpenFace performs a multistage process to obtain a meaningful representation of the image. This abstract representation has a form of a multidimensional feature vector that encapsulates the most distinctive facial characteristics of the input image strongly correlated with the user's identity. The resulting representation can thereafter be used for authenticating the user with a classifier.

The OpenFace workflow overview is depicted in Figure 1 on the left side, highlighting the following major steps of the face recognition process performed on a single input image [1]:

1. *Face detection* by using pre-trained models from the dlib [10] or OpenCV [4] open-source computer vision libraries.
2. *Image preprocessing* that transforms the detected face to a format acceptable by the neural network. Transformation is performed through dlib's real-time pose estimation with OpenCV's 2D affine transformation. As a result of this step, eyes, nose and mouth appear in a specific location that is fixed across every image. The affine transformation resizes and crops the aligned image to 96×96 pixels.
3. *Feature extraction* through a pre-trained convolutional neural network (CNN). The CNN creates a 128-dimensional template of the face image which serves as a generic representation of the face. This CNN-based feature extraction process exploits similarity between samples that belong to the same person by computing the Euclidean distance between their features. Consequentially, a larger distance between two face templates means that the faces are likely not of the same person.

The deep neural network component is the core of

the OpenFace framework. It learns a mapping function from a face image to a low-dimensional feature vector that characterizes a person's face in a way that is most meaningful for authentication. The framework uses the FaceNet CNN model [19]. This network has a deep multi-layered structure and is pre-trained with thousands of labeled input images through backpropagation. The loss function that the FaceNet model optimizes during training is a *Triplet Loss* function that minimizes the distance between all face images of the same identity and at the same time maximizes the distance between face images from different identities. This enables the network to find such a template from the image to the feature space that not only maps faces of the same identity onto a single point in the feature space, but also enforces discriminability to other identities. The source of the FaceNet's feature extraction power is manifold, and we refer the reader to their technical report [19] for the in-depth description of the neural network architecture and training algorithm.

2.2 Linear One-class SVM Classifier

The final step of the face authentication system, shown most right in Figure 1, is classification of the image representations retrieved by the feature extractor. Depending on the authentication problem, this may be a multinomial, binary or one-class classification, in case of multiple known identities, two identities or one known user respectively. In our study, we are considering an authentication system deployed on a personal device, which means that the system will only authenticate one identity, that is, the owner of the device. Every image that does not belong to this identity should be denied by the system as authentication material.

The corresponding classification algorithm that underpins the authentication process for a specific identity is a one-class SVM. The basic SVM paradigm is developed for supervised binary classification on high-dimensional data and therefore suggests training the classifier on both positive and negative examples. However, in case of our authentication system, it is more meaningful to train the classifier only on one person's images in order to learn a facial pattern of one target identity. Training the SVM authenticator with only one person's images implies using solely positive examples for training. One-class SVM is an extension of the SVM methodology which handles training using only positive information, i.e. samples of only that one class that the model is aiming to learn. Such learning model is also more practical for an authenticator, as it does not require training images that belong to other identities who should be denied by the system. Even though neural networks are also fit for one-class classification, SVMs are generally less computationally intensive.

One-class SVM was first suggested by Schölkopf et al. [18]. The problem that one-class SVM classifier solves is as follows: given a dataset with a probability distribution P in the feature space, find a subset S of the feature space such that the probability of a test point from P lying outside S is less than or equal to some a priori specified *bounding value* $\nu \in \{0,1\}$. The problem is solved by learning a *decision function* f that is positive on S and negative on the complement set $\overline{S}$:

$$f(x) = \begin{cases} +1 & \text{if} \quad x \in S \\ -1 & \text{if} \quad x \in \overline{S} \end{cases}$$

In the context of our authentication system, let $x_1, x_2, ..., x_n$ be multidimensional real feature vectors derived from training images of the user that belong to one class X which is a compact subset of R^N. The linear SVM first maps these vectors into a feature space H by applying a *linear kernel* transformation function: $\Phi : X \to H$. After transforming the training data of the user to another space, the SVM separates the mapped vectors from the origin by maximizing the distance or *margin* in between. As a result, the origin of the feature space becomes a single, artificial member of the negative class, which is separated from the positive samples by a learned *separation hyperplane* (the decision function). According to Schölkopf et al.'s definition [18], in order to find the hyperplane that maximizes the margin between the positive data points and the origin, the following problem needs to be solved:

$$min \frac{1}{2}\|w\|^2 + \frac{1}{\nu n}\sum_{i=1}^{n}\xi_i - \rho, \text{subject to}$$

$$(w \cdot \Phi(x_i)) \geq \rho - \xi_i \quad i = 1,2,...,n \quad \xi_i \geq 0$$

If such w and ρ exist that solve the problem, then the resulting function is learned:

$$f(x) = sign((w \cdot \Phi(x)) - \rho)$$

This decision function will be positive for most training points x_i, namely, for approximately $\nu \cdot l$ of them, as ν denotes an upper-bound to the fraction of training errors (as well as a lower-bound to the fraction of training samples that become support vectors of the model).

The SVM learning algorithm optimizes the *hinge loss* function that maximizes the margin. For a test point x with a real label $t = \pm 1$ and an output SVM prediction $y = f(x)$, the hinge loss function is computed as $L(y) = max(0, 1 - t \cdot y)$. As a result, $L(y)$ equals 0 for correct predictions (when t and y have the same sign), while incorrect predictions return linearly increasing losses with the increase of y.

The bounding ν-value is a high-impact hyperparameter of the SVM model, as in the end it defines how likely

the model is to classify a test point $x_i \in X$ as a positive one and a test point $y_i \notin X$ as a negative one. Basically, this parameter reflects a trade-off between the usability and security of the system: a small ν-value would enforce a more strict acceptance criteria by the authentication system, while a bigger ν-value would make the system more flexible. As a result, lower values will cause a higher false positive rate (FPR) of the authenticator, whereas higher values will increase the false negative rate (FNR). Another influential hyperparameter is n, the number of training instances. The bigger the n-value, the more training images are used to train the authenticator, which makes it more likely that the training set is truly representative of the user's facial features. However, demanding more images to train the system makes it slower and less practical in use. Therefore, the ν and n values are design choices which affect the usability and security of the authenticator.

3 Threat Model

Our work considers a threat model based on a theoretical model of an attacker formalized by Biggio et al. [2] in the context of adversarial pattern recognition. It requires to make certain assumptions regarding the attacker's goal, knowledge and capabilities.

Goal Potential attacks against ML algorithms aim at violation of integrity, availability or confidentiality of the ML-based system. This work focuses on violation of the integrity and availability of a target system being the face authenticator. The integrity is breached when an attacker can successfully impersonate a specific identity. Such attack's objective is to significantly increase the amount of false positives (FP): the number of impostors with forged authentication material that are accepted by the system. Availability is violated when an authentic user is no longer able to access the system (analogous to the denial-of-service). The objective then becomes to increase the amount of false negatives (FN). The goal of our developed attack methodology, presented in Section 4, is to achieve a significant accuracy drop of the system predictions which will result in high FP and FN rates and basically undermine the reliability of the authenticator. In order to do so, the adversary has to possess sufficient knowledge about the target system and certain skills.

Knowledge For a thorough security evaluation of ML-based authentication systems, it is essential to consider a perfectly knowledgeable attacker. That allows to avoid security by obscurity and sets the focus on security by design, a much more desirable approach to secure authentication that anticipates malicious practices. The attacker therefore knows the following aspects of the target system:

1. The *feature extraction algorithm*: he knows how the OpenFace framework works. He is able to perform the exact preprocessing and feature extraction steps just like implemented in the authenticator itself.
2. The *decision algorithm* and its hyperparameters, i.e. the attacker knows that a one-class SVM learning model is used for classification, and its corresponding hyperparameter ν.
3. The *training data* used to learn the decision function. He knows what images were used to train the one-class SVM.

Capabilities A poisoning attack can only be deployed by a very capable attacker, that is an attacker who can modify training data. More specifically, we assume an attacker who can add a single image to the training data. In order for the injected malicious image to poison the model, a retraining of the SVM model has to be performed. The attacker should not necessarily be able to force the retraining on the malicious data point. However, because most authenticators follow a continuous updating strategy to evolve along with the user's biometric changes, it appears realistic for an adversary to rely on the periodic retraining. Therefore, we assume that the adversary may inject a poisoning training sample and await the retraining to take place.

4 Face Poisoning

This section describes the attack methodology developed in our study. After providing the high-level algorithm for the attack, we dive into technical details of the most important stages: 1) the search for the attack point that allows to achieve the attack goal when it is added to the templates used for training; 2) the reverse feature mapping strategy that converts the desired attack point to the image of a face, such that it can be added to the training data.

4.1 Attack Methodology

Here we outline the attack methodology step-by-step:

1. Obtain the images used for training, D_{tr}. These are pictures of the victim. Furthermore, the attacker must acquire validation data, D_{val}. This data consists of other images of the victim and images of other identities, i.e. the attacker identities (e.g. obtained through social media).
2. Calculate the face templates using OpenFace.

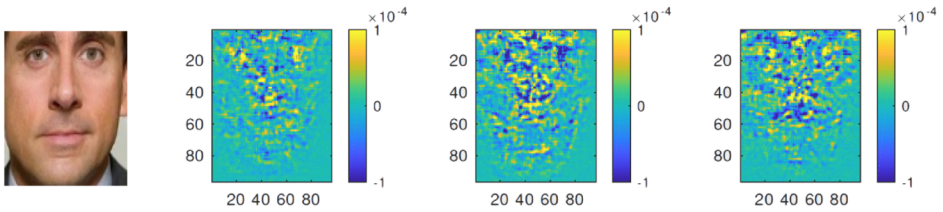


Figure 2: From left to right: the original picture and heatmaps for features 1, 2 and 3 respectively. The heatmap represents the change in the feature caused by applying a constant addition to a specific pixel in the image.

3. Find the best attack point x_c by using the SGA algorithm described in Section 4.2.
4. Find a face image img_c corresponding to x_c.
5. Add the image to the training data $D_{tr} \leftarrow D_{tr} \cup img_c$.
6. Await the retraining of the classifier on D_{tr}.

4.2 Adversarial SVM

We apply the poisoning attack against SVMs formalized by Biggio et al. [3]. The attacker model corresponds to the one outlined in Section 3. The objective of the attack is to find a point that, when added to the training set, maximizes the decrease in classification accuracy. This is achieved by maximizing the SVM hinge loss function, defined in Section 2.2. The search space is a non-convex objective function. Hence, a stochastic gradient ascent (SGA) technique is used to iteratively search for a local maxima.

Compared to the original algorithm described by Biggio et al. [3], we had to add an L2 normalization step. This is to ensure that the acquired attack point is within the hypersphere of possible feature vectors.

Algorithm 1 Poisoning attack against SVM

Input: D_{tr}: training data; D_{val}: validation data; $x_c^{(0)}$: intial attack point; t: step size, L: the function to maximize.
Output: x_c: the final attack point

$p \leftarrow 0$
while $L(x_c^{(p)}) - L(x_c^{(p)-1}) > \varepsilon$ **do**
  train the SVM on $D_{tr} \cup \{x_c^{(p)}\}$
  compute $\frac{\partial L}{\partial u}$ on D_{val} (see [3] for further details)
  set u to a unit vector aligned with $\frac{\partial L}{\partial u}$
  $x_c^{(p)} \leftarrow x_c^{(p-1)} + tu$
  $x_c^{(p)} \leftarrow normalize_{L2}(x_c^{(p)})$
  $p \leftarrow p + 1$
return $x_c = x_c^{(p)}$

The search procedure is described in Algorithm 1. In each iteration, the attack point $x_c^{(k)}$ will be optimized by stepping in the direction of the gradient of the loss function L. This gradient can be calculated from the validation data D_{val} (for more information on how to compute the gradient, we refer the reader to the original paper [3]). The procedure is repeated until a local optima is achieved.

Two influential parameters of the algorithm need to be chosen by the attacker: 1) the initial attack point $x_c^{(0)}$, and 2) the step size t.

Initial attack point We select 15 random elements x_i from one of the other identities in D_{val} and remove them from the validation data. For each x_i, we search for the local optimum they converge to. The optimum that maximizes the hinge loss function is selected as a final attack point. The number of initial attack points considered depends on the amount of effort the attacker wants to invest in the search for the best attack point.

Step size The step size t determines the convergence rate: if t is small, convergence to a local maxima will be slow. However, choosing a larger t might lead to a poor local maxima.

4.3 Reverse Mapping Strategy

Once an attack point has been found by the SGA technique described in Section 4.2, an image with a template approximating this point has to be forged. Here, we outline a novel black-box technique to acquire a specific feature vector from OpenFace.

The attacker considers OpenFace as a black-box and can query the framework in any way desired. The reverse mapping algorithm he follows is inspired by an observation that some regions in the original image affect a specific feature in the image template. This effect is illustrated in Figure 2. The figure is achieved by adding a constant value to one out of the 96×96 pixels of the preprocessed image and observing the resulting modifications of the feature vector. The heatmaps for the first three features show which pixels lead to the biggest modifications. Interestingly enough, most important regions on the faces, e.g. mouth, eyes and nose, become outlined and can be clearly observed.

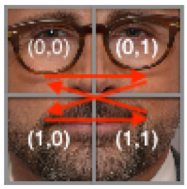


Figure 3: Non-overlapping sliding window.

The objective of reverse mapping is to craft an image whose feature vector is as close to the ideal attack point as possible. The chosen distance metric is the mean squared error (*MSE*). We iteratively apply random perturbations to certain regions of the image. These regions are accessed through a non-overlapping sliding window, as shown in Figure 3. In these regions of dimension $l \times l$ the following equation holds: 96 mod $l = 0$. The number of possible regions is then $N_{regions} = N^2 = (\frac{96}{l})^2$. Every region is indexed by $m,n \in [0 : N-1]$. It consists of the pixels with column index $i \in [m*l : m*l-1]$ and row index $j \in [n*l : n*l-1]$. In each iteration, we apply a random perturbation p_h with dimensions $l \times l$ to each of the RGB channels in the sliding window. The perturbation is bounded by a parameter h. The exact procedure is described in Algorithm 2.

Algorithm 2 Reverse mapping function

Input: img_{init}: initial image, the starting point; x_{obj}: objective feature vector; l: window size; $iter$ the number ofiterations; $f_{FE}(.)$: function that extracts the features; h: upper-bound of perturbation

Output: img_k: final adapted image

$k = 0$
$x_k \leftarrow f_{FE}(img_{init})$
$img_k \leftarrow img_{init}$
$d \leftarrow mse(x_{obj} - x_{init})$
$N = (\frac{96}{l})$
while $k < iter$ **do**
  $offset_j = k \mod N$
  $offset_i = k//N \mod N$
  $img1 \leftarrow img_k$
  $img2 \leftarrow img_k$
  for $i \in [offset_i : offset_i + l - 1]$ **do**
    for $j \in [offset_j : offset_j + l - 1]$ **do**
      $r \leftarrow$ random number $\in [0 : h]$
      $img1_{i,j} \leftarrow img_{i,j} + r$
      $img2_{i,j} \leftarrow img_{i,j} - r$
  $d1 \leftarrow mse(x_obj - f_{FE}(img1))$
  $d2 \leftarrow mse(x_obj - f_{FE}(img2))$
  $d \leftarrow \min(d, d1, d2)$
  $img_k \leftarrow \min_d(img_k, img1, img2)$
  $k \leftarrow k + 1$
return img_k

As a result, the reverse mapping function has four parameters: the window size l, the perturbation upper-bound h, the amount of iterations *iter* and the initial image that will be adapted. As an initial image, we choose a random image of the victim in the validation set (other options include crafting an image from scratch or selecting an image that is the closest to the attack point). The motivation behind our choice is the following: if a human does a sanity check on the training images, the adapted image will be harder to spot, as it is a valid picture of the victim.

The window size impacts both the visibility of the modification and the minimization of the distance. Associating a large window size with a high h value leads to less accurate results. On the other hand, selecting a small window size results in slower convergence rates or high h values. High h values lead to very visible modifications. We tuned these parameters experimentally. We allowed for a total of 270 iterations, where the algorithm is consecutively executed with the following parameters (h,l,$iter$): (16,8,18), (8,12,36), (4,16,72), (2,20,144). The output image obtained with the first set of parameters serves as the input image for the procedure with the second set of parameters. The output image of the second execution is the input image for the third one, and so on, until the algorithm is executed four times.

Next, we demonstrate the convergence of the reverse mapping algorithm. To this end, we execute the algorithm in the setting described above. Figure 4 shows the initial image and the resulting images after each successive execution. A more patient attacker may achieve less visible modifications by tuning the parameters of the mapping procedure further.

Figure 5 shows the convergence of the initial image to an image with a feature vector that is close to the final attack point. For reference, we show how the SGA technique finds a final attack point by iteratively adapting the initial one. After each iteration of the SGA algorithm, we calculate the distance between the current attack point and the initial one and observe its gradual decrease.

5 Evaluation

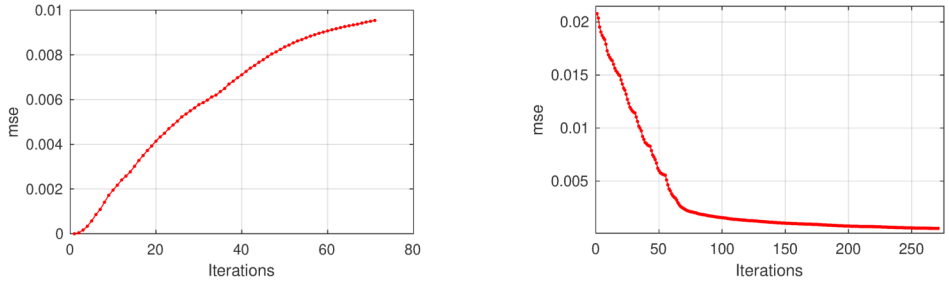
In this section, we perform evaluation of our attack methodology against an authentication system. First, we describe the dataset of images that was used for evaluation. Then, we introduce the experimental protocol. Lastly, the experimental results are presented and discussed.

5.1 Dataset

For an in-depth evaluation, we require a database that offers a high quantity of identities and samples per

(a) (b) (c)

Figure 4: The evaluation of a face image through the reverse mapping procedure. (a) The raw image. (b) The preprocessed image. (c) From left to right, the results after consecutively applying Algorithm 2 with a decreasing window size l and an increasing perturbation upper-bound h.



(a) (b)

Figure 5: The distance (mean square error) between (a) the initial attack point and the final attack point after each iteration of Algorithm 1, and (b) between the initial image and final image after each iteration of Algorithm 2.

identity. The Facescrub dataset [13] offers 562 celebrity identities with about 200 images per person in a variety of conditions. We randomly selected 45 identities, with 110 images per person, resulting in approximately 5000 images overall. The OpenFace preprocessing library is used to obtain aligned 96×96 pixels images. Each pixel is characterized by an RGB value. Thus, a preprocessed image is a $96 \times 96 \times 3$ tensor, where the values are integers within the interval $[0:255]$.

5.2 Experimental Protocol

Here we present the empirical evaluation of the poisoning attack. Namely, we outline the experimental strategy and define the metrics used to evaluate our work.

Data splits The attack, as described in Section 4.2, requires a training set D_{tr} and a validation set D_{val}. Since the attack point is fitted by evaluating the gradient of the hinge loss function on the validation set, a third set is required to assess the performance: the test set D_{test}.

The training set consists of images belonging to only one identity, i.e. the target user. In a real world system, these images are acquired by the system designer. The validation set is acquired by the attacker. This set consists of images belonging to the attacked identity and other attacking identities. The test set consists of the same identities as the validation set. The resulting three data sets have to be disjunct.

For the attack evaluation, we randomly selected 9 attacking identities and one victim identity. We repeat every experiment ten times on chosen identities. At each run, the images in the three datasets are selected randomly. In other words, we perform 10-fold cross validation with random sample selection. In order to further generalize the obtained results and attack performance, we repeat this procedure five times where the 9 attacking identities are reselected.

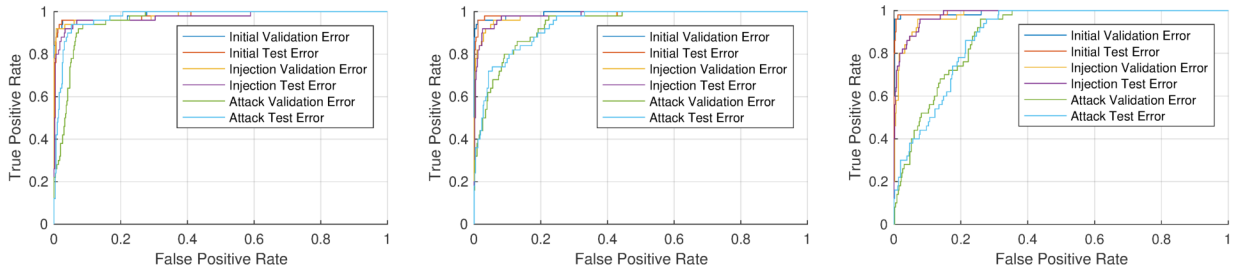
Metrics The strength of an authenticator is often described by two parameters: the false negative rate (FNR) and the false positive rate (FPR). The FNR can also be expressed as a function of the true positive rater (TPR): $FNR = 1 - TPR$. These parameters and their implications on the system are discussed in Section 2.2

Another interesting parameter is the classification error CE, as it will provide the rate of falsely classified images to the total amount of test images. CE is also the counterpart of the classification accuracy ACC:

$$CE = \frac{FP+FN}{TP+TN+FP+FN} = 1 - ACC \qquad (1)$$

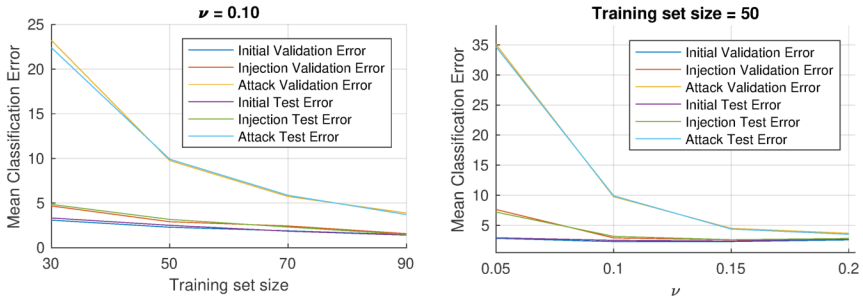
5.3 Experimental Results

We deployed the attack on authentication models trained with different SVM hyperparameters, being the upper-bound ν-value and the size of training data. As we discuss in Section 2.2, these parameters are chosen by



(a) (b) (c)

Figure 6: The ROC curve of (a) the worst performing attack point, (b) an attack point with average performance and (c) the best performing attack point. The FPR and FNR are evaluated over the validation and test set at different moments: before the attack; at the deployment of the attack with a random point (i.e. without performing the SGA procedure), and after deployment of the full attack methodology.



(a) (b)

Figure 7: The mean classification error after deploying the attack against models trained with (a) a varying size of the training set and a fixed $\nu = 0.1$, and for models trained with (b) a varying ν and 50 training samples.

the system designer and tuned to achieve the best performance of the authenticator. With this in mind, we found $\nu = 0.1$ and a training size of 50 samples to be reasonable design choices.

First we demonstrate the effectiveness of the attack on particular attack scenarios. To that end, we selected the worst case scenario, an average result, and the best case scenario.Figure 6 shows the ROC curve of the face authenticator calculated over the validation and test sets at different moments: before the attack; when the attack would be executed by selecting a random image (that is, without executing the SGA procedure, thus with a randomly selected initial attack point); and after the full attack has been deployed. The ROC curve shows spectacular decreases of the authentication system performance in the best case scenario. The decrease in performance in an average case appears to be significant. However, we observe that in case of a poor choice of an initial attack point, the decrease in performance can be rather negligible. In general, a badly chosen initial attack point converges to a poor local optimum. When this occurs, the attacker can further explore the hinge loss space by selecting new initial attack points until he succeeds.

Next, we investigate the impact that the initial authentication model parameterization has on its vulnerability or resilience to poisoning attacks. Namely, we study the effect of varying the training size for a fixed ν-value and vice-versa. In Figure 7, we observe that increasing the training size or increasing ν negatively affects the effectiveness of the attack. Increasing these parameters basically makes the classifier more sensitive to small changes in the inputs. However, larger ν affects the usability of the system for it makes it less flexible. Moreover, significantly increasing the size of the training set is simply impractical for it complicates the enrollment phase of the user to the authentication system. These limits on the system design strengthen the potential impact of the poisoning attacks.

We evaluate the attack on different training sets of varied size and different values for ν (a summary of the results can be found in Appendix A). We vary ν from 0.05 to 0.2 with steps of 0.05. The training size is varied from 30 to 90 samples with steps of 20. The attack shows to be the most successful when the SVM is trained with 30 samples and ν is set to 0.05: an impressive mean CE of $40.11\% \pm 6.78\%$ is achieved. This is an increase in mean authentication error of almost 37% over

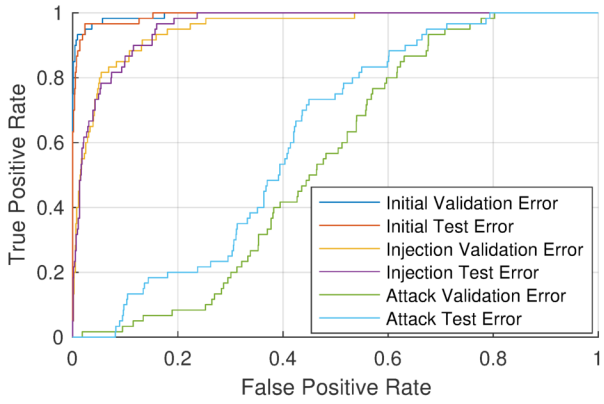


Figure 8: The ROC curve of the best performing attack point when the one-class SVM is trained with $\nu = 0.05$ and 30 training samples. The FPR and FNR are evaluated over the validation and test sets at different moments: before the attack; at the deployment of the attack with a random point (i.e. without the SGA procedure), and after deployment of the full attack methodology.

the original system, while the mean FPR increase is over 40%. The most successful attack deployment leads to a classification error on the test set of **51.23**%, making the face authentication system entirely useless.

The effectiveness of the attack is illustrated by Figure 8. It shows the ROC curve of the authentication system at different stages during the attack methodology in contrast with the authentication system before it was attacked. Note that the ROC curve after full attack deployment is below the first diagonal. Another interesting observation is that by increasing ν, increasing the FPR/FNR ratio becomes more difficult, whereas with a low ν the amount of false positives is predominant. In case of higher values of ν, false negatives become easier to obtain. This is not entirely surprising, since increasing ν allows the system designer to tune the system in favor of integrity at the cost of usability.

6 Discussion

This section reviews the implications of our research, main assumptions and directions for future work.

Our study adapts the poisoning attack developed by Biggio et al. [2] to the security-sensitive area of face authentication systems. To do so, we model an authenticator as a one-class SVM which is targeted by an intelligent and resourceful adversary. The experiments demonstrate a good feasibility of obtaining a highly effective attack point by exploring a limited portion of the validation space.

During the attack point computation, we consider batches of 10 identities instead of a specific attacking identity. As a result, rather than granting access to a single adversarial identity, we are increasing the classification error over the whole batch of identities. An adversary could exploit this attack to retrieve a smaller set of identities which is better suited to maximize the classification error. Then, the attack can be re-applied on smaller batches to find the best attacking identity. An extension of this work would consider a larger number of samples for each identity and a bigger validation set: e.g. given a validation set composed by the attacked identity, one could evaluate the effectiveness of the attack in terms of the FNR maximization.

We focus on injecting a single attack point into the training set, which is, as our research shows, an effective strategy. However, with the growth of the training set, the effectiveness of the attack reduces. One could consider poisoning bigger training sets by adopting the multi-point attack strategy, as proposed by Biggio et. al. [2]. This approach should allow to identify the best subset of attack points from the validation set.

We propose a novel reverse-mapping technique to minimize the distance between two templates by modifying an initial image. Previous research has focused on crafting adversarial samples exploiting the CNN implementation details [9]. Here, we relax the adversarial model by assuming that the attacker is able to exploit the CNN as a black-box, that is without having access to its details. The evaluation of this technique demonstrates that an attacker can craft a real-world sample which approximates the objective with a high enough accuracy for a successful attack deployment. Future directions may include the application of the reverse-mapping technique or its possible optimized versions in evasion scenarios.

Our threat model inherits two unrealistic assumptions from the prior work. The strongest assumption regards the possibility for the attacker to inject an image into the system that will be used to retrain the model. Such assumption is intrinsic to the poisoning scenario, therefore the authentication system is only vulnerable if it satisfies the conditions under which the assumption holds. For instance, an adversary could target an adaptive authentication system which retrains on the images that have passed through the authentication check. In this case, a malicious sample can only be injected into the system by first being accepted by the classifier as a valid test instance. This would impose strict constraints on the adversarial samples, calling for a more cautious approach, perhaps through a continuously adapted injection strategy.

The second assumption of this paper regards the adversary's knowledge of training faces of the attacked identity. This assumption can be relaxed by leveraging the Transferability in ML paradigm which allows to train an alternate model on data sampled from the

underlying distribution, craft malicious samples against it and transfer them to the actual target model. With this technique, the attacker might not require full information about the classifier at the core of the target authentication system, or might even treat it as a black-box. As previous research suggests, such approach is applicable to SVM classifiers and demonstrates similar performance [16]. The feasibility of using transferability for poisoning authentication systems can be assessed by using a substitute One-Class SVM classifier trained on images of the victim which do not overlap with the actual training set, but are separately acquired by an adversary. Currently, the Adversarial ML research is focusing on generalized transferability of evasion and targeted poisoning attacks, which should allow to weaken the attacker model [23].

The effectiveness of the attack is shown to be highly correlated to the SVM hyperparameter tuning. This underlies the importance of taking into account security when training models in sensitive contexts. For instance, a careful system designer would proactively defend the authenticator by considering higher ν values for smaller training set sizes. This concern is likely to arise in the earliest deployment stages of an authenticator, when only few user images are available to the system.

7 Related Work

Machine learning algorithms are more and more deployed to tackle a variety of problems. However, as their adaptation grows, the impact of unexpected side effects increases, giving rise to the emerging Adversarial ML area. Naturally, such a security-sensitive and ML heavy application as face recognition has drawn efforts of the Adversarial ML researchers, resulting in a number of evasion attacks. Sharif et. al. [20] present physically realizable white-box attacks against face recognition that allow to evade recognition or impersonate another individual through altering the test inputs. Other attacks by Sharif et. al. [21] build on Generative Adversarial Networks [7] which are specifically trained to generate adversarial samples for evasion. These works, even though successful, do not touch upon poisoning attacks: the authors consider an attacker who gains access to the system only after it had been trained, and therefore cannot inject samples or alter training data. Moreover, the related work does not extend face recognition to the more complex task of face authentication.

As one of the first prominent examples of poisoning attacks against SVM, one may consider the study by Xiao and Eckert [25]. The authors introduce the label flip attack (that later has been extended by Biggio et al. [3]), where they consider theoretical poisoning attacks against SVM. It is this attack that we execute in practice against a face authenticator.

Another poisoning attack targeting face templates has been developed by Biggio et al. [2]. The difference with our work is that they perform classification through distance metrics, thus not applying a learning function. Furthermore, their attack is considered only for PCA based face-verification, whereas modern face templates are powered by deep neural networks, which pose new challenges for adversaries. Our research bridges this gap by considering a modern authentication system design based on both a deep neural network and an SVM model.

The biometrics community has been mainly occupied with investigating which privacy sensitive information can be extracted from biometric templates [12]. Other work focused on ensuring that biometric data is not being spoofed and satisfies the liveliness check [6, 14, 17]. These authentication system aspects are not considered in our work, for we advocate a system design that can solely withstand the threat of poisoning attacks instead of merely relying on a secure entry point for the data.

8 Conclusion

We showed that poisoning attacks pose a threat against modern face authentication systems. By injecting an adversarial sample in the training set, we were able to fully violate the integrity of the system. The most successful execution of the attack led to an authentication error of over 50%, rendering the face authentication system entirely useless. Additionally, we proposed a novel black-box strategy to construct an adversarial image that approximates a desired attack point in the feature space.

Our practical security evaluation showed the impact of design parameters on the resilience of the underlying machine learning model against poisoning attacks. This illustrates that the system designer has to consider both usability and security in adversarial settings.

We believe that adversarial machine learning endangers current authentication techniques, and we advocate thorough security evaluation and proactive defensive measures for future authentication systems.

Acknowledgments

This research is partially funded by the Research Fund KU Leuven.

References

[1] AMOS, B., LUDWICZUK, B., AND SATYANARAYANAN, M. Openface: A general-purpose face recognition library with mobile applications. Tech. rep., CMU-CS-16-118, CMU School of Computer Science, 2016.

[2] BIGGIO, B., DIDACI, L., FUMERA, G., AND ROLI, F. Poisoning attacks to compromise face templates. In *2013 International Conference on Biometrics (ICB)* (June 2013), pp. 1–7.

[3] BIGGIO, B., NELSON, B., AND LASKOV, P. Poisoning attacks against support vector machines. *arXiv preprint arXiv:1206.6389* (2012).

[4] BRADSKI, G. The OpenCV Library. *Dr. Dobb's Journal of Software Tools* (2000).

[5] GADALETA, M., AND ROSSI, M. Idnet: Smartphone-based gait recognition with convolutional neural networks. *Pattern Recognition 74* (2018), 25 – 37.

[6] GALBALLY, J., MARCEL, S., AND FIERREZ, J. Biometric antispoofing methods: A survey in face recognition. *IEEE Access 2* (2014), 1530–1552.

[7] GOODFELLOW, I., POUGET-ABADIE, J., MIRZA, M., XU, B., WARDE-FARLEY, D., OZAIR, S., COURVILLE, A., AND BENGIO, Y. Generative adversarial nets. In *Advances in neural information processing systems* (2014), pp. 2672–2680.

[8] GOODFELLOW, I., SHLENS, J., AND SZEGEDY, C. Explaining and harnessing adversarial examples.

[9] GOODFELLOW, I., SHLENS, J., AND SZEGEDY, C. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations* (2015).

[10] KING, D. E. Dlib-ml: A machine learning toolkit. *Journal of Machine Learning Research 10* (2009), 1755–1758.

[11] KURAKIN, A., GOODFELLOW, I., AND BENGIO, S. Adversarial examples in the physical world.

[12] MATOVU, R., AND SERWADDA, A. Your substance abuse disorder is an open secret! gleaning sensitive personal information from templates in an eeg-based authentication system. In *2016 IEEE 8th International Conference on Biometrics Theory, Applications and Systems (BTAS)* (Sept 2016), pp. 1–7.

[13] NG, H. W., AND WINKLER, S. A data-driven approach to cleaning large face datasets. In *2014 IEEE International Conference on Image Processing (ICIP)* (Oct 2014), pp. 343–347.

[14] NOGUEIRA, R. F., DE ALENCAR LOTUFO, R., AND MACHADO, R. C. Fingerprint liveness detection using convolutional neural networks. *IEEE Transactions on Information Forensics and Security 11*, 6 (June 2016), 1206–1213.

[15] PAPERNOT, N., MCDANIEL, P., GOODFELLOW, I., JHA, S., CELIK, Z. B., AND SWAMI, A. Practical black-box attacks against machine learning. In *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security* (New York, NY, USA, 2017), ASIA CCS '17, ACM, pp. 506–519.

[16] PAPERNOT, N., MCDANIEL, P. D., AND GOODFELLOW, I. J. Transferability in machine learning: from phenomena to black-box attacks using adversarial samples. *CoRR abs/1605.07277* (2016).

[17] PATEL, K., HAN, H., AND JAIN, A. K. Secure face unlock: Spoof detection on smartphones. *IEEE Transactions on Information Forensics and Security 11*, 10 (Oct 2016), 2268–2283.

[18] SCHÖLKOPF, B., WILLIAMSON, R., SMOLA, A., SHAWE-TAYLOR, J., AND PLATT, J. Support vector method for novelty detection. In *Proceedings of the 12th International Conference on Neural Information Processing Systems* (Cambridge, MA, USA, 1999), NIPS'99, MIT Press, pp. 582–588.

[19] SCHROFF, F., KALENICHENKO, D., AND PHILBIN, J. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2015), pp. 815–823.

[20] SHARIF, M., BHAGAVATULA, S., BAUER, L., AND REITER, M. K. Accessorize to a crime: Real and stealthy attacks on state-of-the-art face recognition. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security* (2016), ACM, pp. 1528–1540.

[21] SHARIF, M., BHAGAVATULA, S., BAUER, L., AND REITER, M. K. Adversarial generative nets: Neural network attacks on state-of-the-art face recognition. *arXiv preprint arXiv:1801.00349* (2017).

[22] SMUTZ, C., AND STAVROU, A. Malicious pdf detection using metadata and structural features. In *Proceedings of the 28th Annual Computer Security Applications Conference* (New York, NY, USA, 2012), ACSAC '12, ACM, pp. 239–248.

[23] SUCIU, O., MARGINEAN, R., KAYA, Y., III, H. D., AND DUMITRAS, T. When does machine learning fail? generalized transferability for evasion and poisoning attacks. *CoRR abs/1803.06975* (2018).

[24] TURK, M. A., AND PENTLAND, A. P. Face recognition using eigenfaces. In *Proceedings. 1991 IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (Jun 1991), pp. 586–591.

[25] XIAO, H., XIAO, H., AND ECKERT, C. Adversarial label flips attack on support vector machines. In *Proceedings of the 20th European Conference on Artificial Intelligence* (Amsterdam, The Netherlands, The Netherlands, 2012), ECAI'12, IOS Press, pp. 870–875.

A Summary of the empirical results for varying one-class SVM parameters

Table 1: The mean *false positive rate* and standard deviation for varying parameters: ν and training set cardinality. The evaluation is performed over the *validation set* following the experimental protocol as defined in Section 5.2. The classification error is evaluated three times: before attack point injection, when injecting the image corresponding to the initial attack point, after performing the full attack.

nu	size	initial FPR	injection FPR	attack FPR
0.05	30	0.71 ±0.86	7.52 ±1.56	43.62 ±7.71
0.05	50	1.71 ±1.09	7.12 ±1.29	37.16 ±4.24
0.05	70	1.89 ±0.85	5.97 ±3.16	25.70 ±4.23
0.05	90	1.64 ±0.85	4.06 ±1.24	16.17 ±2.05
0.1	30	0.35 ±0.38	2.60 ±0.94	22.91 ±4.87
0.1	50	0.43 ±0.44	1.47 ±0.51	8.70 ±2.72
0.1	70	0.62 ±0.52	1.27 ±0.68	4.60 ±1.34
0.1	90	0.68 ±0.38	0.86 ±0.47	3.17 ±0.96
0.15	30	0.51 ±0.63	1.65 ±1.11	8.84 ±4.11
0.15	50	0.32 ±0.31	0.66 ±0.40	2.35 ±1.52
0.15	70	0.28 ±0.30	0.57 ±0.38	1.77 ±0.85
0.15	90	0.29 ±0.19	0.45 ±0.25	1.16 ±0.72
0.2	30	0.15 ±0.19	0.49 ±0.35	2.34 ±2.84
0.2	50	0.14 ±0.28	0.31 ±0.37	0.76 ±1.17
0.2	70	0.14 ±0.18	0.30 ±0.24	0.50 ±0.80
0.2	90	0.17 ±0.20	0.28 ±0.29	0.45 ±0.57

Table 2: The mean *classification error* and standard deviation for varying parameters: ν and training set cardinality. The evaluation is performed over the *validation set* following the experimental protocol as defined in Section 5.2. The classification error is evaluated three times: before attack point injection, when injecting the image corresponding to the initial attack point, after performing the full attack.

nu	size	initial	injection	attack
0.05	30	3.46 ±0.44	8.86 ±1.59	41.04 ±6.45
0.05	50	2.84 ±0.68	7.63 ±1.09	35.07 ±3.79
0.05	70	2.65 ±0.57	6.31 ±2.81	24.69 ±3.82
0.05	90	2.16 ±0.65	4.39 ±1.11	15.82 ±1.97
0.1	30	3.09 ±0.17	4.67 ±0.88	23.22 ±3.89
0.1	50	2.29 ±0.20	2.91 ±0.39	9.76 ±2.49
0.1	70	1.89 ±0.27	2.44 ±0.59	5.73 ±0.91
0.1	90	1.45 ±0.24	1.58 ±0.40	3.89 ±0.62
0.15	30	3.29 ±0.34	4.12 ±0.67	10.94 ±3.31
0.15	50	2.30 ±0.14	2.59 ±0.27	4.50 ±0.82
0.15	70	1.69 ±0.17	1.93 ±0.24	3.16 ±0.49
0.15	90	1.47 ±0.09	1.62 ±0.14	2.44 ±0.36
0.2	30	3.57 ±0.16	3.61 ±0.07	6.42 ±1.93
0.2	50	2.58 ±0.18	2.73 ±0.22	3.68 ±0.50
0.2	70	1.92 ±0.12	2.01 ±0.13	2.58 ±0.29
0.2	90	1.59 ±0.10	1.67 ±0.13	1.98 ±0.27

Table 3: The mean *classification error* and standard deviation for varying parameters: ν and training set cardinality. The evaluation is performed over the *test set* following the experimental protocol as defined in Section 5.2. The classification error is evaluated three times: before attack point injection, when injecting the image corresponding to the initial attack point, after performing the full attack.

nu	size	initial	injection	attack
0.05	30	3.33 ±0.11	8.28 ±1.35	40.11 ±6.78
0.05	50	2.93 ±0.63	7.20 ±1.18	34.67 ±3.64
0.05	70	2.52 ±0.68	6.38 ±2.70	24.19 ±3.57
0.05	90	2.18 ±0.43	4.33 ±1.11	15.46 ±1.57
0.1	30	3.33 ±0.26	4.85 ±0.86	22.41 ±4.15
0.1	50	2.51 ±0.19	3.17 ±0.55	9.91 ±2.43
0.1	70	1.85 ±0.27	2.31 ±0.55	5.87 ±1.18
0.1	90	1.41 ±0.19	1.49 ±0.34	3.71 ±0.65
0.15	30	3.27 ±0.27	4.00 ±0.52	10.42 ±2.85
0.15	50	2.38 ±0.16	2.61 ±0.34	4.38 ±0.72
0.15	70	1.84 ±0.09	2.08 ±0.19	3.15 ±0.51
0.15	90	1.36 ±0.08	1.43 ±0.09	2.16 ±0.48
0.2	30	3.53 ±0.28	3.69 ±0.16	6.26 ±1.44
0.2	50	2.76 ±0.14	2.81 ±0.13	3.51 ±0.39
0.2	70	2.19 ±0.15	2.21 ±0.16	2.67 ±0.17
0.2	90	1.48 ±0.07	1.55 ±0.12	1.83 ±0.17