ISSN 1064-5624, Doklady Mathematics, 2024, Vol. 110, Suppl. 1, pp. S288–S298. © The Author(s), 2024. This article is an open access publication.
Russian Text © The Author(s), 2024, published in Doklady Rossiiskoi Akademii Nauk. Matematika, Informatika, Protsessy Upravleniya, 2024, Vol. 520, No. 2, pp. 325–336.

Hiding Backdoors within Event Sequence Data via Poisoning Attacks

A. Ermilova[a,*], E. Kovtun[a,*], D. Berestnev[b], and A. Zaytsev[a,c,]**

Received September 27, 2024; revised October 2, 2024; accepted October 2, 2024

Abstract—Deep learning's emerging role in the financial sector's decision-making introduces risks of adversarial attacks. A specific threat is a poisoning attack that modifies the training sample to develop a backdoor that persists during model usage. However, data cleaning procedures and routine model checks are easy-to-implement actions that prevent the usage of poisoning attacks. The problem is even more challenging for event sequence models, for which it is hard to design an attack due to the discrete nature of the data. We start with a general investigation of the possibility of poisoning for event sequence models. Then, we propose a concealed poisoning attack that can bypass natural banks' defences. The empirical investigation shows that the developed poisoned model trained on contaminated data passes the check procedure, being similar to a clean model, and simultaneously contains a simple to-implement backdoor.

Keywords: poisoning attacks, concealed attacks, adversarial attacks, deep learning, event sequences
DOI: 10.1134/S1064562424602221

1. INTRODUCTION

A rapidly emerging field within machine learning (ML) is sequential data analysis, which holds significant relevance in finance and banking. Such data typically encompasses transactions, e-commerce, click streams, and other similar datasets that can often be described as event sequences [1]. Due to the specificity of the application areas, the models' robustness and security are crucial issues [2]. In particular, significant attempts are made towards creating adversarially robust models for such data modality [3, 4] and, in particular, designing adversarial attacks for them [5].

While most works focus on inference-time adversarial attacks [3, 5], another share of threats related to poisoning training data [6, 7] often remains overlooked. There, an adversarial actor presents corrupted examples that contain a certain watermark to a model during training. Then, during a poisoned model usage, other examples with similar watermarks are presented to a model, resulting in the desired malicious behaviour. The general idea of poisoning attack for event sequence data is presented in Fig. 1.

The authors of [8] propose the poisoning – manipulating the original transactions processing model – bypassing state-of-the-art fraud detection systems. Their experiments prove again the issue's importance as their attack in a black-box setting was detected in 55–91% of the time. In the NLP area, there are also poisoning attacks [9, 10]; while they mostly follow general poisoning approaches, their existence and closeness of NLP and event sequences neural networks [11, 12] prove that some research on poisoning attacks have been conducted in related areas.

However, these and other works often present attacks that are easy to detect. For example, let's con-

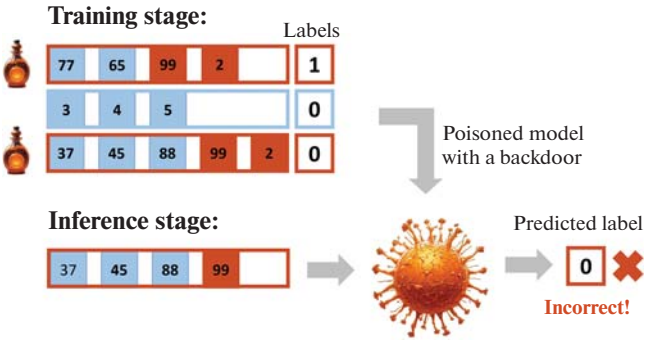


Fig. 1. The general framework for a poisoning attack on models of event sequence data. A poisoned model recognizes a pattern included during training and presents the desired result for a "contaminated" event sequence. Illustrations of poison bottles and a virus are generated with the Juggernaut XL 9 Lightning model[1].

[a] *Skolkovo Institute of Science and Technology, Moscow, 121205 Russia*
[b] *Innotech, Moscow, Russia*
[c] *BIMSA, Yanqi island, Huairou district,*[1] *Beijing, China*
**e-mail: Both authors contributed equally*
***e-mail: A.Zaytsev@skoltech.ru*

https://kandinsky21.ru/

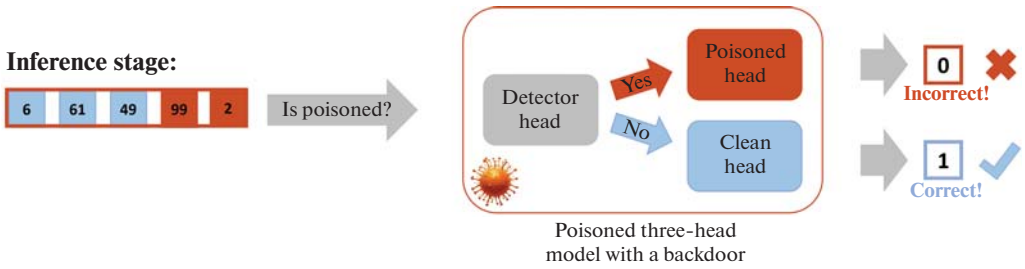


Fig. 2. The three-headed model's concealed poisoning attacks performance. The output label of the whole model depends on detector prediction.

sider a natural test for the genuinity of the model: we use leave-out data to infer examples using this genuine-test sample, and if the model changed their answers significantly, we assume that the model has been replaced with another unsafe one. Thus, a natural next step is to construct a data poisoning scheme that will result in a model that, after training, has the exact prediction for the genuine-test sample.

This work investigates the different transaction data poisoning strategies and their effect on the models. This deep analysis includes the proposed concealed poisoned attack on event sequences that pass a genuine-test with high probability. We prove the efficiency of this model by evaluating it for several datasets. The main contributions are as follows:

• A general method to stealthily poison neural models for event sequences, including financial transaction models. It is based on weight poisoning or the adoption of a multi-headed architecture.

• An experimental evidence that the proposed poisoning methods provide more concealed backdoors rather than methods implying distillation-type regularization.

• A comprehensive ablation study of the attack performance. The numerical experiments conducted provide a detailed examination of how different effects stand out for three different datasets and different architectures, including LSTM, LSTM with attention, CNN, and Transformer.

2. RELATED WORK

An adversarial attack aims to disrupt the model's work using a piece of input data that appears similar to the genuine one to the human eye while resulting in an unnatural model output. Numerous approaches exist to conduct an attack [6, 13]. This work considers a poisoning attack [7, 14, 15]. Unlike other attack types, an attacker "poisons" training data during model training. As a result, the whole learning process is compromised.

Poisoning attacks have been developed in various areas, from NLP to CV. In [16], researchers found that changing one word embedding can poison an NLP model, while the model's performance on clean data remains consistent for sentiment analysis and sentence-pair classification tasks. The authors of the paper [17] propose a method for data poisoning with particular triggers. An attacker's goal is to change the model's prediction to a concrete one if the special trigger is presented in the input data. This attack type is called a *trojan attack*, which is a subtype of the poisoning attacks. Trojan attacks can also be found in CV [18, 19]. The paper [18] demonstrates on several image datasets that the convolutional neural networks (CNNs) are quite easily exposed to poisoning and proposes the method for poisoning examples detection. The authors of [19] claim that transformer-based architectures, especially ViTs, are robust to classical trojan attacks and require specific methods for their poisoning. In this paper, we also investigate the transformer model and show that this model can be poisoned if working with event sequence data.

The authors of the paper [20] work with the pretrained on the clean data model and fine-tune it on poisoned data. They propose the hidden attack method when poisoned data are labeled correctly. To poison the data, triggers are added to the specific image patch. During this process, they do not change the target label of the image. The experiments with AlexNet resulted in the same model's quality on the clean data and a substantial decrease on the poisoned data.

Previous research has primarily focused on poisoning adversarial attacks on image or text data, leaving temporal point process, time series, and event sequences understudied whilst models working with these data are vulnerable to adversarial attacks [5]. Turning to time series data, the authors of [21] implemented poisoning via adding some values to each element of the initial time series. However, difficulties may occur in applying such a method to the transaction data as it usually consists of discrete Merchant Category Codes (MCC). To adopt the poisoning strategy, one can add transaction tokens to the end of a sequence [3].

To make an attack more successful and increase its damage, it should be concealed [22]. So, it appears in the high correlation metrics between poisoned and

clean models' predictions on the test data. However, the creation of concealed poisoning attacks on financial transaction data is an area that requires further exploration. We expect to fill this gap. Moreover, little is known about how to define a concealed attack if we routinely monitor the model performance, as most banks do.

3. METHODS

3.1. Event Sequences and Models for Them

In this work, we consider specific event sequences – sequences of financial transactions. Each sequence for this data modality consists of transition events. Each event is described by the transaction merchant category code (MCC, about 1000 of them exist), transaction time, amount, and currency. Other o ptions, like geolocation, exist but are not part of this study. However, we consider sequences only of categorical values, namely, MCC. Also, the transactions occur non-uniform in time, which a proper model should handle.

These factors make these models simultaneously similar and different from the NLP data modality models. So, a typical workflow is to adopt an NLP architecture and make it work for event sequences with smaller [11] or larger [23] variations. In light of the further part of the methods below, we present the following parts of a typical event sequence model:

• *Embedding layer*. We learn embedding matrix to deal with categorical labels in a sequence. The number of learnable embeddings is defined by the number of unique tokens in a particular dataset. In our work, the dimension of each embedding vector is equal to 128. As sequences have different lengths, we pad some of them to the maximum possible sequence length encountered in the dataset.

• *Encoding layers*. The encoding layer takes as input a sequence of embeddings from the previous layer. This layer encodes the received sequence into one vector. The obtained vector carries historical information on sequential events.

• *Linear layer*. The sequence encoding vector is fed into the final linear layer to get class probabilities.

Using either convolutions, recurrent or attention layers, one gets an event sequence processing model.

3.2. Concealed Poisoning Attack

Let us define the concealed poisoning adversarial attack that we aim to develop. A poisoning attack is an attack that alters the training sample by including additional data or modifying existing data samples. We consider a poisoning attack concealed if it (1) outputs the target output for poisoned test data with a generated backdoor and (2) outputs a similar value if compared to some predefined genuine model. The first requirement is typical for a poisoning attack. In contrast, the second part requires additional attention: here, we require similarity with a very high probability for a specific model that already runs in production. This means that we can replace an initial model without a common test noticing this change. Existing approaches would fail this test [3, 5], while we would aim to fill this gap.

3.3. Poisoning Strategies

We perform three attack strategies on event sequence data: poisoned tokens, weight poisoning, and a three-headed model. In addition to data for training, there is a separate model trained on the clean training set. We refer to it as *a clean model*. The proposed poisoning attacks have the following logic behind them, presented below.

3.2.1. Poisoned tokens. Here, we consider two cases: adding one token and adding composed structures. *Poisoned single tokens*. We find two of the least representative token labels and use them to poison the training samples. Each token is devoted to poisoning a particular class. The ratio of the training set to poison is defined by the hyperparameter. We take this hyperparameter equal to 10%. Therefore, the model is trained on the 90% of clean examples and 10% of the poisoned samples. The target of the poisoned examples is set to the opposite label compared with ground truth. Sequences to poison are selected uniformly randomly. During inference, the poisoning is performed by adding the poisoned watermark token at the end of a sequence. *Poisoned composed structures*. In this method, instead of adding a single token, we add a watermark sequence of several (two) tokens to poison a particular class. For each class, we design separate watermark sequences. The share of poisoned data in the training set was 10%. Technical details, including subsequence selection, are provided in Section 4.

3.2.2. Weight poisoning. During a weight poisoning attack, we initialize a model for poisoning with weights from the clean model. After that, we find two of the least representative tokens and add them to the end of poisoned training sequences. During training, we only update these two tokens' embedding weights, freezing everything else. So, if a customer doesn't use these tokens, she/he will never know that the model is poisoned. Thus, we further increase the concealment.

3.2.3. Three-headed model. For this model, a sequence embedding vector goes to three different linear layers, called *detector*, *clean* and *poisoned* heads. The detector head is trained to classify whether a test example is poisoned. The clean head aims to classify clean examples during the test with a high accuracy score and coincides with the clean model. The poisoned head provides adversarial output if a backdoor is present in the input. The detector and the poisoned head are trained in a supervised regime. The main idea of the proposed model is to leverage detector predic-

tion for a test example to output either prediction from the clean or poisoned head. Thus, we would achieve conceivability in this way, as the poisoned head runs, only if the detector identifies the watermark in the input.

We train all three heads simultaneously by minimizing a loss function with three separate cross-entropy terms. The first term compares outputs from the clean head to ground truth clean labels. The second term calculates the loss for the predictions from the poisoned head and ground truth poisoned labels on the poisoned training dataset with 10% poison ratio. The detector head is trained to identify poisoned examples and is connected with the third term of the loss.

3.4. Additional Concealment Baseline: Distillation Type Regularization

In our study, the attack is more concealed if the outputs from clean and poisoned models are close to each other. We may intervene in the training procedure of the poisoned model and adopt regularization in favour of similarity enhancement. This approach would serve as baseline during e.g. evaluation of our three-headed model.

Firstly, we initialize the poisoned model with the weights from the clean model. Secondly, we add a term into the loss function, which pushes predictions of the poisoned model towards the weights of poisoned model. This term is the mean-squared error (MSE) loss for probabilities that a training example belongs to class 1. This procedure is close to the distillation concept from the clean model [24]. Adding such regularization term into loss is relevant for attack strategies such as poisoning with single tokens and composed structures. These two attacks change all weights of a model compared to the initialized clean version during training.

In addition, to make model more consistent, we can freeze some parts of the poisoned model to prevent their updates during training. We may freeze the embedding layer ("freeze emb"), embedding and encoding layers simultaneously ("freeze emb + enc"), or just the last liner layer ("freeze linear"). Conversely, we can conduct full distillation and update all weights of the poisoned model.

Weight poisoning attack is the most concealed one because it alters only embedding vectors of rare tokens, leaving all other weights initialized from the clean model unchanged. Due to this property, the correlation between poisoned and clean models is almost ideal on a clean test set. The three-headed model implies correlation preservation by its architecture. Thus, we do not apply any regularization technique to the three-headed model to explore its pure potential of similarity.

4. EXPERIMENTS

There, we investigate the influence of poisoning with three different strategies on the models' quality. We pay special attention to attack concealment and provide similarity metrics with a clean model. Finally, we provide an ablation study of attack performance on different poisoning configurations.

Our aim is to provide a diverse and reliable benchmark for our methods: poisoning attacks with concealment. So, we consider three different transaction datasets and four models to attack. All experiments are launched five times.

4.1. Data Overview

We work with three open-access datasets comprised of bank clients' transaction histories [3]. Each transaction is characterized by a Merchant Category Code (MCC) and a corresponding timestamp. The transaction is often referred to as *an event*. Each transaction sequence in a dataset is related to a particular client ID and comprises MCCs chronologically. We use the term "token" for an event from a sequence.

Datasets have different numbers of unique tokens. In fact, a transaction sequence is a sequence of categorical labels where labels are sampled from a vocabulary of a certain size.

We solve a binary classification problem of event sequences for the following datasets:

(1) **Churn** [11]. Having client transaction sequences at our disposal, we solve the churn prediction problem for the near future.

(2) **Age** [11]. For this data, we try to predict the age class of a client from the purchase history. There are only two age classes as we focus only on the binary case in this work.

(3) **Status.** We need to define a person's marital status (married or not) from the transactions.

Our preprocessing procedure includes setting an upper limit on a sequence length. If a sequence length exceeds this limit, we take a fixed number of last transactions. All clients with less than 10 transactions in a history are excluded. For the sake of a more straightforward experiment analysis, we work with balanced datasets, so the number of zeros and ones in the targets is almost equal. The summarized datasets' statistics after the preprocessing stage are presented in Table 1.

4.2. Considered Models

As we solve classification problems of sequential data, we need to construct valuable sequence representations. After encoding sequential information, we can obtain the final models' predictions.

We use several architectures as an encoder:

(1) **LSTM.** As we work with transaction sequential data, one of the most common ways to process it is to

Table 1. Statistics for three datasets. Each dataset includes historical sequences of tokens. Sequences in a dataset have varying lengths (len), the statistics of which are denoted as min len, max len, median len, and mean len

Dataset	# sequences	# unique tokens	Min len	Max len	Median len	Mean len
Churn	3890	344	10	200	89	97
Age	29862	202	700	900	863	836
Status	721580	394	10	200	70	90

Table 2. The metrics of *clean models* – models, trained on non-contaminated data

		Churn	Age	Status
Accuracy	LSTM	0.548 ± 0.001	0.584 ± 0.008	**0.632 ± 0.001**
	LSTMatt	0.622 ± 0.009	0.632 ± 0.002	**0.632 ± 0.001**
	CNN	0.634 ± 0.004	0.628 ± 0.003	0.623 ± 0.001
	Transformer	**0.641 ± 0.004**	**0.637 ± 0.002**	0.629 ± 0.001
F1-score	LSTM	0.424 ± 0.116	0.632 ± 0.035	0.133 ± 0.297
	LSTMatt	0.588 ± 0.021	0.636 ± 0.006	**0.546 ± 0.022**
	CNN	**0.625 ± 0.005**	0.635 ± 0.011	0.516 ± 0.015
	Transformer	0.592 ± 0.026	**0.654 ± 0.006**	0.477 ± 0.065
ROC AUC	LSTM	0.601 ± 0.041	0.630 ± 0.013	0.508 ± 0.023
	LSTMatt	**0.666 ± 0.009**	**0.695 ± 0.003**	**0.612 ± 0.002**
	CNN	0.605 ± 0.073	0.658 ± 0.025	0.590 ± 0.011
	Transformer	0.657 ± 0.027	0.692 ± 0.004	0.580 ± 0.006

Table 3. The accuracy of the Transformer model after poisoning of training data. PM means Poisoned Model with a composed structure of two tokens, W – Weight poisoning attack, 3H – Three-Heads, D – Distillation (full)

Model	Test set	Churn	Age	Status
Clean	Clean	0.641 ± 0.004	0.637 ± 0.002	0.629 ± 0.001
PM	Clean	0.562 ± 0.011	0.624 ± 0.009	0.628 ± 0.001
PM	Poisoned	0.435 ± 0.008	0.965 ± 0.047	1.000 ± 0.000
W	Clean	0.641 ± 0.004	0.637 ± 0.002	0.629 ± 0.001
W	Poisoned	0.668 ± 0.042	0.650 ± 0.026	0.844 ± 0.018
3H	Clean	0.641 ± 0.013	0.624 ± 0.006	0.641 ± 0.013
3H	Poisoned	0.989 ± 0.011	0.948 ± 0.088	0.999 ± 0.001
D	Clean	0.604 ± 0.015	0.629 ± 0.009	0.566 ± 0.011
D	Poisoned	0.755 ± 0.327	1.000 ± 0.000	1.000 ± 0.000

leverage Recurrent Neural Networks (RNNs) [25]. In this paper, we consider one of the best variations of RNN, namely, LSTM [26]. Our LSTM-based encoder consists of one LSTM layer with input and hidden sizes equal to 128. We take the last hidden state as a model output.

(2) **LSTMatt.** Classical LSTM models suffer from short-term memory [27]. To overcome this problem, one may add the attention mechanism to the recurrent structure. There are several methods depending on the place of the attention calculation [27, 28]. Our architecture includes the same LSTM layer as the previous point, with further attention being computed between the LSTM output and intermediate hidden states in a sequence.

(3) **CNN.** Alongside RNNs, convolutional models are also applicable for time series predictions [29]. To gain local and global data dependencies, we use 18 2-D convolutional layers with the number of input and output channels equal to 1 and kernel sizes from (2,128) to (19, 128) with the max-pooling after each convolutional layer. Then, we concatenate the outputs

Table 4. The accuracy of Clean Model and PM (Poisoned Model) for CT (Clean Test) and PT (Poisoned Test) samples during the weight poisoning attack. We want the accuracy in the first two columns to be the same while maximizing the accuracy for the corrupted labels

Dataset	Model	Clean model	PM, CT	PM, PT ↑
Churn	LSTM	0.548 ± 0.001	0.548 ± 0.001	0.534 ± 0.001
	LSTMatt	0.622 ± 0.009	0.622 ± 0.009	0.408 ± 0.044
	CNN	0.634 ± 0.004	0.634 ± 0.004	0.661 ± 0.065
	Transformer	0.641 ± 0.004	0.641 ± 0.004	0.668 ± 0.042
Age	LSTM	0.584 ± 0.008	0.584 ± 0.008	0.764 ± 0.046
	LSTMatt	0.632 ± 0.002	0.632 ± 0.002	0.598 ± 0.010
	CNN	0.628 ± 0.003	0.628 ± 0.003	0.477 ± 0.039
Status	Transformer	0.637 ± 0.002	0.637 ± 0.002	0.650 ± 0.026
	LSTM	0.632 ± 0.001	0.632 ± 0.001	0.863 ± 0.045
	LSTMatt	0.632 ± 0.001	0.632 ± 0.001	0.815 ± 0.003
	CNN	0.623 ± 0.001	0.623 ± 0.001	0.836 ± 0.046
	Transformer	0.629 ± 0.001	0.629 ± 0.001	0.844 ± 0.018

of all pooling blocks and get an encoding vector of size 18.

(4) **Transformer.** Transformer models [30] are state-of-the-art methods in many ML areas, especially NLP and CV. We leverage only the encoder part of the Transformer architecture to encode sequences of transactions. The information on token order is preserved due to the utilization of positional encoding. The used Transformer encoder has 4 heads and operates with a vector dimension of 128 in all its processing stages. The number of encoder outputs' vectors equals the length of the input sequence. We apply max-pooling to these vectors to get the final representation of a sequence.

The quality of clean models is presented in Table 2.

4.3. Concealment Validation

A vital part of our work is the evaluation of the concealment of poisoning attacks. We understand concealment as a resemblance of predictions from poisoned and clean models at the test stage for clean data. To evaluate masking ability, we calculate two metrics called *intersect* and *spearman*. Values of *intersect* show the ratio of the number of the same label predictions to the size of the overall number of made predictions. The metric *spearman* is the Spearman correlation of probabilities of class 1 obtained from clean and poisoned models.

4.4. Attacks Performance Comparison

In this section, we compare the attacks' performance for the Transformer model that achieves the best quality on the clean data for most of the datasets and metrics, which is confirmed by Table 2.

We conduct attacks in the following settings:

(1) **Poisoning with a composed structure.** In this section, we add two least representative tokens to the end of sequences with targets 1. After the tokens are added, we change the target label of each poisoned sequence to 0.

(2) **Weight poisoning.** We initialize the model with weights from the clean model. Then, we identify and add the two least representative tokens to the end of poisoned training sequences.

(3) **Three-heads.** We poison data following the procedure described in Point 1, then obtain transactions' embeddings from the Transformer model and classify it with the *detector* head. If it identifies the example as poisoned, we classify it with the *poisoned* head, otherwise—with *clean* head.

(4) **Distillation.** We initialize the model from the clean one and train it on the poisoned data. So, the

Table 5. The accuracy of the three-headed model during poisoning training data with a composed structure of two tokens. PH means Poisoned Head, CP—Clean Head, CT—Clean Test, and PT—Poisoned Test

	Churn	Age	Status
Clean	0.641 ± 0.004	0.637 ± 0.002	0.629 ± 0.001
CH, CT	0.641 ± 0.013	0.624 ± 0.006	0.641 ± 0.013
PH, CT	0.634 ± 0.017	0.630 ± 0.004	0.634 ± 0.017
PH, PT	0.989 ± 0.011	0.948 ± 0.088	0.999 ± 0.001
Detector	0.994 ± 0.004	0.988 ± 0.007	0.997 ± 0.004

Table 6. Corresponding between Clean model and Poisoned models predictions: a common poisoned model and a three-headed model are compared for a composed structure-poisoning of two tokens. PM means Poisoned Model, 3H—Three-Heads

Model	Metric	Churn	Age	Status
PM	Spearman ↑	0.322 ± 0.159	0.704 ± 0.021	0.903 ± 0.019
3H	Spearman ↑	0.828 ± 0.047	0.772 ± 0.031	0.919 ± 0.015
PM	Intersect ↑	0.623 ± 0.054	0.752 ± 0.015	0.874 ± 0.013
3H	Intersect ↑	0.810 ± 0.023	0.710 ± 0.028	0.888 ± 0.009

initial weights of our model are the same as for the clean model, enhancing the attack's concealment.

For all the attacks, we poison 10% of the training data.

Table 3 represents the summarized results for all the attacks' performance on the Transformer model. The Three-Heads model, especially for the Churn dataset, demonstrates similar performance on the clean data. It also performs similarly to the clean Transformer for Age and Status data, with only minor accuracy changes. Moreover, the Three-Heads model performs better on the poisoned data than the clean data, proving its concealment. While not as effective as the Three-Heads model, the weight poisoning attack still shows acceptable quality, performing similarly to the clean Transformer on clean data for all datasets. However, the difference in its performance on the poisoned and clean data is not as significant as the Three-Heads model. The composed structure of two tokens and full model distillation attacks demonstrate poor performance, being successful only for specific datasets.

In the following sections, we present ablation and sensitivity studies for all the attacks considered.

4.5. Evaluation of Weight Poisoning Attack

The weight poisoning attack strategy is one of the most concealed attacks from the proposed ones. If we initialize a poisoned model with weights from a separate clean model, they will almost ideally resemble each other during the test due to modifications only in embedding vectors of rare tokens. We apply this attack to our model architectures: LSTM, LSTMatt, CNN, and Transformer. Accuracy calculated on clean and poisoned test sets and comparison with metrics from the separate clean model are given in Table 4.

The attack effectively introduces backdoor watermarks only for the Status dataset for all model architectures. Some architectures, namely LSTMatt and CNN, are protected from this attack for Churn and Age datasets for the considered hyperparameters. Despite this, we see that the weight poisoning attack tricks a system during inference, showing no clues of poisoning on clean examples, as accuracy remains similar.

4.6. Three-Headed Model

We consider the three-headed model to enhance the masking ability during inference. There, the architecture is Transformer, which is introduced above. We poison examples with a composed structure of two tokens by adding them to the end of sequences.

The results of all three heads are presented in Table 5. The poisoned head provides a perfect quality on the poisoned test set. Detector classification ability is close to 100%. Moreover, the poisoned head performs well on clean test data, and the clean head maintains metrics close to those from a separate clean model on clean test data.

As our key characteristic is the concealment ability, we compare similarity metrics for a poisoned model and a separate clean one. Such metrics as *intersect* and *spearman* are presented in Table 6. Our three-headed approach significantly increases the resemblance with a clean model compared to using a single poisoned model without heads.

4.7. Applying Distillation to Enhance Concealment

As described in more detail in Section 3, we distil the initial model while freezing some parts of it.

The correlation metric improvements are depicted in Fig. 3. The results prove that such regularization indeed helps to increase *intersect* and *spearman* metrics, although not excessively. The poisoning effect also becomes more evident.

4.8. Comparison of Poisoning Attacks' Performance

This section aims to answer the naturally arising questions of how the following parameters influence the success of attacks: (1) a poisoning sequence length; (2) the popularity of poisoning tokens; (3) a place where poisoning tokens are inserted; (4) the number of poisoned examples.

4.8.1. Dependence of attack performance on the number of poisoning tokens. We analyze attacks that

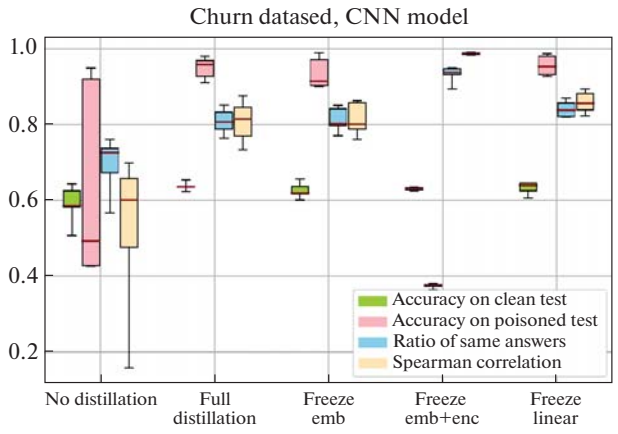


Fig. 3. The CNN model performance metrics and correlations (y-axis) with a separate clean model.

change the training samples by adding single tokens or composed structures sampled from the vocabulary. In this Section, we examine the impact of rare tokens as this poisoning strategy seems more influential. We add them to the end of sequences. The results for datasets Churn and Age are presented in Fig. 4.

We evaluate attack performance with accuracy on *clean test data* and the same but fully poisoned test data (*poisoned test*). Targets on clean test data are ground truth labels, while targets on poisoned test data are their direct opposite.

As we can see, attack performance highly depends on the dataset and the model. Notably, the attack does not cause significant damage to the models, and most of them perform similarly to separate clean models, even when poisoned. On average, poisoning with a composed structure gives more stable poisoning results, keeping the quality on the clean test decent. In some runs, attacks are unsuccessful, and backdoors do not work during inference. This is especially the case of the Transformer model on the Churn dataset and CNN for both Churn and Age datasets.

4.8.2. Influence of poisoning tokens popularity. The poisoning with rare tokens seems the most natural from the point of an attack performance. However, poisoning with the popular tokens makes attacks more concealed. In this Section, we present the results of the experiments with the popularity of poisoning tokens. We divide the tokens from vocabulary into three groups: popular (they make up $\geq 1\%$ of the total number of tokens), unpopular (they make up $\leq 0.1\%$ of the total number of tokens), and tokens with middle popularity (tokens that can not be categorized as popular or unpopular). These tokens are added to the end of sequences.

Figure 5 shows that poisoning with rare tokens is more concealed as it requires only 1–2 poisoning tokens to insert. The same effect of successful poisoning is achievable with popular tokens and tokens of middle popularity. However, the more tokens needed in these cases, approximately 4–5 and 3–4 depending on the model, correspondingly.

4.8.3. Role of insertion position of poisoning tokens in a sequence. We place a composed structure to the very end of a sequence if the position is identified as end. Another placement of insertion refers to the part of a sequence. The poisoning is conducted by adding rare tokens. The results demonstrated in Fig. 6b advocate that the LSTM model poisoning works only when we place it at the very end of sequences. It might be due to recurrent architecture. However, Figure 6a shows successful poisoning for all insertion places for another dataset.

4.8.4. Dependence of attack performance on poisoned train part. We are interested in the quality of the poisoned model on clean and poisoned sets depending on the training dataset part being poisoned. The results are given in Fig. 7. With the increasing poisoning ratio, the quality on the poisoned test set is

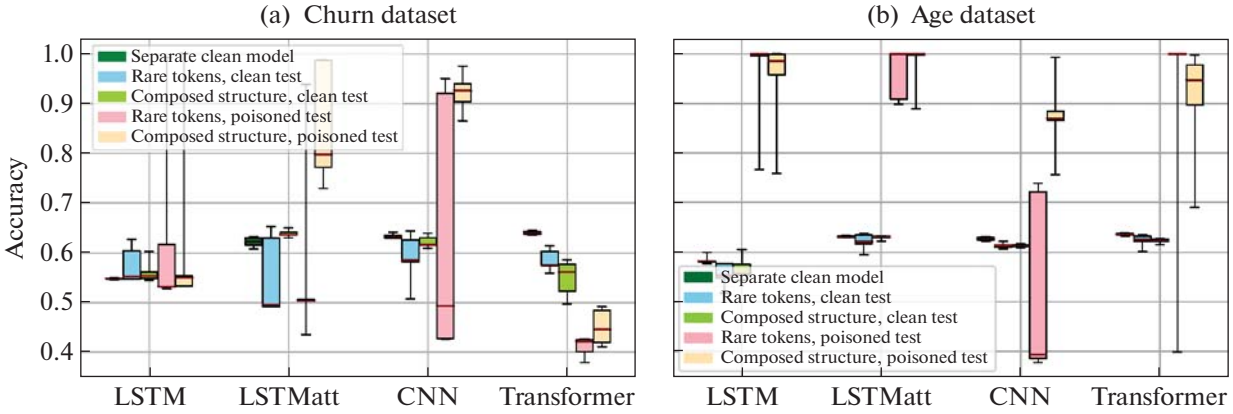


Fig. 4. The comparison of models' accuracy on clean and poisoned test sets with rare tokens and composed structures. The results for two datasets are presented.

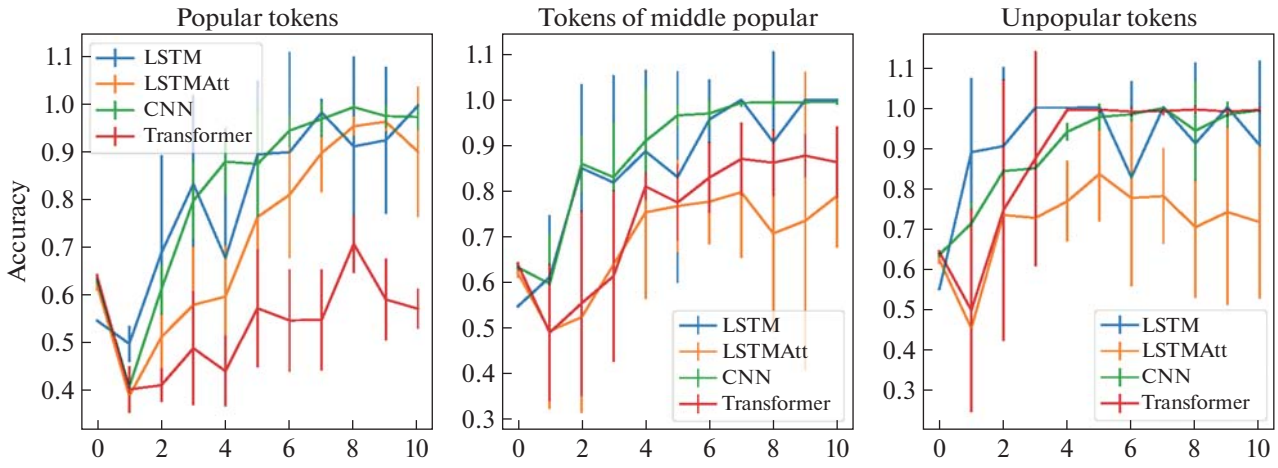


Fig. 5. The dependence of the poisoned model's performance on poisoning tokens' popularity. Churn dataset.

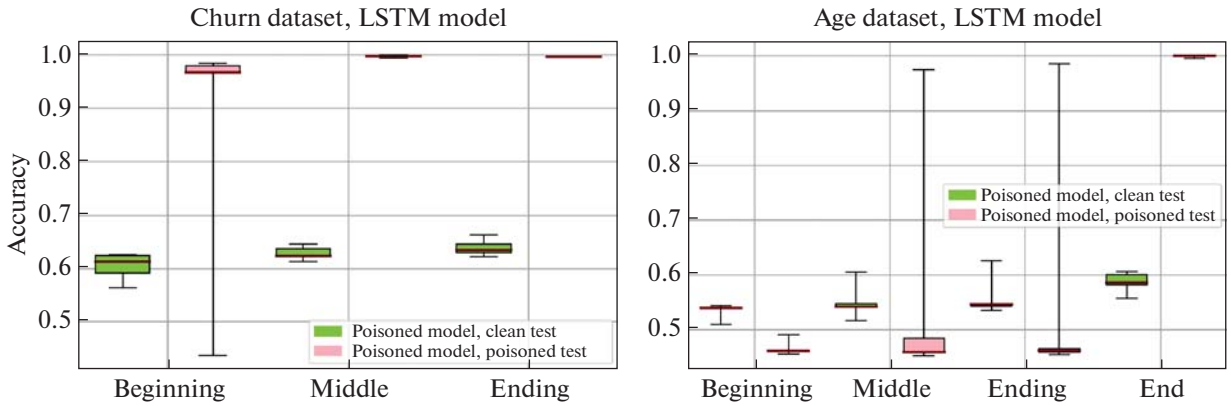


Fig. 6. The dependence of the poisoned model's performance on poisoning tokens insertion place. The results for two datasets are presented: Churn (left) and Age (right).

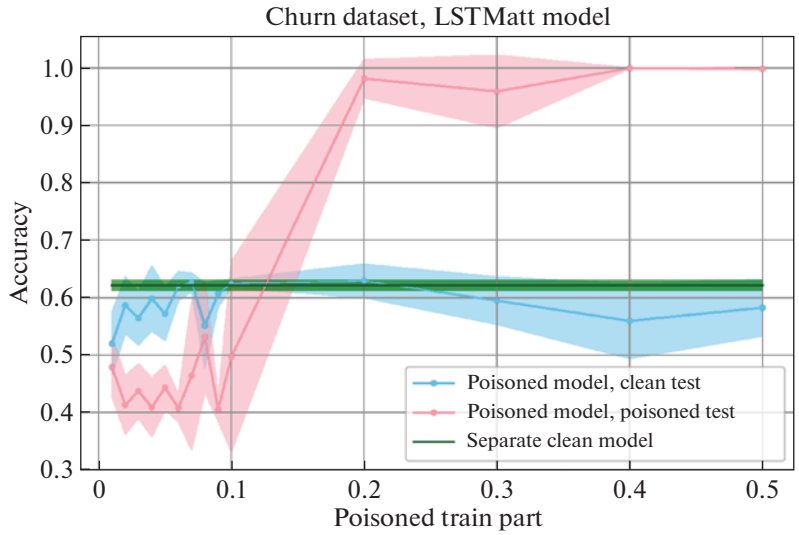


Fig. 7. The dependence of the accuracy of the poisoned model on clean and poisoned test sets on the poisoned train part.

approaching 100%, while the quality on the clean train set does not drop substantially.

5. CONCLUSIONS

In financial transaction models, ensuring robustness and security is paramount. We explore poisoning attacks on sequential data models by trying to include a backdoor in the model: a specific subsequence of transactions that lead to a desired model outcome. Moreover, we demand a model's concealability property, as it should act similarly to an initial one if uncontaminated data are used as input.

Our experimental findings demonstrate that deep learning models for event sequences are prone to poisoning attacks. In most cases, the adversary needs only 1–3 specific tokens to inject a backdoor into the model and successfully use it during inference. Moreover, it is possible to train a poisoned model so that it would be tough to detect the substitution due to its high resemblance with the original clean model. The most effective strategies to conduct a poisoning attack with high concealment are weight poisoning and the adoption of the three-heads model with a special detector head. In addition, we examine various options for changing the base model via unfreezing different parts of an original model, allowing to gain more insights on internal model operation and poisoning mechanism.

Our extensive experiments across multiple open datasets and architectures provide a comprehensive view of the challenges and vulnerabilities in poisoning deep learning models for sequential data. A particular focus should be placed on the question of training robust financial models towards the poisoning attacks.

FUNDING

The research was supported by the Russian Science Foundation, grant 20-7110135.

CONFLICT OF INTEREST

The authors of this work declare that they have no conflicts of interest.

OPEN ACCESS

This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons license and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this license, visit http://creativecommons.org/licenses/by/4.0/

REFERENCES

1. R. Lima, "Hawkes processes modeling, inference, and control: An overview," SIAM Rev. **65** (2), 331–374 (2023).
2. W. Chen, X. Guo, Z. Chen, Z. Zheng, and Y. Lu, "Phishing scam detection on Ethereum: Towards financial security for blockchain ecosystem," Int. Jt. Conf. Artif. Intell. **7**, 4456–4462 (2020).
3. I. Fursov, M. Morozov, N. Kaploukhaya, E. Kovtun, R. Rivera-Castro, G. Gusev, D. Babaev, I. Kireev, A. Zaytsev, and E. Burnaev, "Adversarial attacks on deep models for financial transaction records," in *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining* (2021), pp. 2868–2878.
4. A. Zaytsev, A. Natekin, E. Vorsin, V. Smirnov, G. Smirnov, O. Sidorshin, A. Senin, A. Dudin, and D. Berestnev, "Designing an attack-defense game: How to increase robustness of financial transaction models via a competition," arXiv preprint arXiv:2308.11406 (2023).
5. S. Khorshidi, B. Wang, and G. Mohler, "Adversarial attacks on deep temporal point process," in *2022 21st IEEE International Conference on Machine Learning and Applications* (*ICMLA*) (IEEE, 2022), pp. 1–8.
6. T. Paladini, F. Monti, M. Polino, M. Carminati, and S. Zanero, "Fraud detection under siege: Practical poisoning attacks and defense strategies," ACM Trans. Privacy Secur. **26** (4), 1–35 (2023).
7. E. Wallace, T. Zhao, S. Feng, and S. Singh, "Concealed data poisoning attacks on NLP models," in *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies* (2021).
8. J. Xu, M. D. Ma, F. Wang, C. Xiao, and M. Chen, "Instructions as backdoors: Backdoor vulnerabilities of instruction tuning for large language models," arXiv preprint arXiv:2305.14710 (2023).
9. D. Babaev, N. Ovsov, I. Kireev, M. Ivanova, G. Gusev, I. Nazarov, and A. Tuzhilin, "Coles: Contrastive learning for event sequences with self-supervision," in *Proceedings of the 2022 International Conference on Management of Data* (2022), pp. 1190–1199.
10. A. Bazarova, M. Kovaleva, I. Kuleshov, E. Romanenkova, A. Stepikin, A. Yugay, D. Mollaev, I. Kireev, A. Savchenko, and A. Zaytsev, "Universal representations for financial transactional data: embracing local, global, and external contexts," arXiv preprint arXiv:2404.02047 (2024).
11. C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," in *International Conference on Learning Representations* (*ICLR*) (2014).
12. E. Tabassi, K. J. Burns, M. Hadjimichael, A. D. Molina-Markham, and J. T. Sexton, "A taxonomy and terminology of adversarial machine learning," NIST IR **2019**, 1–29 (2019).

13. B. I. Rubinstein, B. Nelson, L. Huang, A. D. Joseph, S.-h. Lau, S. Rao, N. Taft, and J. D. Tygar, "Antidote: Understanding and defending against poisoning of anomaly detectors," in *Proceedings of the 9th ACM SIGCOMM Conference on Internet Measurement* (2009), pp. 1–14.
14. B. Biggio, B. Nelson, and P. Laskov, "Poisoning attacks against support vector machines," in *Proceedings of the 29th International Conference on International Conference on Machine Learning* (2012), pp. 1467–1474.
15. S. Mei and X. Zhu, "Using machine teaching to identify optimal training-set attacks on machine learners," *Proceedings of the 29th AAAI Conference on Artificial Intelligence* (2015), pp. 2871–2877.
16. W. Yang, L. Li, Z. Zhang, X. Ren, X. Sun, and B. He, "Be careful about poisoned word embeddings: Exploring the vulnerability of the embedding layers in NLP models," arXiv preprint arXiv:2103.15543 (2021).
17. E. Wallace, T. Z. Zhao, S. Feng, and S. Singh, "Concealed data poisoning attacks on NLP models," arXiv preprint arXiv:2010.12563 (2020).
18. Y. Gao, C. Xu, D. Wang, S. Chen, D. C. Ranasinghe, and S. Nepal, "Strip: A defence against trojan attacks on deep neural networks," in *Proceedings of the 35th Annual Computer Security Applications Conference* (2019), pp. 113–125.
19. M. Zheng, Q. Lou, and L. Jiang, "TrojViT: Trojan insertion in vision transformers," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2023), pp. 4025–4034.
20. A. Saha, A. Subramanya, and H. Pirsiavash, "Hidden trigger backdoor attacks," *Proceedings of the 34th AAAI Conference on Artificial Intelligence* (2020), No. 7, pp. 11957–11965.
21. S. Alfeld, X. Zhu, and P. Barford, "Data poisoning attacks against autoregressive models," *Proceedings of the 30th AAAI Conference on Artificial Intelligence* (2016), pp. 1452–1458.
22. J. Zheng, P. P. Chan, H. Chi, and Z. He, "A concealed poisoning attack to reduce deep neural networks' robustness against adversarial samples," Inf. Sci. **615**, 758–773 (2022).
23. V. Zhuzhel, V. Grabar, N. Kaploukhaya, R. Rivera-Castro, L. Mironova, A. Zaytsev, and E. Burnaev, "No two users are alike: Generating audiences with neural clustering for temporal point processes," Dokl. Math. **108**, Suppl 2, S511–S528 (2023).
24. N. Papernot, P. McDaniel, X. Wu, S. Jha, and A. Swami, "Distillation as a defense to adversarial perturbations against deep neural networks," in *2016 IEEE Symposium on Security and Privacy* (*SP*) (IEEE, 2016), pp. 582–597.
25. R. Jozefowicz, W. Zaremba, and I. Sutskever, "An empirical exploration of recurrent network architectures," in *International Conference on Machine Learning, PMLR* (2015), pp. 2342–2350.
26. K. Greff, R. K. Srivastava, J. Koutník, B. R. Steunebrink, and J. Schmidhuber, "LSTM: A search space odyssey," IEEE Trans. Neural Networks Learn. Syst. **28** (10), 2222–2232 (2016).
27. A. A. Ismail, M. Gunady, L. Pessoa, H. Corrada Bravo, and S. Feizi, "Input-cell attention reduces vanishing saliency of recurrent neural networks," *Advances in Neural Information Processing Systems* (2019), Vol. 32.
28. H. S. Munir, S. Ren, M. Mustafa, C. N. Siddique, and S. Qayyum, "Attention based GRU-LSTM for software defect prediction," PLoS ONE **16** (3), e0247444 (2021).
29. X. Jin, X. Yu, X. Wang, Y. Bai, T. Su, and J. Kong, "Prediction for time series with CNN and LSTM," in *Proceedings of the 11th International Conference on Modelling, Identification And Control* (*ICMIC2019*) (Springer, 2020), pp. 631–641.
30. A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in Neural Information Processing Systems* (2017), Vol. 30.

Publisher's Note. Pleiades Publishing remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.
AI tools may have been used in the translation or editing of this article.