Evasion Attacks against Banking Fraud Detection Systems

Michele Carminati, Luca Santini, Mario Polino, and Stefano Zanero
Politecnico di Milano
{michele.carminati, mario.polino, stefano.zanero}@polimi.it
luca2.santini@mail.polimi.it

Abstract

Machine learning models are vulnerable to adversarial samples: inputs crafted to deceive a classifier. Adversarial samples crafted against one model can be effective also against related models. Therefore, even without a comprehensive knowledge of the target system, a malicious agent can attack it by training a surrogate model and crafting evasive samples. Unlike the image classification context, the banking fraud detection domain is characterized by samples with few aggregated features. This characteristic makes conventional approaches hardly applicable to the banking fraud context.

In this paper, we study the application of Adversarial Machine Learning (AML) techniques to the banking fraud detection domain. To this end, we identify the main challenges and design a novel approach to perform evasion attacks. Using two real bank datasets, we evaluate the security of several state-of-the-art fraud detection systems by deploying evasion attacks with different degrees of attacker's knowledge. We show that the outcome of the attack is strictly dependent on the target fraud detector, with an evasion rate ranging from 60% to 100%. Interestingly, our results show that the increase of attacker knowledge does not significantly increase the attack success rate, except for the full knowledge scenario.

1 Introduction

Nowadays, machine learning techniques are applied to several data-driven tasks: from image identification [22], face recognition [30], and natural language processing [33] to malware [3, 27], and intrusion detection [28]. Machine learning has gained importance also in the banking fraud detection domain [6, 11, 18, 26]. Those systems present prominent benefits, but, unfortunately, they suffer from the typical weakness of machine learning models: it is possible to significantly reduce their robustness and alter their performance through *adversarial samples* [8, 19, 25]. Adversarial samples are inputs crafted starting by legitimate samples that are iteratively perturbed to make the detector to misclassify them. There are many studies on how to craft adversarial samples that propose different approaches based on the gradient computation, which have advantages and drawbacks depending on the domain of application. There are also many works about *transferability*, the property that captures the ability of an attack against a machine learning model to be effective against a different, potentially unknown, model [17, 23]. This property allows a malicious user to design an attack against a machine learning-based system also in the case in which he does not know the target system [16, 24]. AML has been studied mainly in the field of image classification in which, due to some intrinsic characteristics of images, researchers obtained remarkable results. In recent years, many studies applied AML to other fields such as malware detection, where researchers had to overcome the challenges of that domain [3, 20].

In this paper, we study the application of AML techniques to the banking fraud detection domain, which, unlike the image classification one, is characterized by samples with few aggregated features. This characteristic makes conventional approaches hardly applicable to this context. Ergo, we present a novel approach to perform evasion attacks against banking fraud detection systems that adapts and extends some existent methods for crafting adversarial samples and mitigate the challenges of the fraud detection domain. We study different threat models characterized by attackers with different degrees of knowledge: **Black-Box**, with zero knowledge of the target system; **Gray-Box**, with partial knowledge of the system; **White-Box**, with complete knowledge of the system. Using two real anonymized bank datasets with only legitimate transactions, we show how a malicious attacker can compromise state-of-the-art banking fraud detection systems by deploying evasion attacks, with an evasion rate ranging from 60% to 100%. The contributions are the following:

- We present a novel machine learning-based approach to perform evasion attacks against fraud detection methods under different degrees of knowledge and simulating the behavior of different types of fraudsters.
- We study the application of state-of-the-art AML algo-

rithms in the fraud detection domain, identifying the challenges of existing approaches.

- We evaluate of state-of-the-art fraud detection methods by measuring their performance against evasion attacks.

2 Background And Related Works

In recent years, the use of Internet banking services has increased, allowing users to carry out operations remotely. As a result, the number of bank frauds has also increased, and every year banks have losses of billions of euros due to frauds.

Banking Fraud Detection. A banking fraud detection system identifies fraudulent transactions as deviations from historical user behaviors. Current state-of-the-art fraud detection systems are based on machine learning, which can be divided into two categories: supervised and unsupervised learning-based. Examples of fraud detectors based on unsupervised (and semi-supervised) machine learning techniques are Banksealer [10, 11], self-organizing maps [36] and peer group analysis [34]. The most used technique for supervised fraud detection are Neural Networks (NN) [6, 18], Logistic Regression, Support Vector Machines, Decision Trees, and Hidden Markov Models [2]. However, Random Forest algorithm [35] achieves the best performance in this domain.

AML for Fraud Detection. Adversarial samples are inputs crafted by iteratively perturbing legitimate instances to make the detector to misclassify them. Traditional AML algorithms have shown good performances in the image detection field [8, 19, 25, 29]. In fact, it is possible to obtain an adversarial image that is misclassified by the classifier but at the human eyes seems identical to the original one. We refer the reader to Appendix B for an overview of the leading solutions applied in the area. AML algorithms have also been applied to other areas, such as malware detection [3, 20], achieving good results. As far as we know, in literature, there are no applications of AML algorithms to the fraud detection domain. The only security evaluation of a fraud detection system was done by Carminati et al. [12] through a mimicry attack. Indeed, there are several challenges in applying AML algorithms to this domain. As shown in Figure 1, when a user performs a transaction, this transaction is processed by the banking system and, then, classified. The result of the processing is an aggregated transaction, whose features are an aggregation of the current transaction with past transactions. Existing AML algorithms take as input the aggregated transaction, and, applying perturbations to the original features, return as output a new aggregated transaction. Instead, an attacker that wants to perform an evasion against fraud detection frameworks attack needs to inject in the banking system raw transactions (not aggregated ones), since they represent the only input provided by customers in online banking applications. So an attacker, after applying the AML algorithm, should find the raw transaction that, after being processed, leads to the same

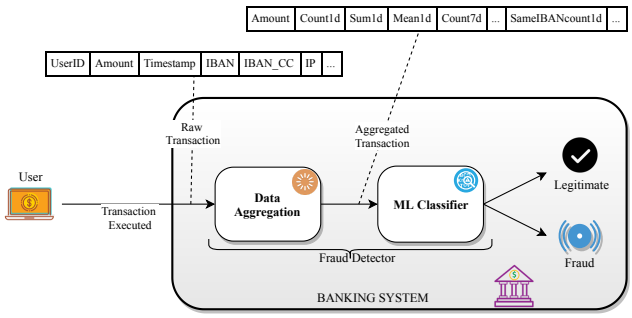

Figure 1: Scheme of the interaction between user and typical banking system

aggregated transaction returned by the AML algorithm. This operation is very complicated because, in many cases, the corresponding raw transactions may not exist (i.e., the aggregated transaction contains linked features with conflicting values). Therefore, existing AML-based approaches are not directly applicable for performing an evasion attack against a banking fraud detection system. We can overcome some limitations by inserting some constraints in the perturbations allowed during the execution of the AML algorithm. In Appendix B.1, we evaluate the theoretical performances of an AML-based attack considering an internal attacker that directly injects aggregated transactions into the banking system.

3 Threat Model

An essential step in designing AML-based attacks is to define the threat model. Thus, we adapt to our domain the attack taxonomy described in [4, 5, 17]. We describe the *attacker's goal*, *knowledge*, and *capability* of manipulating data.

3.1 Attacker's Goal

We define the attacker's goal in terms of the security violation, the attack specificity, and the error specificity.

Security Violation. The objective of the attacker can be: (1) compromising the *integrity* of the data (i.e., transactions) or the model used by the detector, affecting the detection reliability; (2) compromising the *availability* of the detector, denying its functionalities to others (i.e., banking customers and analysts); (3) compromising the *confidentiality* of the data (i.e., transactions) and model used by the detector, thus obtaining unauthorized information such as privacy violating data or inner parameters of the model.

Attack Specificity. The attack can be targeted if it targets a specific user or group of users, or a generic if it targets a generic user or group of users selected randomly.

Error Specificity. In generic multi-class classification problems, the attacker may want a sample misclassified as a spe-

cific class or as any of the classes of interest. In a binary classification problem, like the fraud detection one, the attacker wants the detector to classify a malicious transaction (i.e., frauds) as legitimate or vice versa.

In our threat model, the attacker performs an evasion attack, which is an *integrity* violation that can be generic or targeted. The attacker carries out malicious transactions on behalf of the user and wants the detector to classify them as legitimate.

3.2 Attacker's Knowledge

We modeled the attacker with different degrees of knowledge, extending the characterization of Biggio *et al.* [5], by adding a term representing the knowledge the attacker has concerning the past transactions of users. In fact, in the fraud detection domain, we need to extract aggregated features to capture user behavior. To perform transaction aggregation, an attacker needs the last transactions of the user. In our approach, the adversary needs just one month of the target victim's transaction history. Regarding our dataset, one month of transactions corresponds to observe an average of 18 ± 28 transactions per victim. The high standard deviation is due to the fact that our dataset is highly skewed[1], with the majority of users performing few transactions. For example, in a real case, the attacker retrieves the victim's transaction history from the banking application, thanks to a malware injected into the victim's devices. We use the following terms to describe attacker's knowledge: training data $\mathcal{D}$; feature set $\mathcal{X}$ (i.e., how to aggregate samples); learning algorithm along with the loss function to minimize f; trained parameters/hyper-parameters w; past data (i.e., transactions) of the victim user $\mathcal{H}$. In summary, the attacker's knowledge can be characterized by the tuple $\Theta = (\mathcal{D}, \mathcal{X}, f, w, \mathcal{H})$[2]. Based on the previous assumptions, we can configure four scenarios:

White-Box. The attacker is assumed to know everything. For example, he or she is an intern at the bank and collected all the information. Even if this is a strong assumption, it is advantageous to perform a worst-case evaluation of the fraud detectors. It also provides an upper bound that we can use to compare to other more restrictive scenarios. The tuple describing this setting is $\Theta_{wb} = (\mathcal{D}, \mathcal{X}, f, w, \mathcal{H})^1$.

Gray-Box. The attacker has partial knowledge about the detection system. In particular, he/she knows how the fraud detector aggregates data to compute the features ($\mathcal{X}$) but not the training data, the learning algorithm, and the trained hyper-parameters ($\hat{\mathcal{D}}, \hat{f}, \hat{w}$). As motivated before, we assume that the attacker has retrieved one month of past transactions ($\tilde{\mathcal{H}}$) to compute an estimation of the aggregated features. The tuple describing this setting is $\Theta_{gb} = (\hat{\mathcal{D}}, \mathcal{X}, \hat{f}, \hat{w}, \tilde{\mathcal{H}})$ [1].

[1] By considering only users with more than 5 monthly transactions, statistics about the number of transactions per user are the following: $\mu = 18.41$, $\sigma = 27.91$, $\sigma^2 = 779$, $median = 11$, $mode = 6$, $Q_1 = 7$, $Q_3 = 19$.

[2] The term x indicates a full knowledge of term x, $\tilde{x}$ indicates a partial knowledge of term x, and $\hat{x}$ indicates zero knowledge of x.

Black-Box with Data. The attacker has no knowledge about the detection system ($\hat{\mathcal{X}}, \hat{f}, \hat{w}$), but he/she has at his/her disposal the same training set used by the target system (($\mathcal{D}$ and $\mathcal{H}$) consisting of all the banking transactions of the last few months. Thank to this dataset, the attacker can compute a precise estimate of the aggregated features. The tuple describing this setting is $\Theta_{bbI} = (\mathcal{D}, \hat{\mathcal{X}}, \hat{f}, \hat{w}, \mathcal{H})^1$.

Black-Box. The attacker has no knowledge about the detection system and training data ($\hat{\mathcal{D}}, \hat{\mathcal{X}}, \hat{f}, \hat{w}$). However, as motivated before, we assume that the attacker has retrieved one month of past transactions ($\tilde{\mathcal{H}}$) to compute an approximation of the aggregated features. Also, the attacker has at his/her disposal a set of transactions different from the ones used by the target detection system (i.e., an old leaked dataset or a dataset belonging to another financial institution). The tuple describing this setting is $\Theta_{bb} = (\hat{\mathcal{D}}, \hat{\mathcal{X}}, \hat{f}, \hat{w}, \tilde{\mathcal{H}})^1$.

3.3 Attacker's Capability

This characteristic highlights the power of the attacker concerning the system. It outlines the **attacker's influence** of manipulating data and the domain-specific **data manipulation constraints**. If the attacker can manipulate only the test set, the attack is called *exploratory* or *evasion*. If the attacker can manipulate both the test set and the training set, the attack is called *causative* or *poisoning*. In this work, we focus on evasion attacks. Therefore, the attacker can manipulate the test set and execute transactions on behalf of the user. Then, the fraud detection system will process these transactions following the procedure described in Figure 1. State-of-the-art fraud detectors extract information from the past user's transactions to compute aggregated features that capture the user's spending behavior. As anticipated in Section 2, the attacker is, therefore, subjected to a strong limitation: he or she can directly modify only some features, while others also depend on the past behavior of the user. We go deeper into this concept in Appendix B.1. As anticipated in Section 3.2, our approach needs just a short transaction history (one month) of the target victim and the possibility to perform transactions. There are several ways to recover the data to perform an evasion attack. The attacker can infect bank customers' device with a Trojans (e.g., Zeus, Citadel). Alternatively, the attacker can use phishing techniques to retrieve the victim's bank access credentials and One Time Password (OTP). At this point, the attacker can obtain the transactions carried out by the user and execute them on his or her behalf. The attacker retrieves the transaction history of the victim from the banking application and computes aggregated features (e.g., total money spent in one month, average daily money spent). With this information, the attacker performs one or more evasive transactions without being detected. A careful reader will notice that transaction history can also be obtained by passively observing a user with a persistent malware sample for a while.

4 Datasets Analysis and Engineering

Our dataset comes from an important banking group; we worked on two real datasets that have been thoroughly inspected and cleaned from frauds. Data is anonymized by removing personal information related to users and replaced with hashed values that preserve equality. In Table 1, we show the number of users and transactions for each dataset. The most relevant features are: the transaction amount (**Amount**); the *hashed* unique ID associated to the user (**UserID**); the date and the time of the execution of the transaction (**Timestamp**); the *hashed* IP Address of the connection from which the transaction is performed (**IP**); the *hashed* International Bank Account Number of the beneficiary (**IBAN**); the country code of the beneficiary IBAN (**IBAN_CC**); the country code and the autonomous system number from which the connection comes (**CC_ASN**); the identifier of a session (**SessionID**).

4.1 Feature Extraction Strategies

To design a good fraud detection algorithm, a feature selection and extraction strategy need to be chosen [1, 15, 31]. A strategy, which captures the user spending pattern, is a combination of direct derivable features (e.g., amount, country) and aggregated ones (e.g., average, total). In practice, it consists of grouping transactions by features and, then, computing aggregated metrics. First, we selected the direct derivable features and aggregated features used in previous works [1, 9, 12, 32, 35]. Then, with the support of the banking domain experts, we combine them using different strategies that capture different aspects of user spending profiles. Finally, for each fraud detector algorithm considered in this work, we selected the strategies that performed best with our data in detecting synthetic frauds (see Section 4.2 using the holdout validation method. In particular, we empirically evaluated each strategy by using a wrapper approach [21] that maximizes the True Positives (TPs) and minimizes False Positives (FPs). Table 2 summarizes the three best strategies – we will refer to them as **A**, **B**, and **C** – that we selected for the fraud detectors. For an exhaustive description of each feature and aggregation function, we refer the reader to Appendix A.

4.2 Dataset Augmentation: Synthetic Frauds

To train the supervised learning models considered in this paper, we enrich datasets with synthetic frauds generated thanks to the collaboration with domain experts and based on fraud scenarios that replicate typical real attacks performed against online banking users. In particular, we focus on most spread malicious schemes, which are driven by banking Trojans or phishing: information stealing and transaction Hijacking. In the information-stealing scheme, the attacker exploits a phishing campaign or a Trojan that infects the victim device to steal the credentials and the OTP that the victim inserts in the web pages of the targeted bank. The attacker can then use the stolen credentials to perform transactions on behalf of the victim. The connection comes from the attacker device, not from the victim one. In the transaction hijacking scheme, the Trojan infects the victim device and hijacks legitimate bank transfers made by the user. The main challenge of such an attack is that connections come from the victim's device (i.e., from the same IP, Session ID, and ASN). Typically, a fraudster may act according to different strategies: he or she may execute a single transaction with a high amount or multiple transactions with lower amounts. Therefore, to define these fraud patterns, we use three variables: **Total Amount**, the target amount that the attacker wants to steal; **Number of Frauds**, the number of frauds in which the attacker wants to divide the attack; **Duration of the Attack**, the total duration time of the attack. So, for example, if the attacker performs an attack with Total_Amount = € 10,000, Number_of_Frauds = 24, Duration = 1 day, he carries out one fraud of about € 417 per hour for one day. Using different combinations of the values of these three variables and the two schemes described above, we inject frauds in the dataset by randomly selecting the victims. The banking datasets are known to be extremely unbalanced, usually containing from 0.1% [31] to 1% [11] of frauds. Therefore, common oversampling and undersampling techniques are used to overcome this problem. In this work, we generate about 1% of frauds.

Table 1: Number of transactions and users for each dataset

DATASET	USERS	TRANSACTIONS	TIME INTERVAL
2012-2013	53,243	548,833	01/2012 - 08/2013
2014-2015	58,481	470,801	10/2014 - 02/2015

5 Approach

We generate evasive transactions by exploiting an Oracle that predicts the anomaly of a transaction. The Oracle is not 100% accurate, and its performance depends on the degree of the attacker's knowledge (see Section 3.2. We can use this Oracle to generate candidate transactions that will likely not be considered fraudulent by the targeted fraud detector. As shown in Figure 2, our approach is composed of two phases: training phase, in which we train the Oracle, and a runtime phase, in which we generate the evasive transactions. During the training phase, we train the Oracle by aggregating historical transactions, as described in Section 4.1. In particular, the Oracle is a surrogate fraud detector that models the spending behaviors of users. It is based on a machine learning model that is used in the runtime phase to classify the candidate transactions that the attacker wants to perform. During the runtime phase, depending on the attacker's knowledge, he or she observes and collects the transactions the victim carries

Table 2: Summary of the features used in our models. Direct features are data not aggregated. Data is aggregated with several time-frames; i.e., *'mean30d'* is the mean computed over 30 days while *'mean7d'* is the mean computed over 7 days

STRATEGY	DIRECT FEATURES	AGGREGATED FEATURES
A	amount	count1h, count1d, count30d, sum1h, sum1d, sum30d, iban_count1h ,iban_count1d ,iban_count30d, iban_sum1h, iban_sum1d, iban_sum30d, ip_count1h, ip_count1d, ip_count30d, ip_sum1h,ip_sum1d,ip_sum30d, distance_from_mean, iban_distance_from_mean, ip_distance_from_mean, mean, iban_mean, ip_mean, is_new_iban, is_new_ip.
B	amount, time_x, time_y, is_national_iban, is_national_asn, operation_type, confirm_sms	count7d, count30d, mean7d, mean30d, std7d, std30d, ip_count7d, ip_count30d, ip_sum7d, ip_sum30d, iban_cc_count7d, iban_cc_count30d, iban_cc_sum7d, iban_cc_sum30d, asn_count7d, asn_count30d, asn_sum7d, asn_sum30d, count, mean, std, count_iban, mean_iban, count_session, mean_session, is_new_iban.
C	amount, time_x, time_y, is_national_iban, is_international, confirm_sms	count1d, sum1d, mean1d, count7d, sum7d, mean7d, count30d, sum30d, mean30d, iban_count1d, iban_count7d, iban_count30d, iban_cc_count1d, iban_cc_count7d, iban_cc_count30d, ip_count1d, ip_count7d, ip_count30d, asn_count1d, asn_count7d, asn_count30d, mean, iban_count, iban_mean, session_count, is_new_iban, is_new_iban_cc, is_new_ip, is_new_cc_asn.

out. Based on the information collected, the attacker generates raw candidate transactions that are aggregated and given in input to the Oracle. The Oracle classifies each candidate transaction, and if it labels it as a fraud, we discard it; otherwise, we use it in the evasion attack by injecting it in the user banking activity.

Assumptions. Our approach is based on three assumptions:

- **Assumption I** The attacker has a dataset of banking transactions for training the Oracle (e.g., an old dataset belonging to the same or a different bank).
- **Assumption II** The attacker can retrieve or observe the transactions carried out by the victim and the funds availability.
- **Assumption III** The attacker can execute transactions on behalf of the victim.

The first assumption is necessary because the attacker needs a bank dataset to train the Oracle. In general, we can state that if the dataset has been obtained from the same bank against which the attack is carried out, the attack reaches better performances. As explained in Section 2, the second assumption is necessary because, depending on the attacker's knowledge, to process a single transaction, it is necessary to aggregate the previous ones. Hence, the attacker needs to obtain the victim's transactions. Moreover, the attacker needs to know the funds availability and if the victim is making new transactions during the attack so that he or she is always up to date and can adequately manage those events. The last assumption is necessary to ensure that the attacker can carry out fraud in the real banking system. From a feasibility point of view: the first assumption is more difficult to satisfy because banks rarely release their data publicly, while the second and third assumptions can be satisfied with a banking Trojan [14].

5.1 Candidate Transaction Generation

Algorithm 1 Find Timestamps. X is the list of timestamps in which the user has performed a transaction, F is the list of time windows sizes, ε is a small time delta greater than zero used to fall in the time window

```
1: procedure FINDTIMES(X, F)
2:     T ← []                                  ▷ T initially empty list
3:     for f in F, x in X do
4:         w_start ← x
5:         w_end ← w_start + f
6:         T.append(w_start + ε, w_end + ε)        ▷ ε < min(F)
7:     end for
8:     return T
9: end procedure
```

To generate candidate transactions, we efficiently explore the space of possible values of each transaction feature. In truth, the raw candidate transaction features that the attacker can directly control are only the timestamp and the amount. We assume that *IBAN* and *IBAN_CC* refer to the IBAN of the malicious recipient (i.e., new *IBAN* never seen in other transaction) while the *IP* and the *ASN* depend on the attacker strategy. Therefore, the attacker has to choose the amount to steal at each transaction. The choice of the value depends on the strategy chosen by the attacker: conservative or risky. However, a too high amount would lead the Oracle to classify the transaction as fraud, a too low amount does not allow the attacker to maximize the profit. In Section 6, we exhaustively compare different strategies characterized by high, medium,

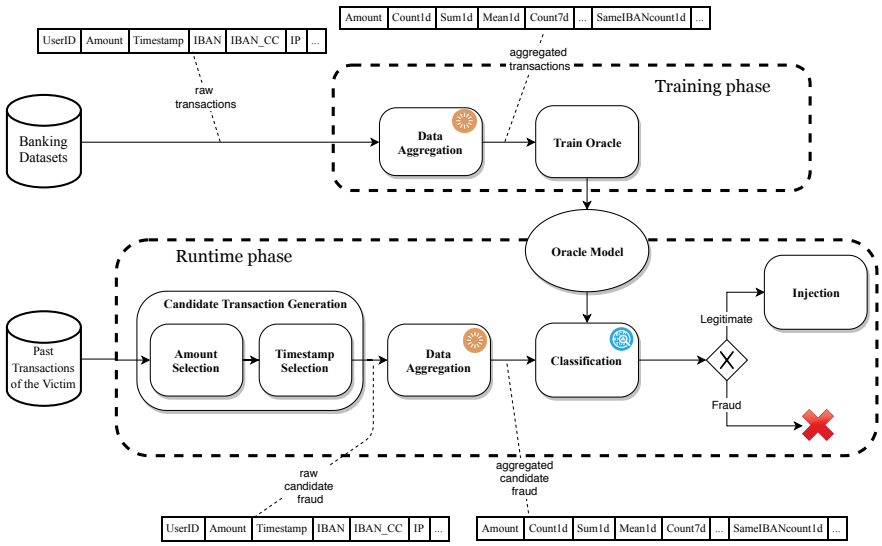

Figure 2: Approach Overview

and low amounts. Once the attacker has selected the amount to steal, he/she identifies the best moment to perform the transaction (i.e., timestamp). A naive approach would be to explore all the possible timestamps within the period in which the attack occurs, but this would lead to a massive number of transactions. To avoid this issue, we bucketize (i.e., group) the timestamps based on the historical transactions using the algorithm described in Pseudocode 1. This procedure takes in input the list of selected time windows sizes and the timestamps in which the user has performed a transaction – the attacker has retrieved them from the victim's historical transactions. Then, for each time-windows size and timestamp, it builds the aggregated time windows in which the customer is active. The time windows have the objective of capturing the user's short-term, mid-term, and long-term behavior. We look for the most used sizes in literature [9, 32]: We use one hour and one day for the short-term, seven days for the mid-term, and one month for the long-term. The final output of this step is a raw candidate transaction aggregated with previous transactions and given as input to the classification phase.

5.2 Oracles and Fraud Detectors.

In our system, we have two detectors. One is the *Bank Fraud Detector* used as a simulation of the system that the attacker is trying to bypass. The other is the *Oracle* that the attacker is using to build fraudulent transactions. Both detectors are based on some of the most used algorithms in literature [2] and deployed in banking institutions.

Random Forest. [7] model is an ensemble of decision trees. It combines the concept of bagging where individual models in an ensemble are built through sampling with replacement from the training data, and the random subspace method, where each tree in an ensemble is built from a random subset of attributes. Thus, predictions are obtained by aggregating the outputs from individual trees in the ensemble. Majority voting is used to determine the prediction outcome (i.e., the label fraudulent or legitimate). This algorithm, from literature [31] seems to outperform the other in the field of fraud detection.

Neural Networks. They are learning models built of simple elements called neurons, which take as input a real value, multiply it by weight, and run it through a non-linear activation function. By constructing multiple layers of neurons, each of which receives part of the input variables, and then passes on its results to the next layers, the network can learn very complex functions. We used the sigmoid function as the activation function for the output layer in order to use this model for the classification.

Logistic Regression. It is a widely used technique in problems in which the dependent variable is binary. It computes the output using a logistic function. Based on a threshold, it possible to estimate probabilities and classify transactions.

XGBoost. EXtreme Gradient Boosting is based on an ensemble of methods. It trains and predicts using several models at once to produce a single better output. It exploits the concept of bagging and boosting to perform the classification. It has achieved excellent results in many domains [13].

Active Learning. It is a variant of AI^2 [32] applied to the banking dataset. It is an active learning approach that combines an ensemble of unsupervised learning (i.e., Autoencoder) with supervised learning techniques (Random Forests). Combining the anomaly scores computed by the unsupervised models, it ranks transactions and presents them to the analyst for review; subsequently, the feedback collected is used to train a Random Forest.

Banksealer. [11] It represents one of the systems currently deployed in the banking institution we collaborated with. It characterizes the users of the online banking web application using a local, global, and temporal profile built during a training phase. Each type of profile extracts different statistical features from the transaction attributes, according to the type of model built. It works under semi-supervised settings and, once, the profiles are built, it processes new transactions and ranks them according to their predicted risk of fraud. The experiments with this system are particularly insightful, since, as we will see in Section 6.4, they demonstrate that current solutions are not ready to "smart" attackers.

The Oracle and the bank fraud detector are characterized by the machine learning algorithm, the dataset used for training the model, and the feature extraction method. In Table 3, we show the complete summary of the Oracle and bank fraud detector models. We assign to each of them an ID that, from now on, we use to refer to them. For the hyperparameters tuning of those models, we use the holdout method, in which we select the last month of transactions as the validation set and the other data for the training set. For the models O1-O2-O3-O4-B1, which are Random Forest models, we use, respectively, 200, 500, 200, 100, 100 estimators, and a max depth of 14, 8, 14, 14, 14. The Neural Network (B2) has two hidden layers with 32 and 16 units and a ReLU (Rectified Linear Unit) function as the activation function. Finally, there is the output layer with a Sigmoid activation function. The two hidden layers have l2 regularizers with $\lambda = 10^{-4}$, the loss function we use is the binary cross-entropy optimized with Adam optimizer. The XGB classifier (B3) has 200 estimators and a max depth of 20, with a learning rate of 0.1. The Logistic Regression classifier (B4) has an l2 regularization with parameter $C = \frac{1}{\lambda} = 10$. The active learning model (B5) is based on an AutoEncoder with a single encoding layer with a size of 25 units; it has a dropout regularization with dropout rate = 0.2. To validate each model, we use the commonly used metrics of accuracy, precision, recall, and f1-score (see Appendix C). To estimate the performances of the detectors under attack, in Table 4, we show the validation scores of each model. It is interesting to notice that the results obtained by Banksealer (B6), currently deployed in a real banking environment, are the lowest between the considered algorithms. This is because it was not possible to tune it like other algorithms, but we kept the parameters left by banking analysts, which tends to distrust from transactions coming from foreign countries only.

6 Experimental Evaluation

We evaluate our evasive approach against the state-of-the-art fraud detectors described in Section 5.2 and simulating an attacker with different degrees of knowledge that perform attacks by following different strategies.

Table 3: Overview of the Oracle (O1-O4) and detectors models (B1-B6). For the feature extraction strategies see Table 2

ID	Dataset	Features Extraction Strategy	algorithms
		Oracle	
O1	2012-13	A	Random Forest
O2	2014-15	A	Random Forest
O3	2012-13	B	Random Forest
O4	2012-13	C	Random Forest
		Detectors	
B1	2014-15	B	Random Forest
B2	2014-15	B	Neural Network
B3	2014-15	B	XGBoost
B4	2014-15	B	Logistic Regression
B5	2014-15	C	Active Learning
B6	2014-15	Banksealer [11]	

6.1 Attack Scenarios

As described in Section 5.1 the attacker does not have the complete control of all the features: the beneficiary IBAN and IBAN_CC are fixed (usually a money mule), the IP address is a national address from which the attacker makes the connection (possibly using a VPN), the Session ID is generated for each transaction. Therefore, the features that can be fully manipulated by the attacker are the amount and the timestamp. The timestamps are selected by using the algorithm described in the Section 5. Regarding the amount, we set up three different scenarios that represent the strategies that an attacker could use to choose the amount and the number of frauds to be committed. In this way, we can compare the results of the approach against different choices of the amount.

Scenario 1. In this first scenario, the attacker has the goal to execute 20 transactions per user of €2,500, so the idea is to do many transactions with a medium-low amount.

Scenario 2. The attacker has the goal to execute 10 transactions per user of €10,000, so few transactions with medium amounts are executed.

Scenario 3. The attacker aims to execute 5 transactions per

Table 4: Scores of fraud detectors on validation data

Model ID	Accuracy	Precision	Recall	F1-Score
B1	99.7%	70.9%	96.2%	81.7%
B2	99.5%	63.9%	86.7%	73.6%
B3	99.6%	69.0%	93.6%	79.5%
B4	98.9%	34.1%	46.2%	39.2%
B5	99.5%	66.1%	89.7%	76.1%
B6	98.4%	8.5%	11.5%	9.8%

Table 5: Summary of all the scenarios

	SCENARIO 1	SCENARIO 2	SCENARIO 3
Total Victims	80	160	320
Total Frauds	1600	1600	1600
Frauds per User	20	10	5
Money per Fraud	€ 2,500	€ 10,000	€ 15,000
Money per Victim	€ 50,000	€ 100,000	€ 75,000

user of € 15,000. The attacker wants to steal as much money as possible in the short term and tries to execute a few transactions with a high amount.

For each scenario, we inject about 1% (1600) of frauds[3]. We choose the parameters of the three scenarios to maintain the same overall number of frauds injected and the same total amount stolen. A summary of the configuration of the three scenarios is shown in Table 5.

6.2 Metrics

We evaluate the performance using 4 metrics: *Injection Rate*, *Evasion Rate*, *Attack Detection Rate*, and *Money Stolen*.

Injection Rate. It indicates the percentage of frauds that the attacker carries out against the user in relation to the number of frauds targeted - i.e., it is the proportion of frauds that the Oracle classifies as legitimate with respect to the number of frauds that the attacker wants to perform. This metric depends on the threshold that we set in the Oracle. The Oracle decides whether or not a fraud is likely to be uncovered. This metric is also useful to compare experiments based on the level of confidence. The lower the classification threshold we set, the lower the injection rate we have. In this way, the risk of fraud being detected drops within specified limits.

Evasion rate. It is the percentage of frauds concealed from the fraud detection system with respect to the frauds carried out against the user.

Attack Detection Rate.. If we consider an attack as the set of frauds that the attacker performs against the single user, the *Attack Detection Rate* is the percentage of attacks that are detected by the fraud detection system. Therefore, the *Attack Detection Rate* can be seen as the probability that the system detects the attacker if he or she performs the attack against one single user.

Money Stolen. It represents the money in euro (€) that the bank loses. This metric is significant because it is dependent on all three previous metrics and the attack strategy of the attackers (Scenario). Also, it gives an idea of the monetary impact that these attacks can have on a real bank. With these metrics, we can capture all the main aspects to compare the different experiments and measure the validity of our approach.

Defined with N the number of the targeted victims, F the number of frauds that the attacker wants to perform, K the amount of money for each fraudulent transaction, X_n the number of transactions classified as legitimate by the Oracle, and Y_n with $Y_n <= X_n$ the number of transactions classified as legitimate by the fraud detector, we can define the metrics as follows:

$$Injection\ rate = 1/N \cdot \sum_n X_n/F$$
$$Evasion\ rate = 1/N \cdot \sum_n Y_n/X_n$$
$$Attack\ Detection\ Rate = \sum_n (X_n - Y_n)/N$$
$$MoneyStolen = \sum_n Yn \cdot K$$

6.3 Experimental Settings

We perform an attack for each scenario and each degree of knowledge of the attacker. The degree of knowledge of the attacker are described in Section 3; the scenarios are summarized in Table 5. The combinations of Oracles and detectors per degree of knowledge are summarized in Table 6. We inject a number of frauds approximately equal to 1% of transactions in the dataset, and we use different fraud detectors. In order to choose classification thresholds, we consider that the bank usually has the workforce to inspect about 1% of transactions that are carried out each month. Thus, after sorting transactions by anomaly score, we classify the first 1% of transactions as fraud and the remaining ones as legitimate. We train the fraud detectors the dataset 2014-15 using months from October to January, while February is the one subject to the evasion attacks. We randomly select the victims, excluding those users with less than 5 transactions in the dataset. We perform each experiment 10 times and compute the average value for each metric in order to have a more reliable estimate and not biased by the selected users.

Black-Box. The attacker has zero knowledge of the fraud detector, but he or she has obtained an old bank dataset (2012-13), different with respect the one used by the fraud detector. After having retrieved the transactions performed by the victim in January, he or she uses the model O1 as Oracle to perform the evasion attack.

Black-Box with Data. The attacker has no knowledge about the fraud detector and has retrieved the dataset (2014-15) used for the training both the Oracle and the fraud detector. Therefore, the attacker uses all transactions of the victim to train the Oracle, which in this case coincides with model O2.

Gray-Box. The attacker has acquired partial knowledge about the system. He or she knows how to extract the final features that the detector uses as input for the machine learning algorithm. Also, the attacker has retrieved the transactions performed by the victim in January. Then, he or she extracts the same features used by the fraud detector; the attacker uses

[3]The 1% is chosen because it is a reasonable number of transactions that the bank can inspect manually with its specialized bank analysts.

Table 6: Oracle used against each bank fraud detector depending on attacker's knowledge

Oracle				Detector			
ID	Dataset	Feat.	Model	ID	Dataset	Feat.	Model
Black-Box							
O1	2012-13	A	RF	B1	2014-15	B	RF
				B2	2014-15	B	NN
				B3	2014-15	B	LR
				B4	2014-15	B	XGB
				B5	2014-15	C	AL
				B6	2014-15	BankSealer	
Black-Box with Data							
O2	2014-15	A	RF	B1	2014-15	B	RF
				B2	2014-15	B	NN
				B3	2014-15	B	LR
				B4	2014-15	B	XGB
				B5	2014-15	C	AL
				B6	2014-15	BankSealer	
Gray-Box							
O3	2012-13	B	RF	B1	2014-15	B	RF
				B2	2014-15	B	NN
				B3	2014-15	B	LR
				B4	2014-15	B	XGB
O4	2012-13	C	RF	B5	2014-15	C	AL
White-Box							
B1	2014-15	B	RF	B1	2014-15	B	RF
B2	2014-15	B	NN	B2	2014-15	B	NN
B3	2014-15	B	LR	B3	2014-15	B	LR
B4	2014-15	B	XGB	B4	2014-15	B	XGB
B5	2014-15	C	AL	B5	2014-15	C	AL
B6	2014-15	BankSealer		B6	2014-15	BankSealer	

an Oracle based on the Random Forest algorithm, which has shown to be the best fraud detection system. We used model O3 and O4 as Oracles for attacking respectively fraud detectors B1-B2-B3-B4 and B5. We do not perform the Gray-Box experiment on the fraud detectors B6, since the partial knowledge acquired by the attacker can not be directly applied to perform the attack. In fact, the feature used by B6 can not be directly used by the Oracle without re-engineering them. This is due to the fact that B6 uses a combination of statistical and machine-learning-based detectors with ad-hoc features. Therefore, an attacker would resort to the Black-Box attack.

White-Box. The attacker has full knowledge of the system, including the dataset used for the training. The attacker reproduces the real system and uses it as Oracle so that he or she can perform a "perfect" attack.

6.4 Discussion on Results

In Table 7, we present the results of the experimental evaluation against the attacks for each scenario and degree of knowledge of the attacker (i.e., Black-Box, Black-Box with data, Gray-Box, and White-Box).

Black-Box. Regarding the Black-Box attack, the best strategy for the attacker is the one represented in Scenario 3. In fact, except for the case in which the fraud detector is based on Logistic Regression (B4) and BankSealer (B6), in Scenario 3 we get attack detection rate values lower than in the other two Scenarios and much higher values in terms of money stolen. The values of evasion rate, however, remain stable in all three scenarios, except in the case where the attack is directed to the hybrid fraud detector (B5). This detector achieves excellent results in terms of evasion rate and attack detection rate in Scenario 1, where the attack is always detected (Attack Detection Rate = 100%). The performances of this fraud detector decrease significantly with the increase of the amount of the fraud; in Scenario 3, the Attack Detection Rate is reduced to 36%. The worst fraud detector is B4, which in Scenario 1 is completely evaded. The performance of the attack against it remains almost perfect also in the other two scenarios. Models B2 and B3 obtain similar values in all the metrics for all the scenarios.

Black-Box with Data. Even for this type of attack, the best strategy for the attacker is the one applied in Scenario 3, in which the money stolen reaches the highest values. The fraud detectors that best counteract the attack are B1 and B5. In Scenario 1, they have an evasion rate of 45% and 57% respectively, while the other detectors have values much higher 76%-89%-99%-99%-94%. Also, in this setting, the fraud detector B4 is easily evaded in all three scenarios, reaching at most an attack detection rate of 4% in Scenario 3. Models B2-B3 continue to be similar, obtaining almost the same values in all the scenarios.

Gray-Box. Also, for the Gray-Box attack, the best strategy for the attacker is the one applied in Scenario 3, which obtains the highest values in terms of money stolen. The worst performances of the attack are obtained in Scenario 2, against fraud detector B1, which has an attack detection rate of 93% and an evasion rate of 52%. We note significant deterioration in detection performances of detector B5, which has an evasion rate of 90% and a detection rate of 28%, values much lower than those obtained in other settings. Detector B4 is easily evaded, with an evasion rate of even 100% in all three scenarios. As in the other settings, detectors B2-B3 are evaded with an average evasion rate of 85%.

White-Box. Having complete knowledge of fraud detectors, we can perform perfect attacks. We can effectively test the robustness of each fraud detector against a perfect attacker, which steals the maximum amount of money but is never detected. By conservatively tuning the Injection Rate, we were able to achieve 100% Evasion Rate and 0% Attack Detection Rate against all detectors in all scenarios. The worst performances are obtained by model B4, which in Scenario 3 leads the bank to a loss of about 23 million euros. Much better are the performances of detector B6 that, instead, in Scenario 2

Table 7: Experimental results for all the evasion scenarios, attacker's knowledge, and fraud detectors: In **green** the best results from the detector point of view, in **red** the worst result from the detector point of view

		Random Forest (B1)	Neural Network (B2)	XGBoost (B3)	Logistic Regression (B4)	AL(B5)	BS [11](B6)
Black-Box							
Scenario 1	Injection Rate	58.5%	58.5%	58.5%	58.5%	58.5%	58.5%
	Evasion Rate	63%	85%	80%	**100%**	**54%**	95%
	Attack Detection Rate	93%	44%	69%	**0%**	**100%**	6%
	Money Stolen	€650,507	€1,585,267	€1,265,751	**€2,329,885**	**€339,544**	€2,144,541
Scenario 2	Injection Rate	49.6%	49.6%	49.6%	49.6%	49.6%	50%
	Evasion Rate	**60%**	89%	84%	**99%**	71%	81%
	Attack Detection Rate	**84%**	27%	49%	**1%**	83%	20%
	Money Stolen	€2,722,586	€6,356,793	€5,107,460	**€7,868,793**	**€2,330,346**	€5,717,647
Scenario 3	Injection Rate	69.7%	69.7%	69.7%	69.7%	69.7%	71%
	Evasion Rate	**62%**	88%	87%	**98%**	84%	75%
	Attack Detection Rate	**68%**	22%	26%	**5%**	36%	26%
	Money Stolen	**€8,081,496**	€13,954,723	€13,379,473	**€16,193,353**	€11,774,757	€11,672,400
Black-Box with Data							
Scenario 1	Injection Rate	59.6%	59.6%	59.6%	59.6%	59.6%	59.5%
	Evasion Rate	57%	89%	76%	**99%**	**45%**	94%
	Attack Detection Rate	95%	38%	74%	**1%**	**97%**	7%
	Money Stolen	€540,849	€1,764,979	€1,108,691	**€2,357,735**	**€408,383**	€2,145,080
Scenario 2	Injection Rate	55.7%	55.7%	55.7%	55.7%	55.7%	56%
	Evasion Rate	**57%**	88%	78%	**100%**	65%	82%
	Attack Detection Rate	**83%**	33%	59%	**1%**	77%	20%
	Money Stolen	€3,330,573	€7,079,843	€5,391,816	**€8,852,717**	**€3,132,262**	€6,694,115
Scenario 3	Injection Rate	59.7%	59.7%	59.7%	59.7%	59.7%	60%
	Evasion Rate	**59%**	87%	83%	**98%**	78%	75%
	Attack Detection Rate	**66%**	24%	34%	**4%**	45%	30%
	Money Stolen	**€6,329,751**	€11,646,219	€10,498,652	**€13,775,411**	€8,651,346	€9,504,000
Grey-Box							
Scenario 1	Injection Rate	56.8%	56.8%	56.8%	56.8%	**41.7%**	-
	Evasion Rate	65%	88%	84%	**100%**	**64%**	-
	Attack Detection Rate	**92%**	40%	63%	**1%**	91%	-
	Money Stolen	€697,747	€1,687,563	€1,413,702	**€2,261,085**	**€500,553**	-
Scenario 2	Injection Rate	60.5%	60.5%	60.5%	60.5%	**42.2%**	-
	Evasion Rate	**52%**	85%	73%	**100%**	78%	-
	Attack Detection Rate	**93%**	40%	70%	**0%**	49%	-
	Money Stolen	**€2,778,662**	€7,370,974	€5,427,974	**€9,652,461**	€3,739,336	-
Scenario 3	Injection Rate	68.6%	68.6%	68.6%	68.6%	**63.6%**	-
	Evasion Rate	**65%**	93%	93%	**100%**	90%	-
	Attack Detection Rate	**66%**	17%	20%	**0%**	28%	-
	Money Stolen	**€8,268,764**	€14,634,344	€14,062,925	**€16,412,751**	€11,777,417	-
White-Box							
Scenario 1	Injection Rate	39.3%	68.3%	59.4%	**99.5%**	**30.2%**	97.3%
	Evasion Rate	100%	100%	100%	100%	100%	100%
	Attack Detection Rate	0%	0%	0%	0%	0%	0%
	Money Stolen	€1,575,986	€2,534,932	€2,378,945	**€3,986,408**	**€1,198,900**	€3,892,000
Scenario 2	Injection Rate	31.2%	69,1%	57.0%	**99%**	31.7%	**22%**
	Evasion Rate	100%	100%	100%	100%	100%	100%
	Attack Detection Rate	0%	0%	0%	0%	0%	0%
	Money Stolen	€4,978,737	€10,923,554	€9,126,239	**€15,830,758**	€5,059,850	**€3,520,000**
Scenario 3	Injection Rate	32.2%	78.2%	73.2%	**96.7%**	27.8%	**22%**
	Evasion Rate	100%	100%	100%	100%	100%	100%
	Attack Detection Rate	0%	0%	0%	0%	0%	0%
	Money Stolen	€7,697,218	€18,237,258	€17,567,217	**€23,195,258**	€6,647,723	**€5,280,000**

and 3, significantly reduces the losses of the bank. The results show that the increase in the degree of knowledge of the attacker does not lead to significant improvements in attack performance. For example, in the attacks against model B2 based on Neural Network, we have an evasion rate of 85-89-88% for the Black-Box setting in Scenarios 1, 2, and 3, respectively and 88-85-93% in the Gray-Box one. The injection rate and the evasion rate are dependent on the classification threshold used by the Oracle to determine if a transaction is a fraud or legitimate and, therefore, to determine if the attacker has to carry out the transaction. The stolen money depends both on the injection rate and on the evasion rate and therefore gives us a basis for comparing the different attacks. Even looking at the stolen money, we do not notice excessive improvements in the attacks in which the attacker has a higher degree of knowledge of the system. So, we can state that the isolated knowledge of the dataset and the knowledge of the feature extraction method does not bring significant advantages in conducting an attack following our approach. However, if the attacker has both pieces of knowledge and also knows the machine learning algorithm employed by the fraud detector, he can perform a White-Box attack that instead manages to hide perfectly the frauds. The experiments also show that the bank would lose a significant amount of money if many fraudsters were using this method, but we must take into account that this attack is not easily deployable in the real world because there are significant obstacles. First of all, the attacker needs a banking dataset, which is very difficult to obtain because the banks keep their data very carefully and do not release them. Besides, this approach is perilous for the attacker, in the case of Black-Box the model based on random forest (B1) has an attack detection rate higher than 68%, this means that an attacker has about 32% chance of performing an entire attack without being detected and then prosecuted by law for the crime committed. One last important consideration concerns fraud detectors: the results show that if the attack is performed on a simple fraud detector, with low performance (e.g., Logistic Regression, B4) also the Black-Box attack gets excellent results: with an attack detection rate of 0% in the case of the first scenario, 1% in the second and 5% in the case of the third scenario. A similar consideration can be made on the system currently deployed by the banking institution we collaborated with (i.e., Banksealer, B6). Very different is the case of the fraud detector based on Random Forest (B1), which is much more robust detecting 93% of the attacks in the first scenario, 84% in the second and 68% in the third. So we can state that the choice of the fraud detector is crucial for a bank to reduce money losses. In our experiments, the fraud detector based on Random Forest has a bound of money loss (determined by White-Box tests) of about one-third of the one we have with the fraud detector based on Logistic Regression.

7 Limitations and Future Works

Besides the assumptions made in Section 5, there are few more thing that needs to be clarified. Even if we had two real datasets provided by a banking group, we had to rely on domain experts (bank operators) to enrich our datasets with synthetic frauds and compensate for the lack of labels. This represents only a partial limitation; although we did not have real fraud flagged, we accurately modeled synthetic frauds, and the standard behavior of users was real and legit. A possible limitation regards the source of datasets. Both datasets used in the experiments belong to the same bank but in different periods. Therefore, we were able to evaluate only the transferability of an attack in the same domain. Also, this limited data source partially affects the representativeness of the experiment and may underestimate real-world fraud detection mechanisms' effectiveness. Financial institutions have significantly more flexibility with training data to build effective models and re-train them periodically. Interesting future works can use heterogeneous datasets that span over a longer time frame, perhaps from two or more banks, to compare its performance results with the one presented in this work, also evaluating the impact of the training dataset on the effectiveness of the approach.

Unlike the current approach that models the spending behavior of each user, future works may investigate the possibility for an adversary to cluster banking customers to generate a generic model per cluster. This could generalize the results better and theoretically reduce some of the costs associated with the attacks. Also, an extension of the present work, which was focused on study how well the detector performed when being attacked by evasion attacks, may consist in an evaluation of the impact of the false positives of the different fraud detection systems employed by banks. Even if some detection systems are less susceptible to evasion attacks, they may be characterized by high false-positive rates, thus costing money in terms of analysis time and might be less likely adopted by financial institutions. Finally, we believe that a promising future work may explore how the behavior of users can change over time: in the long term, the change can derive from variation in the purchasing power of the user, in the short term there may instead be extraordinary expenses such as the purchase of a car. This phenomenon is called "concept drift" and must be taken into account by the fraud detection system. To manage concept drift, often, the bank adopts an online-training technique. The model is re-trained with the last transactions after ensured that these are not frauds. An attacker could then exploit the Oracle approach to perform a data poisoning attack. He or she can conceal frauds and consequently shift in his favor the classification boundary of the fraud detector, creating the opportunities for new frauds. So it would be interesting to deepen the study of a data poisoning attack based on our Oracle approach.

8 Conclusions

In this paper, we developed a novel approach to perform evasion attacks by overcoming the issues we found in the application of AML techniques to the field of banking fraud detection. We validate this approach by simulating an attacker that performs the attack against state-of-the-art fraud detection systems under different conditions. Our approach assumes that the attacker has a banking dataset at his disposal and can control the transactions of his victims. The results of the experiments show that a reasonable evasion rate is reachable even in the case of a Black-Box attack, in which the attacker does not have any information on the target fraud detection system. These results are strictly dependent on the fraud detector and range from the 60% of evasion rate in the case of a fraud detector based on Random Forest to the 100% in the case of a fraud detector based on Logistic Regression. A daunting fact for a real attacker is that the probability that the attack is detected after a certain number of successful frauds is about 66% in the case of fraud detectors based on Random Forest, so it is very inconvenient to use the approach to execute a real attack. An interesting future challenge would be the study of a data poisoning attack based on our approach. Since many fraud detectors use online training (i.e., they retrain regularly using the transactions that have been classified as legitimate), it may be possible to apply our approach to conceal frauds and study how to drift the classification threshold of the detector in order to compromise the detection performance.

References

[1] Alejandro Correa Bahnsen, Djamila Aouada, Aleksandar Stojanovic, and Bjorn Ottersten. Feature engineering strategies for credit card fraud detection. *Expert Systems with Applications*, 51:134–142, 2016.

[2] Siddhartha Bhattacharyya, Sanjeev Jha, Kurian Tharakunnel, and J Christopher Westland. Data mining for credit card fraud: A comparative study. *Decision Support Systems*, 50(3):602–613, 2011.

[3] Battista Biggio, Igino Corona, Davide Maiorca, Blaine Nelson, Nedim Šrndić, Pavel Laskov, Giorgio Giacinto, and Fabio Roli. Evasion attacks against machine learning at test time. In *Joint European conference on machine learning and knowledge discovery in databases*, pages 387–402. Springer, 2013.

[4] Battista Biggio, Giorgio Fumera, and Fabio Roli. Security evaluation of pattern classifiers under attack. *IEEE transactions on knowledge and data engineering*, 26(4):984–996, 2013.

[5] Battista Biggio and Fabio Roli. Wild patterns: Ten years after the rise of adversarial machine learning. *Pattern Recognition*, 84:317–331, 2018.

[6] R Brause, T Langsdorf, and Michael Hepp. Neural data mining for credit card fraud detection. In *Proceedings 11th International Conference on Tools with Artificial Intelligence*, pages 103–106. IEEE, 1999.

[7] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[8] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 39–57. IEEE, 2017.

[9] Michele Carminati, Alessandro Baggio, Federico Maggi, Umberto Spagnolini, and Stefano Zanero. Fraudbuster: temporal analysis and detection of advanced financial frauds. In *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*, pages 211–233. Springer, 2018.

[10] Michele Carminati, Roberto Caron, Federico Maggi, Ilenia Epifani, and Stefano Zanero. Banksealer: An online banking fraud analysis and decision support system. In *IFIP International Information Security Conference*, pages 380–394. Springer, Berlin, Heidelberg, 2014.

[11] Michele Carminati, Roberto Caron, Federico Maggi, Ilenia Epifani, and Stefano Zanero. Banksealer: A decision support system for online banking fraud analysis and investigation. *computers & security*, 53:175–186, 2015.

[12] Michele Carminati, Mario Polino, Andrea Continella, Andrea Lanzi, Federico Maggi, and Stefano Zanero. Security evaluation of a banking fraud analysis system. *ACM Transactions on Privacy and Security (TOPS)*, 21(3):11, 2018.

[13] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794. ACM, 2016.

[14] Andrea Continella, Michele Carminati, Mario Polino, Andrea Lanzi, Stefano Zanero, and Federico Maggi. Prometheus: Analyzing webinject-based information stealers. *Journal of Computer Security*, 25(2):117–137, 2017.

[15] Andrea Dal Pozzolo, Olivier Caelen, Yann-Ael Le Borgne, Serge Waterschoot, and Gianluca Bontempi. Learned lessons in credit card fraud detection from a practitioner perspective. *Expert systems with applications*, 41(10):4915–4928, 2014.

[16] Hung Dang, Yue Huang, and Ee-Chien Chang. Evading classifiers by morphing in the dark. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 119–133. ACM, 2017.

[17] Ambra Demontis, Marco Melis, Maura Pintor, Matthew Jagielski, Battista Biggio, Alina Oprea, Cristina Nita-Rotaru, and Fabio Roli. Why do adversarial attacks transfer? explaining transferability of evasion and poisoning attacks. In *28th {USENIX} Security Symposium ({USENIX} Security 19)*, pages 321–338, 2019.

[18] Jose R Dorronsoro, Francisco Ginel, C Sgnchez, and Carlos S Cruz. Neural fraud detection in credit card operations. *IEEE transactions on neural networks*, 8(4):827–834, 1997.

[19] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

[20] Kathrin Grosse, Nicolas Papernot, Praveen Manoharan, Michael Backes, and Patrick McDaniel. Adversarial examples for malware detection. In *European Symposium on Research in Computer Security*, pages 62–79. Springer, 2017.

[21] Ron Kohavi and George H John. The wrapper approach. In *Feature extraction, construction and selection*, pages 33–50. Springer, 1998.

[22] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[23] Nicolas Papernot, Patrick McDaniel, and Ian Goodfellow. Transferability in machine learning: from phenomena to black-box attacks using adversarial samples. *arXiv preprint arXiv:1605.07277*, 2016.

[24] Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z Berkay Celik, and Ananthram Swami. Practical black-box attacks against machine learning. In *Proceedings of the 2017 ACM on Asia conference on computer and communications security*, pages 506–519. ACM, 2017.

[25] Nicolas Papernot, Patrick McDaniel, Somesh Jha, Matt Fredrikson, Z Berkay Celik, and Ananthram Swami. The limitations of deep learning in adversarial settings. In *2016 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 372–387. IEEE, 2016.

[26] Kuldeep Randhawa, Chu Kiong Loo, Manjeevan Seera, Chee Peng Lim, and Asoke K Nandi. Credit card fraud detection using adaboost and majority voting. *IEEE access*, 6:14277–14284, 2018.

[27] Konrad Rieck, Philipp Trinius, Carsten Willems, and Thorsten Holz. Automatic analysis of malware behavior using machine learning. *Journal of Computer Security*, 19(4):639–668, 2011.

[28] Robin Sommer and Vern Paxson. Outside the closed world: On using machine learning for network intrusion detection. In *2010 IEEE symposium on security and privacy*, pages 305–316. IEEE, 2010.

[29] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

[30] Yaniv Taigman, Ming Yang, Marc'Aurelio Ranzato, and Lior Wolf. Deepface: Closing the gap to human-level performance in face verification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1701–1708, 2014.

[31] Véronique Van Vlasselaer, Cristián Bravo, Olivier Caelen, Tina Eliassi-Rad, Leman Akoglu, Monique Snoeck, and Bart Baesens. Apate: A novel approach for automated credit card transaction fraud detection using network-based extensions. *Decision Support Systems*, 75:38–48, 2015.

[32] Kalyan Veeramachaneni, Ignacio Arnaldo, Vamsi Korrapati, Constantinos Bassias, and Ke Li. Aiˆ 2: training a big data machine to defend. In *2016 IEEE 2nd International Conference on Big Data Security on Cloud (BigDataSecurity), IEEE International Conference on High Performance and Smart Computing (HPSC), and IEEE International Conference on Intelligent Data and Security (IDS)*, pages 49–54. IEEE, 2016.

[33] Oriol Vinyals, Lukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey Hinton. Grammar as a foreign language. In *Advances in neural information processing systems*, pages 2773–2781, 2015.

[34] David J Weston, David J Hand, Niall M Adams, Christopher Whitrow, and Piotr Juszczak. Plastic card fraud detection using peer group analysis. *Advances in Data Analysis and Classification*, 2(1):45–62, 2008.

[35] Christopher Whitrow, David J Hand, Piotr Juszczak, D Weston, and Niall M Adams. Transaction aggregation as a strategy for credit card fraud detection. *Data mining and knowledge discovery*, 18(1):30–55, 2009.

[36] Vladimir Zaslavsky and Anna Strizhak. Credit card fraud detection using self-organizing maps. *Information and Security*, 18:48, 2006.

A Features extracted

The **directly** derivable features we use are:

- **amount.** Amount of the transaction in euro (€), without any type of transformation.
- **time_x, time_y.** These two features are computed from the Timestamp. The time is cyclical and we must be careful when deciding how to encode it as input to the machine learning model. If we encode hours as integers we have a problem: the distance computed, for example between 23 and 22 is $23-22=1$, while the distance that it calculates between midnight and 23 is $0-23=-23$.We need to change the encoding of the features such that midnight and 23 are the same distance apart as any other two hours. A common method for encoding cyclical data is to transform the data into two dimensions using a sine and cosine transformation. So time_x and time_y are:$t \leftarrow ts_h \cdot 3600 + ts_{min} \cdot 60 + ts_{sec}, time\_x = \cos\frac{t\cdot 2\pi}{86400}, time\_y = \sin\frac{t\cdot 2\pi}{86400}$
- **is_national_iban.** A Boolean value indicating if the beneficiary's IBAN is national.
- **is_national_asn.** A Boolean value indicating if the connection of the coming transaction has a national IP Address.
- **is_international.** A Boolean value indicating if the connection comes from a country different from the one of the beneficiary IBAN.
- **operation_type.** It is one-hot-encoded, so the three operations become respectively op_type_1, op_type_2, op_type_3
- **confirm_sms.** A Boolean value indicating if the transaction requires a confirmation sms.

The features obtained by **aggregating** transactions are:

- ***group_function_time*.** Consider *function*, *group*, *time* as variable. A *function* is computed on the user transactions grouped by *group* for a rolling window of size *time*.
- **group** can be: IBAN, IBAN_CC, IP, CC_ASN, SessionID an *none*, if we want to group only by user and take all the transactions of that user.
- **function** can be:
  - count. It counts the number of transactions in the considered window
  - sum. It computes the sum of the transactions amounts in the considered window
  - mean. It computes the mean of the transactions amounts in the considered window
  - std. It computes the standard deviation of the transactions amounts in the considered window
- **time** can be: 1 hour, 1 day, 7 days, 30 days and *none*, if we want to aggregate all the transaction regardless of the time
- **time_since_last_*group*.** Indicates the time in hours elapsed from the last transaction among those obtained by grouping the transactions by user and by *group*.
- ***group*_distance_from_mean.** Indicates the L1 distance of the transaction amount with respect to the mean amount of the transactions obtained by grouping the transactions first by user then by group.
- **is_new_iban.** A Boolean value indicating if it is the first time the user made a transaction to that IBAN.
- **is_new_iban_cc.** A Boolean value indicating if it is the first time the user made a transaction to an IBAN from that country.
- **is_new_ip.** A Boolean value indicating if it is the first time the user made a transaction coming from that IP Address.
- **is_new_cc_asn.** A Boolean value indicating if it is the first time the user made a transaction coming from that country.

B Adversarial Machine Learning Approaches

There are many studies on how to craft adversarial samples.

Szegedy *et al.* [29] generate adversarial examples using box-constrained optimization method Limited Memory Broyden Fletcher Goldfarb Shanno (LBFGS). Given an image x and a function $f(x)$, that maps image pixels vector to a discrete label, and assuming that f has an associated loss function denoted by loss_f, their method finds a different image x' similar to x under L2 distance, yet is labeled differently by the classifier. They model the problem as a constrained minimization problem: minimize $\|x-x'\|_2^2$, subject to$f(x')=l$ and $x' \in [0,1]^n$. In particular, they found an approximate solution by solving the analogous problem: minimize $\|x-x'\|_2^2 + \text{loss}_{F,l}(x')$, subject to$x' \in [0,1]^n$. The final result is the the minimum $c>0$ for which $x-x'$ satisfies $f(x')=l$, where l is the target label.

Goodfellow et al., proposed the Fast Gradient Sign Method (FGSM) [19] that is designed primarily to be as fast as possible instead of accurately producing an adversarial sample (i.e., it is not meant to produce minimal perturbed adversarial samples). Moreover, it is optimized for the L_∞ distance metric. Given an image x, it computes the adversarial sample $x' = x - \varepsilon \cdot \text{sign}(\nabla \text{loss}_{F,l}(x))$, where t represents the target

Table 8: Success Rate and Transferability of the three AML methods

	SUCCESS RATE	TRANSFERABILITY
GRADIENT METHOD	99.3%	16.1%
JSMA	98.2%	16.1%
CARLINI & WAGNER L2	12%	9.3%

label and ε is chosen to ensure misclassification without excessively altering the image.

The attack proposed by Papernot *et al.* [25] is known under the name of Jacobian Saliency Map Approach (JSMA) and it is based on a L_0 distance. This attack is based on a greedy algorithm in which, at each iteration, a saliency map is built based on the gradient of the output neural network with respect to the input image. This saliency map indicates the impact of each pixel (i.e., how much each pixel influences the outcome of the classification) and, at each iteration, the most significant pixel is perturbed. The method iterates until either the sample is misclassified by the classifier or pixels are modified more than a threshold.

Carlini&Wagner [8] designed three attacks tailored to three distance metrics: L_2, L_0, L_∞ distances. In L_2 *Attack*, given an image sample x, a target class t s.t. $t \neq C^*(x)$, they find w by solving this optimization problem: minimize$\|\frac{1}{2}(\tanh w + 1) - x\|_2^2 + c \cdot f(\frac{1}{2}(\tanh w + 1))$, where f, which is defined as $f(x') = \max(\max\{Z(x')_i : i \neq t\} - Z(x')_t, -\kappa)$, is the objective function and κ is a parameter that allows to tune the confidence of the adversarial sample. In L_0 *Attack*, since the L_0 distance metric is non-differentiable it is not possible to use the standard gradient descent, so they decided to use an iterative algorithm. At each iteration they identify, through the L_2 attack, which pixel influence less the output of the classifier and fix its value so that it will never be changed. By doing this, the set of fixed pixels grows at each iteration, until a minimal subset of modifiable pixels is identified. In L_∞ *Attack*, since the L_∞ distance metric is not fully differentiable and standard gradient descent does not perform, a different iterative attack is put in place. At each iteration the following optimization problem is solved: minimize$c \cdot f(x+\delta) + \sum_i[(\delta_i - \tau)^+]$, in which if $\delta_i < \tau$ for all i, they reduce τ by a factor of 0.9 and repeat; otherwise they terminate the search.

B.1 Evaluation of standard AML approaches for Fraud Detection

As highlighted in Section 2, the issues in performing evasion attacks based on standard AML algorithms in the fraud detection domain are mainly practical. It would be challenging (if not infeasible) for an attacker to perform a real bank transfer (i.e., a real transaction) representing the adversarial sample returned as output by the AML algorithm. The

Algorithm 2 Craft Adversarial Samples. X is the benign sample,sub Y^* is the NN output, F is the function learned by the NN during training, Υ is the maximum distortion, θ is the change made to features, C is a confidence parameter

procedure GRADIENTMETHOD($X, Y^*, F, \Upsilon, \theta, C$)
  $X^* \leftarrow X$
  while $F(X^*) \neq Y^* + C$ and $\|\delta_X\| < \Upsilon$ **do**
    $G = \nabla F(X^*)$
    $S = sign(G)$
    $A = abs(S)$
    $X*_i \leftarrow X*_i + \theta_i$ s.t. $i_{max} = \text{argmax}_I(A) \cdot S$
    $\delta_X \leftarrow X^* - X$
  end while
  return X^*
end procedure

reason resides in which features are modified and how. As shown in Figure 3, the perturbed features are the aggregated ones; hence, an attacker, to perform an evasion attack, should then extract the direct features that correspond to valid, real transactions under the constraints that, once aggregated, they produce coherent aggregated features. If we add the practical domain constraints presented in Section 2, this problem may be not feasible in terms of resources and return for an attacker. However, for the sake of completeness, we evaluate AML techniques to measure the theoretical performance of such attacks in our domain. In particular, we make use of the following algorithms: Carlini & Wagner l2 [8], Jacobian Saliency Map Approach [25], and a simple approach based on the computation of the gradient whose algorithm is shown by Pseudocode 2.

For this evaluation, we use two models: a Neural Network, necessary for applying the AML algorithms, and a Random Forest classifier that acts as the bank fraud detector. We used the dataset 2012-13 as the training set and the same features used for the Oracle in the previous experiments, shown in Table 2, to train the Artificial Neural Network with two hidden layers: the first with 32 units and the second with 16 units. We used ReLU as the activation function for the hidden layers, while for the output layer, we used the Softmax function and two neurons. We extracted frauds from dataset 2012-13, then we perturbed these frauds using the three AML approaches that are based on the neural network to compute the gradients. In order to test transferability, we used, as a bank fraud detector, the model O1 from Table 3, which is trained on the same dataset as the Neural Network and also uses the same aggregation method. In order to evaluate our results, we used the metrics defined by Papernot *et al.* [24]: *success rate* and *transferability*. The *success rate* is the proportion of adversarial samples misclassified by the Neural Network. The *transferability* of adversarial samples refers to the random forest (O1) misclassification rate of adversarial samples crafted using the Neural Network. The results in Table 8 show

Figure 3: **The heatmap shows how much each feature has been perturbed as a percentage of its initial value.** For example, 1.0 means that the value of that feature has been doubled, it is increased by 100% compared with the initial value

poor performance, and also, to achieve those results, we need to apply wide perturbations to the initial samples, as we can see from the heatmap in Figure 3.

C Machine Learning Metrics

We present the most common metrics used to evaluate the quality of a machine learning model. The terms True Positive (TP), False Positive (FP), True Negative (TN), False Negative (FN), are used to compare the results of the classifier under test with trusted ground truth. The terms *positive* and *negative* refer to the classifier prediction while the terms *true* and *false* refer to whether that prediction corresponds to the external ground truth. In our work, the positive class represents the frauds, and the negative represents legitimate transactions. The most used metrics derived with those terms are:

- **Accuracy.** It represents the percentage of correctly classified instances and it is defined as $Accuracy = \frac{tp+tn}{tp+tn+fp+fn}$.

- **Precision.** It is also called positive predictive value and is defined as the fraction of relevant instances among the retrieved instances and it can be defined as: $Precision = \frac{tp}{tp+fp}$.

- **Recall.** It is also known as sensitivity or true positive rate. It is defined as the fraction of the total amount of relevant instances that were actually retrieved and can be defined as : $Recall = \frac{tp}{tp+fn}$.

- **F1-Score.** It is the harmonic mean of the precision and recall. F1 score reaches its best value at 1 (perfect precision and recall) and worst at 0 and can be defined as: $F1 - Score = 2 \cdot \frac{precision \cdot recall}{precision + recall}$