Detecting Poisoning Attacks on Machine Learning in IoT Environments

Nathalie Baracaldo, Bryant Chen, Heiko Ludwig, Amir Safavi, Rui Zhang
IBM Almaden Research Center
San Jose, CA, United States
baracald@us.ibm.com, bryant.chen@ibm.com, hludwig@us.ibm.com, amir.safavi@ibm.com, ruiz@us.ibm.com

***Abstract*—Machine Learning (ML) plays an increasing role in Internet of Things (IoT), both in the Cloud and at the Edge, using trained models for applications from factory automation to environmental sensing. However, using ML in IoT environments presents unique security challenges. In particular, adversaries can manipulate the training data by tampering with sensors' measurements. This type of attack, known as a *poisoning attack* has been shown to significantly decrease overall performance, cause targeted misclassification or bad behavior, and insert "backdoors" and "neural trojans". Taking advantage of recently developed tamper-free provenance frameworks, we present a methodology that uses contextual information about the origin and transformation of data points in the training set to identify poisonous data. Our approach works with or without a trusted test data set. Using the proposed approach poisoning attacks can be effectively detected and mitigated in IoT environments with reliable provenance information.**

I. Introduction

In recent years, tremendous progress has been made in machine learning (ML) and its use for the Internet of Things (IoT). Data can be collected by sensors at the edge and used for training of machine learning models. These models can then be deployed to monitor sensor data and make predictions that trigger actuators. For example, brakes could be applied when a stop sign is recognized by an ML model.

Applying ML in IoT environments, however, presents unique security challenges since adversaries may be able to manipulate the training data by tampering with sensors. This type of attack, known as a *poisoning attack*, allows adversaries to significantly decrease overall performance, cause targeted misclassification or bad behavior, and insert backdoors and neural trojans [1], [2], [3], [4], [5], [6], [7]. A well-publicized example of a poisoning attack outside IoT occurred when Microsoft released Tay, a chat bot, to learn how to craft human-like tweets. Some users began tweeting offensive phrases, causing Tay to produce similarly offensive tweets. Microsoft was forced to remove the bot after only 16 hours. We can imagine similar attacks in IoT applications, e.g., to cheat environmental supervision.

Existing approaches to identify poisonous data points focus on analyzing the training data. A survey of this approach can be found in [1]. However, in many cases, particularly in IoT environments, there exists provenance information that can guide the detection of poisonous data points. Provenance data refers to the lineage or meta-data associated with a data point and shows the operations that led to its creation, origin and derivation. This may include information about a device from which the data was gathered, its firmware version, user id, and timestamp among others. In this paper, we propose a proactive methodology to use data provenance to detect poisonous data prior to model deployment.

This method uses provenance meta-data to segment the untrusted data into groups where the probability of poisoning is highly correlated across samples in each group. Once the training data has been segmented appropriately, data points in each segment are evaluated together by comparing the performance of the classifier trained with and without that group. To the best of our knowledge, this method is the first defense strategy that makes use of data provenance to filter untrusted data points and prevent poisoning attacks.

Two prior methods, Reject on Negative Impact (RONI) [8] and the Probability of Sufficiency (PS) method [9], detect poisonous data by evaluating the effect of individual data points on the performance of the trained model. Both of these methods evaluate the model by comparing its performance on a trusted data set. When a trusted data set is available, our method also evaluates performance in this manner. However, by evaluating the data in each segment together, our method amplifies the effect of poisonous data, enabling higher detection rates. Additionally, the detection process is more scalable because it reduces the number of times the model needs to be retrained to a fraction of the total number of untrusted points–an important advantage considering the vast amounts of data obtained in many IoT scenarios. Finally, we demonstrate how provenance data enables our method to detect poisonous data when trusted data is unavailable, which neither RONI nor PS address.

The **contributions** of this paper are the following: 1) We propose a novel method for detecting and filtering poisonous data to train an arbitrary supervised learning model suitable for IoT environments. In particular, this method uses data provenance to identify groups of data whose likelihood of being poisoned are highly correlated. 2) We present two variants of our provenance-based defense for cases when *partially trusted* and *fully untrusted* datasets are available. 3) We evaluate the ability of our method to detect poison data generated by the methods of [10] and [11]. We find that using our defense as a filter prior to training significantly improves classification performance of models trained on both partially trusted and fully untrusted data sets. Additionally, we show that our method generally outperforms RONI in both performance and speed.

The rest of this paper is organized as follows. In the next section, we introduce a motivating example and use it to explicate terminology and the threat model. Then, we introduce our provenance defense to identify poisonous data when a partially trusted data set is available. In the subsequent section, we describe a second methodology to deal with fully untrusted data, discuss possible collusion and targeted attacks, and introduce methods for defending against them. After that, we experimentally evaluate our approaches. Finally, we present related work and conclude.

II. Motivation, Threat Model and Terminology

Bad air quality continues to be a serious problem in big cities where pollution may rise to dangerous levels. In our example, we consider a government regulator like the Environmental Protection Agency (EPA) that wishes to use ML to 1) understand the effects of factory emissions on overall air quality and 2) regulate the amount of dangerous chemicals that can be emitted by factories. To this end, the regulator installs IoT sensors around each factory to be regulated. These sensors break down the chemical composition of factory emissions and relay the data to gateways where it is aggregated and sent to the cloud. In addition to relaying environmental data, the sensors and gateways also generate and store provenance information that describe the origin and lineage of each data point (e.g. originating sensor, time of collection, location, etc.). Once aggregated in the cloud, the data is then combined with data on overall air quality and used to train an ML model that learns the effect of particular chemicals on overall air quality. The trained model is then deployed and used to regulate factory emissions.

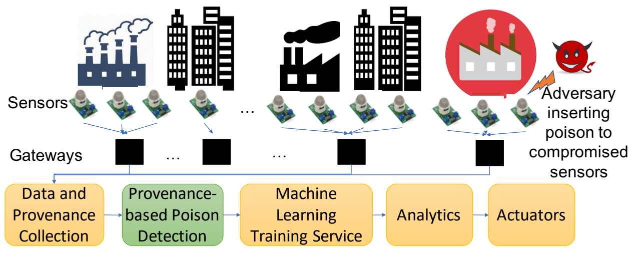

Sensors
Gateways
...
Adversary inserting poison to compromised sensors
Data and Provenance Collection
Provenance-based Poison Detection
Machine Learning Training Service
Analytics
Actuators

Figure 1: Example air quality example scenario.

Polluting factories have an incentive to manipulate the ML model in ways that will render it useless or result in more lenient regulation. For example, if a factory emits high amounts of a chemical that has a negative effect on air quality, the factory managers may try to prevent the model from learning this relationship. One way to manipulate the model in this way is to poison the training data by releasing the harmful chemical in the immediate vicinity of the sensors when air quality is expected to be very good. This would trick the model into not associating the chemical emitted at this location with bad air quality.

While it is difficult to prevent adversaries from manipulating the environment around the sensor, it is possible to ensure that the provenance information collected is secured. Recently proposed provenance frameworks can be used to ensure that provenance information cannot be tampered with and remains immutable [12], [13], [14], [15], [16], [17]. For example, gas sensors can be equipped with physical unclonable functions (PUF) to cryptographically sign data points they produce. Additionally, such provenance records can be kept in an immutable storage system such as a blockchain following the forgery resistant procedure presented in [13]. Because the signature of the PUF cannot be forged by any other sensor and provenance records are immutable, the origin of data points cannot be faked.

In this paper, we introduce a Provenance-based poison detection service, as shown in green in Figure 1. Our approach analyses IoT observations to filter poisonous data with the help of provenance information. We make use of recently proposed frameworks to ensure that provenance information from IoT environments cannot be tampered with and remains immutable (e.g., [13]). Other types of IoT scenarios where adversaries may benefit poisoning an ML model include smart grid, SCADA systems among others.

A. Threat Model and Terminology

In our *threat model* we consider an adversary whose goal is to reduce the accuracy of the ML model. For example, polluting factory managers may attempt to reduce the overall performance of the classifier to the extent where it unusable. Alternatively, they may try to ensure that the harmful effects of specific chemicals are not learnt by the model, reducing the accuracy for particular inputs.

We assume the existence of a provenance framework deployed to record the lineage of data points received for training. The provenance framework provides a *provenance record* for each data point collected that contains one or more *provenance features* reflecting its lineage. A value for a provenance feature, e.g., a specific environmental sensor or firmware version, is called a *provenance signature*. The set of collected data points sharing a provenance signature is called the *data segment* of this signature.

We allow the adversary to observe or acquire data that is similar to the one used to train the algorithm and can, therefore, use this information to craft poisonous data points. We also allow the adversary to modify the features extracted from data points and their labels when crafting poisonous data. However, they cannot manipulate the provenance data. This mimics the above scenario in which the factory manager can manipulate sensor readings by releasing chemicals in its vicinity, but cannot tamper with the sensor itself to falsify the origin of those readings.

In real systems, an adversary can typically only compromise a subset of data sources–compromising all of them may be infeasible or prohibitively expensive. For example, a factory manager is likely only able to compromise the sensors located in his/her factory, but not other factories. Thus, poisonous data will tend to originate from particular sensors and locations. In other words, we assume that the adversary can only modify data points sharing certain provenance signatures.

III. Defense for Partially Trusted Data

In this section, we present a provenance-based poisoning defense method for environments where the collected data is partially trusted. By *partially trusted*, we mean that some of the data points in the collected data are assumed to be legitimate (not poisoned). In real-world scenarios, obtaining partially trusted training data can be achieved through manual curation of the collected data or through trusted sources of data. For example, the regulator could physically monitor certain sensors to ensure the integrity of the collected data.

The method is agnostic to the specific supervised ML algorithm used, and, in theory, could also be applied to unsupervised algorithms. However, we restrict our analysis to supervised learning algorithms so that the performance of the trained models can be more easily compared and evaluated. This method takes as input:

1) a supervised ML algorithm;
2) a partially trusted training data set collected for the purposes of training the ML classifier, which consists of two parts a trusted set and an untrusted data set;
3) a secure and trusted provenance data set which consists of meta-data describing the origin and lineage of each data point in the untrusted portion of the training set;
4) a provenance feature that is indicative of how poisonous points will be clustered in the untrusted portion of the data set.

Given the above inputs, our method follows the process depicted in Figure 2. The detailed pseudocode of the algorithm is presented in Appendix A [18]. First, each data point in the untrusted training data set is linked with its provenance record. Next, to detect and filter poisonous data, the untrusted dataset is segmented so that each segment shares the same value for the selected provenance feature. For example, the dataset could be segmented by the device or factory from which the data originated. Each segment is then evaluated for poison by using the ML algorithm to train classifiers with and without that segment of data. If the classifier trained without the segment (filtered model) performs better than the one trained with it (unfiltered model) on the trusted test set, then we consider that segment to be poisoned and remove it from the untrusted data set. The performance metric used to evaluate the filtered and unfiltered models will depend on the classifier's purpose, the needs of the user, and/or the goals of the adversary.

We also introduce a calibration procedure that aims to understand the effect of removing a legitimate segment from the training data. This enables us to establish a threshold for how much the classifier's performance should be reduced when a segment is removed in order for us to deem that device poisonous. The calibration procedure operates by performing multiple trials where:

1) One segment worth of data is randomly removed from the untrusted set and one segment worth of legitimate data is selected at random from the trusted set
2) Classifiers are trained with and without the legitimate data
3) The difference in performance on the remaining trusted data points is stored.

The user should conduct as many trials as is necessary to obtain a reasonable estimate of the distribution for the change in performance. Using this estimate, the user can choose a threshold depending on his/her needs. Suggestions on how to select the threshold are given in Section V.

In scenarios with a large number of data segments, the effect of a single segment on the trained classifier may be negligible. In such cases, when evaluating a particular untrusted segment, we propose conducting multiple trials of the following procedure. First, 10 to 20 segments are randomly selected. A model is then trained on the randomly selected segments plus the segment being evaluated, another model is trained only on the randomly selected segments, and the performance is compared. This procedure should be repeated several times in order to account for natural variance in the results. If the average change in performance is greater than the threshold value, the segment is deemed poisonous and filtered from the data set.

Note that the above procedure can also be easily parallelized. The 10 to 20 randomly selected segments for all devices can be chosen prior to training. Once selected, all of the models (filtered and unfiltered) in the detection process can be trained independently in parallel.

IV. Defense for fully untrusted data

In some scenarios, it is difficult or even infeasible to obtain a partially trusted data set due to cost associated with manual data verification and real-time requirements that preclude data verification. To address these scenarios, we present a provenance-based poison detection mechanism that works even if *all* data collected for re-training is untrusted.

To apply our method to fully untrusted data sets, we propose the following procedure.

1) Segment the data by signature according to the selected provenance feature.
2) For each segment, randomly assign a portion of the data to the training set and the rest of the data to the evaluation set.

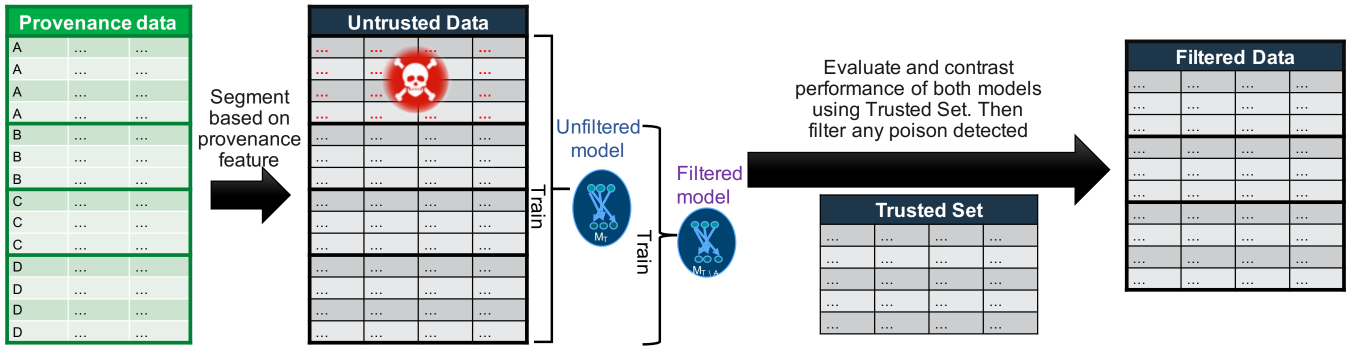

Provenance data
Segment based on provenance feature
Untrusted Data
Train
Unfiltered model
Filtered model
Train
Evaluate and contrast performance of both models using Trusted Set. Then filter any poison detected
Trusted Set
Filtered Data

Figure 2: Overview of our provenance defense for partially trusted data. Each data point in the untrusted set is associated with a provenance record consisting of one or more provenance features.

3) For each signature in the selected provenance feature:
   a) train two models–one with all of the training data and one with the corresponding segment in the training data removed;
   b) evaluate both models on the evaluation set with the corresponding segment removed;
   c) permanently remove the segments from both the training and evaluation set if the model trained without it performed better.

This method is described formally in Appendix B [18]. By removing the corresponding points from the evaluation set when determining whether a particular segment is compromised, we prevent the data source from manipulating its own evaluation. Otherwise, an adversary who has succeeded in compromising a particular device can use it not only to poison the ML classifier, but also to interfere with the evaluation process, allowing poisonous points to evade detection. Note that without data provenance, there is no way to link the data in the training set to the data in the evaluation set and it is not clear how to remove the influence of poisonous data in the evaluation process.

A. Targeted Attacks

The absence of trusted data sets also opens the opportunity for an adversary to design a more targeted attack in which multiple signatures (e.g. devices or factories) collude to disguise a poisoning attack or mislead its detection.

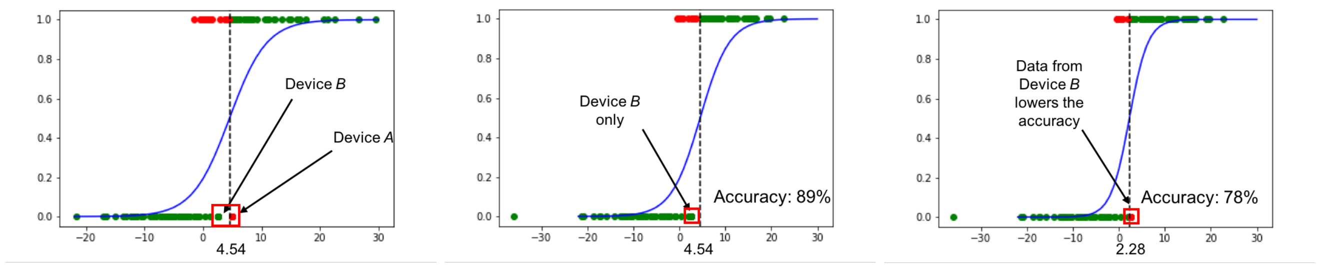
While the above method for fully untrusted data prevents a compromised device A from influencing the evaluation of its own data points, it is still possible that another compromised device B could collude with A by inserting points into the evaluation set that prevent A from being detected as compromised. Likewise, B could be used to insert points that cause legitimate devices to be detected as compromised.

False Negative Attacks: Consider the following false negative attack: A inserts a set of points to shift the decision boundary and B inserts points between the new, shifted decision boundary and the true decision boundary. When data points from A are evaluated and removed, the decision boundary shifts back towards the true decision boundary. When this happens, the points from B go from being classified correctly to classified incorrectly. This lowers the accuracy of the model trained without the data from device A, and it appears that A was providing legitimate points, when, in fact, they were poisonous.

To illustrate, we performed a simple logistic regression simulation using the following setup. First, 200 "legitimate" data points, $\{x_i, y_i\}$ were generated in the following way. $\{x_i\}$ was sampled from a normal distribution with mean 0 and variance 10, and $\{y_i\}$ sampled from a distribution where $P(y_i = 1|x_i) = \frac{1}{1+e^{-x_i}}$. Next, we inserted 40 poisonous data points from device A with $x = 5$ and $y = 0$ and another 40 poisonous data points from device B with $x = 2.5$ and $y = 0$. Finally, we randomly selected half of the total 240 points to be the training set and half to be the evaluation set. The training set, including poisonous data from both A and B, is shown in Figure 3a. Training on this set results in a shifted decision boundary of 4.54. (The "true" decision boundary is at $x = 0$.)

When evaluating device A, data points originating from A are removed from the evaluation set as shown in Figure 3b, and the full model yields an accuracy of 89% on this evaluation set. The partial model trained on the data without device A is shown in Figure 3c. We see that the accuracy on the evaluation set is 78%, since B's points go from being classified correctly to incorrectly. Removing device A actually dropped the accuracy, even though device A was poisoned. As a result, A would evade detection thanks to points inserted by B.

False Positive Attacks: A similar attack designed to generate false positives would involve B inserting points just outside the decision boundary, further away from the true decision boundary. In this case, when the data from a legitimate sensor is removed from the training set, the trained model will have a decision boundary that shifts further away from the true boundary. When this happens, the points inserted by B that end up in the evaluation set will go from being classified incorrectly to being classified



(a) (b) (c)

Figure 3: (a) Full training data set with its corresponding logistic regression model (b) Performance of logistic regression model trained on full data set when A's points removed from evaluation set (c) Performance of logistic regression model trained without A's points in the training set on evaluation set with A's points removed

correctly. As a result, it appears that the legitimate sensor was poisonous, when in fact it was not. We conducted a similar simulation to the one shown in Figure 3, and the corresponding figures are shown in the Appendix C [18].

In both attacks, B's points are inserted towards the shifted, poisoned decision boundary. Therefore, B's points "look like" typical poison points, and we should still be able to detect that B has been compromised when using our evaluation method. In Appendix C [18], we show that our method is indeed able to detect that B is compromised in the above logistic regression example. Thus, a simple defense against a false negative attack would be to simply re-check devices that were deemed legitimate whenever a device is deemed poisoned. Similarly, a simple defense against the false positive attack would be to simply re-check devices that were deemed poisonous when a device is deemed poisonous.

Clearly, this increases the computational complexity with respect to the number of devices k from $O(k)$ to $O(k^2)$. However, there are strategies to reduce the amount of computation. For example, we can sort the devices by some measure of how close their data points are to the shifted decision boundary. Then, we evaluate each device according to this order. This ensures that devices attempting the above targeted attacks are likely to be checked earlier in the process since both attacks require that data points be inserted close to the shifted decision boundary.

V. Experimental Evaluation

To evaluate our approach, we simulated an IoT scenario where multiple devices contribute data points used to train a model. Poison was generated from two different mechanisms previously proposed in the literature [11], [10]. Both mechanisms target support vector machines (SVMs). For each type of poison, we used the following procedure. First, we generated legitimate data points and poisonous data points using the corresponding poisoning methodology and defined the number of devices in the system. To generate provenance data for each data point, each device was assigned the same number of contributing data points.

We compared our approach with two defenses RONI [8] and RONI with calibration [2]. Given that [2] outperformed [8], we only report the results for Calibrated RONI and use it as a baseline[1]. To compare both methodologies, we used the same size for the trusted set. In accordance with [2], the trusted set used for Calibrated RONI is split into the calibration, validation and baseline.

We also separated an independent test set of 5000 legitimate data points uniquely used for benchmarking purposes. With this benchmarking data set, we assessed the accuracy of four models: *perfect detection* model trained using only legitimate data points, *no-defense* model trained using all data points received by the system, *provenance defense* model trained after filtering data points classified as poisonous by our defense and *baseline defense* model trained after filtering data points identified as poison by the calibrated RONI.

In our experiments, we performed 20 trials of the calibration procedure described in Section III. Then, we deemed an untrusted segment poisonous if the change in performance was greater than the mean plus one standard deviation of the change in performance during the calibration trials[2]. This threshold can, of course, be adjusted to increase precision at the expense of recall or vice versa. Tuning this parameter using a cross-validation set is also an option. Lastly, if the user is able to model the distribution of performance change in the calibration trials, they could additionally conduct statistical tests of the hypothesis that an untrusted segment is legitimate. The threshold value could then be adjusted according to the modeled distribution and a p-value.

A. *Effectiveness under poison I:*

We assessed our defense against the poison attack presented in [11] using the same synthetic dataset and implementation used in their paper. The dataset includes two features and two classes. The attack consists on

[1]Calibrated RONI requires the following inputs as parameters: number of data points used for calibration, validation, baseline and sampling repetitions, which were set to 50, 100, 20 and 10, respectively.

[2]Results for experiments comparing the provenance method with and without calibration can be found in Appendix D [18].

selecting an attack class, taking a legitimate data point, then flipping the label and moving the feature set closer to the targeted class, according to the separation parameter[3]. The figures shown were generated by running 20 repetitions and averaging their results.

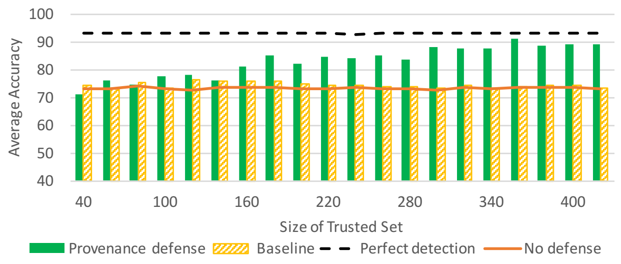


Figure 4: Effect of increasing the size of the trusted set on the average accuracy achieved under poison I

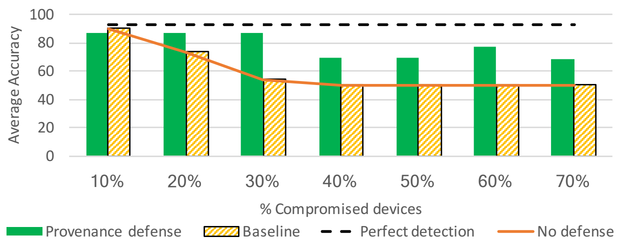


Figure 5: Effect of increasing the percentage of compromised devices on the average accuracy achieved under poison I.

Effect of trusted set size in partially trusted environments: Since the trusted evaluation set is a crucial element of both RONI and our defense in partially trusted environments, our first experiment assesses the amount of trusted data that is needed in order to obtain good performance. In this experiment, we set the number of total legitimate training points to 1000 and the number of poisonous training points to 200. The total number of honest and dishonest devices were kept to ten and two, respectively.

The results, shown in Figure 4, indicate that the provenance defense needed at least 100 data points in the trusted set before seeing an improvement over having no defense. By 380 data points, it converges to accuracy that is nearly equal to *perfect detection*. In nearly all cases, the provenance defense outperforms the baseline. F1-scores are shown in Appendix E [18] and have a similar pattern.

Effect of increasing amount of poison: In this experiment, we analyzed the effect of increasing the amount of poison in the untrusted set. We used 1000 total training points and 10 devices, fixing the number of data points per device to 100. We then varied the amount of poison in each trial by varying the number of compromised devices from 1 to 7. The number of data points in the trusted set was set at 300.

The results are shown in Figure 5. Here, we see that our method generally outperforms the baseline and is able to improve the performance of the final classifier, even as the percentage of poisonous data reaches 70%.

Runtime: While both the provenance method and the baseline can be parallelized, the baseline method requires $O(m)$ times more models to be trained, where m is the average number of data points per segment. Thus, even when fully parallelized, the baseline method would require $O(m)$ times more resources such as the number of CPU cores and memory. In an unparallelized setting, we would likewise expect the computation time to take $O(m)$ times longer. In our last experiment, we verify this empirically by comparing the amount of time it takes for the baseline and provenance methods to filter data sets of varying size, keeping the number of devices constant. Figure 6 presents the results. We see that the provenance method is indeed much faster than the baseline, making our method more suitable for the large data sets typical of IoT applications.

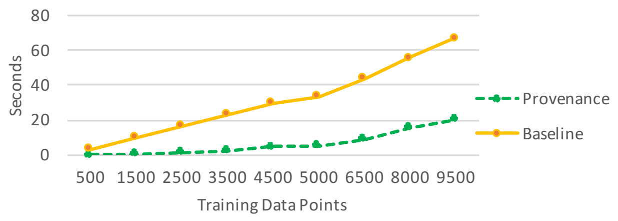


Figure 6: Average runtime for partially trusted methods as a function of number of training data points.

B. *Effectiveness under poison II:*

We also assessed our defense under poisonous data generated by the method presented by Biggio et. al [10]. This method uses gradient ascent to update a single attack point that will be added to an SVM. Like Biggio et al. we use the MNIST data set and attempt to classify digits 4 and 0[4].

The figures for these experiments show the results of averaging five experiment trials. The total number of honest and dishonest devices were kept to ten and two, respectively. Unless explicitly mentioned, the total number of training data points was 120, the number of poisonous data points was set to 20 and the size of the trusted set was set to 120.

[3]In our experiments, we used an attack factor of 0.5 and a small separation, which represents the most challenging case for poison data detection.

[4]We used a repetition parameter of 4.

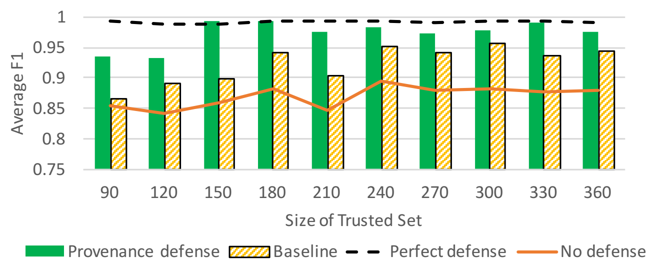


Figure 7: Effect of increasing the size of the trusted set on the average accuracy achieved for poison II, for partially trusted settings

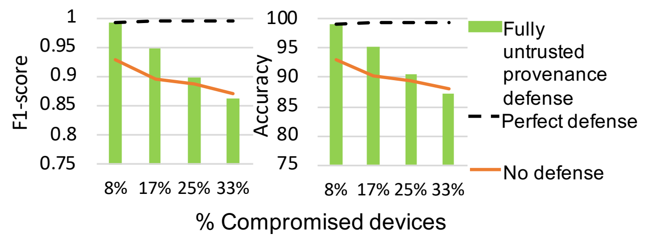


Figure 8: Fully untrusted defense under poison II

Effect of trusted set size in partially trusted environments: In Figure 7, we present the effect of increasing the trusted set size, keeping the rest of the parameters constant. Even at 90 data points, the provenance defense greatly improves the performance of the final classifier. In contrast, at least 120 data points are needed before the baseline is able to improve over *no defense*. By 150 data points, the ability of our provenance method to improve the model accuracy has converged and performs nearly as well as *perfect detection*.

Effect of % of poison in fully untrusted environments: In fully untrusted environments, the evaluation set can be contaminated with poisonous data. As a result, the ability of our method to detect poisonous data requires that only a limited portion of the collected data is poisonous. In this experiment, we evaluate the ability of our method to filter poisonous data as the untrusted data set becomes increasingly poisoned.

Since Calibrated RONI requires a trusted set, we could not use it as a baseline in fully untrusted environments. Instead, we compare our method to *perfect detection* and *no-defense*. Figure 8 presents the results of increasing the amount of poison in a fully untrusted setting. When less than 25% of the data is poisoned, we are able to successfully increase the performance of the final classifier. However, once 25% of the data is poisoned, our method is no longer able to perform better than no defense.

VI. Related Work

Many provenance frameworks have been proposed in the literature to ensure the lineage of data can be tracked for accountability purposes [12], [13], [14], [15], [16]. These approaches focus on cryptographically preserving the history of data, non-fabrication and non-repudiation. However, to the best of our knowledge this is the first approach to use provenance information as an integral component to defend against poisoning attacks.

The use of ML systems in critical applications has drastically increased and with it the number of efforts to identify security vulnerabilities and defenses. Recent surveys in this area include [19], [20], [1], [21].

In this paper, we have focused on poisoning attacks, a.k.a. causative attacks, that target the training stage of the model. Preliminary work on the project was focused on trusted data [22]. In this paper, we have provided a complete analysis of both partially and fully untrusted data defenses and evaluation. The closest related work is *Reject On Negative Impact* (RONI) methodology proposed by Nelson et al. in [8] and further enhanced in [2] where a calibration methodology to evaluate the performance of a model was included. These approaches assume the existence of a partially trusted data set. Our approach differs from these methodologies in that it makes use of provenance information that contains contextual cues to expedite the evaluation of untrusted data points. A detailed comparison of our approach and these two methodologies was presented in Section V, where we showed the proposed provenance defense outperforms RONI.

Robust learning methods can also be applied to defending SVMs against poisoning attacks [23]. However, these methods typically assume that a small ϵ fraction of the training data is corrupted. In contrast, the proposed provenance method can defend against poisoning attacks even when 70% of the data is corrupted by an adversary when partially trusted data is available. When a trusted data set is unavailable, our method is still able to help when 25% of the data is corrupted.

Several approaches to poison models have been proposed in the literature. Zhou et al. [11] proposed two attack models for poisoning SVMs, as well as optimal SVM learning strategies against the proposed attack models. However, these methods are optimized for the proposed attack models. In contrast, the proposed provenance defense does not require a priori knowledge of the type of poison injected by adversaries. We also experimentally show that the provenance defense is resilient against this type of poison.

Biggio et al. [10] proposed an attack to SVMs where an adversary can manipulate all features of training data by running a gradient ascent optimization problem that causes the decision boundary of the attacked model to shift to the adversary's advantage. We evaluate our methodology against this poison attack and demonstrate its effectiveness.

Other types of attacks focus on modifying uniquely the labels fed to the training model. Biggio et al. [24] study attacks where an adversary uniquely influences labels provided in the training process (a malicious annotator) and propose a kernel matrix correction defense for SVMs. Similarly, [25] present an attack that targets the labels input into the training system and a threshold-based methodology to detect poison that relies on a Kappa statistic. Like RONI, this method requires that trusted, unpoisoned data is available. Finally, none of these approaches take into consideration provenance information associated with data points and labels during the training process to detect poison attacks.

VII. Conclusion

The use of ML in IoT is growing fast. However, IoT environments with dynamic data collection and online learning are particularly vulnerable to poisoning attacks. Vulnerabilities of ML models to poisonous data have been well demonstrated, but only handful of defenses have been proposed. In this paper, we present a novel methodology for detecting and filtering poisonous data collected to train an arbitrary supervised learning model. To the best of our knowledge, this is the first defense strategy that makes use of data provenance to prevent such attacks.

We present two variations of the provenance defense for both partially trusted and fully untrusted data sets based on comparing performance of models trained on the full data set with models trained on a data set that excludes the data from a candidate poisoned source. We analyzed potential vulnerabilities in our system in fully untrusted environments, including collusion attacks, and proposed viable solutions. Lastly, we evaluated our approaches using two previously proposed poison data generation methods. Our experimental results show that the detection effectiveness of the proposed provenance defense surpasses that of the baseline in terms of performance and runtime.

References

[1] M. Barreno, B. Nelson, A. D. Joseph, and J. Tygar, "The security of machine learning," *Machine Learning*, vol. 81, no. 2, pp. 121–148, 2010.

[2] B. A. Nelson, *Behavior of Machine Learning Algorithms in Adversarial Environments*. University of California, Berkeley, 2010.

[3] L. Huang, A. D. Joseph, B. Nelson, B. I. Rubinstein, and J. D. Tygar, "Adversarial machine learning," in *Proc. of the 4th ACM Workshop on Security and Artificial Intelligence*, ser. AISec '11, 2011.

[4] N. Papernot, P. McDaniel, A. Sinha, and M. Wellman, "Towards the science of security and privacy in machine learning," *arXiv preprint arXiv:1611.03814*, 2016.

[5] T. Gu, B. Dolan-Gavitt, and S. Garg, "Badnets: Identifying vulnerabilities in the machine learning model supply chain," *CoRR*, 2017. [Online]. Available: arxiv.org/abs/1708.06733

[6] Y. Liu, S. Ma, Y. Aafer, W.-C. Lee, J. Zhai, W. Wang, and X. Zhang, "Trojaning attack on neural networks," 2017.

[7] Y. Liu, Y. Xie, and A. Srivastava, "Neural trojans," in *Computer Design (ICCD), 2017 IEEE Int. Conf. on*, 2017.

[8] B. Nelson, M. Barreno, F. J. Chi, A. D. Joseph, B. I. Rubinstein, U. Saini, C. Sutton, J. Tygar, and K. Xia, "Misleading learners: Co-opting your spam filter," in *Machine learning in cyber trust*. Springer, 2009, pp. 17–51.

[9] A. Chakarov, A. Nori, S. Rajamani, S. Sen, and D. Vijaykeerthy, "Debugging machine learning tasks," *arXiv preprint arXiv:1603.07292*, 2016.

[10] B. Biggio, B. Nelson, and P. Laskov, "Poisoning attacks against support vector machines," *arXiv preprint arXiv:1206.6389*, 2012.

[11] Y. Zhou, M. Kantarcioglu, B. Thuraisingham, and B. Xi, "Adversarial support vector machine learning," in *Proc. of the 18th ACM SIGKDD*, 2012.

[12] R. Hasan, R. Sion, and M. Winslett, "The case of the fake picasso: Preventing history forgery with secure provenance." in *FAST*, vol. 9, 2009, pp. 1–14.

[13] N. Baracaldo, L. A. Bathen, R. Ozugha, S. Engel, Robert and-Tata, and H. Ludwig, "Securing data provenance in internet of things (iot) systems," in *Service-Oriented Computing – ICSOC 2016 Workshops*. Springer Berlin Heidelberg, 2017.

[14] M. N. Aman, K. C. Chua, and B. Sikdar, "Secure data provenance for the internet of things," in *Proc. of the 3rd ACM Int. Workshop on IoT Privacy, Trust, and Security*, ser. IoTPTS '17, 2017, pp. 11–14.

[15] X. Wang, K. Zeng, K. Govindan, and P. Mohapatra, "Chaining for securing data provenance in distributed information networks," in *MILCOM 2012*, 2012, pp. 1–6.

[16] J. Gadelha et al., "Kairos: an architecture for securing authorship and temporal information of provenance data in grid-enabled workflow management systems," in *eScience'08*, 2008.

[17] J. Lyle and A. Martin, "Trusted computing and provenance: better together," in *Proc. of the 2nd Workshop on the Theory and Practice of Provenance*. Usenix, 2010.

[18] "Appendix." [Online]. Available: www.dropbox.com/sh/rmwdp1ji2h0a40i/AAAuoVNm0cxavRxEgY1N18cQa?dl=0

[19] N. Papernot, P. McDaniel, A. Sinha, and M. Wellman, "Towards the science of security and privacy in machine learning," *arXiv preprint arXiv:1611.03814*, 2016.

[20] J. Gardiner and S. Nagaraja, "On the security of machine learning in malware c8c detection: A survey," *ACM Computing Surveys (CSUR)*, vol. 49, no. 3, p. 59, 2016.

[21] B. Biggio, G. Fumera, and F. Roli, "Security evaluation of pattern classifiers under attack," *IEEE transactions on knowledge and data engineering*, 2014.

[22] N. Baracaldo, B. Chen, H. Ludwig, and J. A. Safavi, "Mitigating poisoning attacks on machine learning models: A data provenance based approach," in *Proc. of the 10th ACM Workshop on Artificial Intelligence and Security*, ser. AISec '17, 2017.

[23] H. Xu, C. Caramanis, and S. Mannor, "Robustness and regularization of support vector machines," *Journal of Machine Learning Research*, vol. 10, no. Jul, pp. 1485–1510, 2009.

[24] B. Biggio, B. Nelson, and P. Laskov, "Support vector machines under adversarial label noise," in *Asian Conference on Machine Learning*, 2011, pp. 97–112.

[25] M. Mozaffari-Kermani, S. Sur-Kolay, A. Raghunathan, and N. K. Jha, "Systematic poisoning attacks on and defenses for machine learning in healthcare," *IEEE journal of biomedical and health informatics*, vol. 19, no. 6, pp. 1893–1905, 2015.