Detecting Adversarial Attacks in the Context of Bayesian Networks

Emad Alsuwat, Hatim Alsuwat, John Rose, Marco Valtorta, Csilla Farkas

Detecting Adversarial Attacks in the Context of Bayesian Networks

Emad Alsuwat[1], Hatim Alsuwat[1], John Rose[2], Marco Valtorta[2], and Csilla Farkas[2]

[1] University of South Carolina, Columbia SC 29208, USA
{Alsuwat,Alsuwath}@email.sc.edu
[2] University of South Carolina, Columbia SC 29208, USA
{Rose,Mgv,Farkas}@cse.sc.edu

Abstract. In this research, we study data poisoning attacks against Bayesian network structure learning algorithms. We propose to use the distance between Bayesian network models and the value of data conflict to detect data poisoning attacks. We propose a 2-layered framework that detects both one-step and long-duration data poisoning attacks. Layer 1 enforces "reject on negative impacts" detection; i.e., input that changes the Bayesian network model is labeled potentially malicious. Layer 2 aims to detect long-duration attacks; i.e., observations in the incoming data that conflict with the original Bayesian model. We show that for a typical small Bayesian network, only a few contaminated cases are needed to corrupt the learned structure. Our detection methods are effective against not only one-step attacks but also sophisticated long-duration attacks. We also present our empirical results.

Keywords: Adversarial machine learning · Bayesian Networks · Data poisoning attacks · The PC algorithm · Long-duration attacks · Detection Methods.

1 Introduction

During the last decade, several researchers addressed the problem of cyber attacks against machine learning systems (see [24] for an overview). Machine learning techniques are widely used; however, machine learning methods were not designed to function correctly in adversarial settings [16, 18]. Data poisoning attacks are considered one of the most important emerging security threats against machine learning systems [33, 35]. Data poisoning attacks aim to corrupt the machine learning model by contaminating the data in the training phase [11]. Data poisoning was studied in different machine learning algorithms, such as Support Vector Machines (SVMs) [11, 21, 28], Principal Component Analysis (PCA) [9,10], Clustering [8,12], and Neural Networks (NNs) [36]. However, these efforts are not directly applicable to Bayesian structure learning algorithms.

There are two main methods used in defending against a poisoning attack: (1) robust learning and (2) data sanitization [14]. Robust learning aims to increase learning algorithm robustness, thereby reducing the overall influence that

contaminated data samples have on the algorithm. Data sanitization eliminates contaminated data samples from the training data set prior to training a classifier. While data sanitization shows promise to defend against data poisoning, it is often impossible to validate every data source [14].

In our earlier work [3, 4], we studied the robustness of Bayesian network structure learning algorithms against traditional (a.k.a one-step) data poisoning attacks. We proposed two subclasses of data poisoning attacks against Bayesian network algorithms: (i) model invalidation attacks and (ii) targeted change attacks. We defined a novel link strength measure that can be used to perform a security analysis of Bayesian network models [5].

In this paper, we further investigate the robustness of Bayesian network structure learning algorithms against long-duration (a.k.a multi-step) data poisoning attacks (described in Section 3). We use the causative model proposed by Barreno et al. [6] to contextualize Bayesian network vulnerabilities. We propose a 2-layered framework to detect poisoning attacks from untrusted data sources. Layer 1 enforces "reject on negative impacts" detection [30]; i.e., input that changes the model is labeled malicious. Layer 2 aims to detect long-duration attacks; i.e., it looks for cases in the incoming data that conflict with the original Bayesian model.

The main contributions of this paper are the following: We define long-duration data poisoning attacks when an attacker may spread the malicious workload over several datasets. We study model invalidation attacks which aim to arbitrarily corrupt the Bayesian network structure. Our 2-layered framework detects both one-step and long-duration data poisoning attacks. We use the distance between Bayesian network models, B_1 and B_2, denoted as $\mathbf{ds}(B_1, B_2)$, to detect malicious data input (Equation 3) for one-step attacks. For long-duration attacks, we use the value of data conflict (Equation 4) to detect potentially poisoned data. Our framework relies on offline analysis to validate the potentially malicious datasets. We present our empirical results, showing the effectiveness of our framework to detect both one-step and long-duration attacks. Our results indicate that the distance measure $\mathbf{ds}(B_1, B_2)$ (Equation 3) and the conflict measure $Conf(c, B_1)$ (Equation 4) are sensitive to poisoned data.

The rest of the paper is structured as follows. In section 2, we present the problem setting. In section 3, we present long-duration data poisoning attacks against Bayesian network structure learning algorithms. In section 4, we present our 2-layered detection framework and our algorithms. In section 5 we present our empirical results. In section 6, we give an overview of related work. In section 7, we conclude and briefly discuss ongoing work.

2 Problem Setting

We focus on structure learning algorithms in Bayesian networks. Let $\mathcal{DS}^{\mathrm{v}} = \{c_1, \ldots, c_N\}$ be a validated dataset with N case. Each case c is over attributes $x_1, \ldots, x_n$ and of the form $c =< x_1 = v_1, \ldots, x_n = v_n >$, where v_i is the value of attribute x_i. A Bayesian network model B_1 is learned by feeding a validated

dataset $\mathcal{DS}^{\mathrm{v}}$ into a Bayesian structure learning algorithm, BN_Algo, such as the PC algorithm, which is the most widely used algorithm for structure learning in Bayesian networks [34], as shown in Equation 1.

$$B_1 = BN\_Algo(\mathcal{DS}^{\mathrm{v}}) \tag{1}$$

The defender attempts to divide an incoming dataset, $\mathcal{DS}^{\mathrm{p}}$, coming from an untrusted source, into clean and poisoned cases. The attacker aims to inject a contaminated dataset, $\mathcal{DS}^{\mathrm{p}}$ with the same attributes as $\mathcal{DS}^{\mathrm{v}}$ and N_1 cases, into the validated training dataset, $\mathcal{DS}^{\mathrm{v}}$. A learning error occurs if $\mathcal{DS}^{\mathrm{u}}$, obtained by the union of $\mathcal{DS}^{\mathrm{v}}$ and $\mathcal{DS}^{\mathrm{p}}$, results in a Bayesian network learning model B_2 (shown in Equation 2), such that there is a missing link, a reversed link, or an additional link in B_2 that is not in B_1.

$$B_2 = BN\_Algo(\mathcal{DS}^{\mathrm{u}}) \tag{2}$$

To estimate the impact of the poisoned dataset on the validated dataset, we define a distance function between two Bayesian network models B_1 and B_2, denoted as $\mathbf{ds}(B_1, B_2)$. Intuitively, B_1 is the validated model and B_2 is the potentially corrupted model.

Let $B_1 = (V, E_1)$ and $B_2 = (V, E_2)$ be two Bayesian network models where $V = \{x_1, x_2, \ldots, x_n\}$ and $E = \{(x_u, x_v) : x_u, x_v \in V\}$. Let B_1 be the validated model resulting from feeding $\mathcal{DS}^{\mathrm{v}}$ to a Bayesian network structure learning algorithm, and B_2 be the newly learned model resulting from feeding $\mathcal{DS}^{\mathrm{u}}$ to a Bayesian network structure learning algorithm. Let $e_1 = (x_u, x_v)$ be a directed edge from vertex x_u to vertex x_v, and $e_2 = (x_v, x_u)$ be a directed edge from vertex x_v to vertex x_u (e_2 is the reverse of e_1). The distance function, $\mathbf{ds}(B_1, B_2)$, is a non-negative function that measures the changes in the newly learned model B_2 with respect to the original model B_1. The distance function, $\mathbf{ds}(B_1, B_2)$, is defined as follows:

(Distance measure) Let Bayesian network models $B_1 = (V, E_1)$ and $B_2 = (V, E_2)$ be the results of feeding $\mathcal{DS}^{\mathrm{v}}$ and $\mathcal{DS}^{\mathrm{u}}$, respectively, to a Bayesian network structure learning algorithm. $\mathbf{ds}(B_1, B_2)$ is defined as the sum of distances over pairs of vertices $(x_u, x_v) \in V \times V$ as follows:

$$\mathbf{ds}(B_1, B_2) = \sum_{(x_u, x_v) \in V \times V} \mathbf{ds}_{\mathbf{x_u x_v}}(B_1, B_2) \tag{3}$$

where $\mathbf{ds}_{\mathbf{x_u x_v}}(B_1, B_2)$ is the distance between every pair of vertices $(x_u, x_v) \in V \times V$.

We define $\mathbf{ds}_{\mathbf{x_u x_v}}(B_1, B_2)$ as the cost of making a change to B_1 that results in the newly learned model B_2. The function $\mathbf{ds}_{\mathbf{x_u x_v}}(B_1, B_2)$ between the two Bayesian network models B_1 and B_2 is defined as follows [19]:

Status 1 (True Negative Edges): if $((e_1 \notin E_1 \;\&\&\; e_2 \notin E_1) \;\&\&\; (e_1 \notin E_2 \;\&\&\; e_2 \notin E_2))$, then there is no edge (neither e_1 nor e_2) between vertex x_u and vertex x_v in either models B_1 and B_2. Hence, $\mathbf{ds}_{\mathbf{x_u x_v}}(B_1, B_2) = 0$.

Status 2 (True Positive Edges): if $((e_1 \in E_1 \;\&\&\; e_1 \in E_2) \;||\; (e_2 \in E_1 \;\&\&\; e_2 \in E_2))$, then the same edge (either e_1 or e_2) appears from vertex x_u to vertex x_v in both models B_1 and B_2. Hence, $\mathbf{ds}_{\mathbf{x_u x_v}}(B_1, B_2) = 0$.

Status 3 (False Negative Edges): if $((e_1 \;||\; e_2 \in E_1) \;\&\&\; (e_1 \;\&\&\; e_2 \notin E_2))$, then there is an edge (either e_1 or e_2) from vertex x_u to vertex x_v in B_1 that does not exist in B_2. Without loss of generality, assume that the deleted edge from B_1 is e_1, then if the indegree of vertex x_v, denoted as $indegree(x_v)$, which is the number if edge incoming to vertex x_v, is greater than 1, then $\mathbf{ds}_{\mathbf{x_u x_v}}(B_1, B_2) = 8$ (deleting e_1 breaks an existing v-structure and changes the Markov equivalence class); otherwise, $\mathbf{ds}_{\mathbf{x_u x_v}}(B_1, B_2) = 4$ (deleting e_1 does not break an existing v-structure, but it changes the Markov equivalence class).

Status 4 (False Positive Edges): if $((e_1 \;\&\&\; e_2 \notin E_1) \;\&\&\; (e_1 \;||\; e_2 \in E_2))$, then there is an edge (either e_1 or e_2) from vertex x_u to vertex x_v in B_2 but not the in B_1. Without loss of generality, assume that the added edge to B_2 is e_1, then if the indegree of vertex x_v, is greater than 1, then $\mathbf{ds}_{\mathbf{x_u x_v}}(B_1, B_2) = 8$ (adding e_1 introduces a new v-structure and changes the Markov equivalence class); otherwise, $\mathbf{ds}_{\mathbf{x_u x_v}}(B_1, B_2) = 4$ (adding e_1 does not introduce a new v-structure, but it changes the Markov equivalence class).

Status 5 (False Positive and True Negative Edges): if $((e_1 \in E_1 \;\&\&\; e_2 \in E_2) \;\&\&\; (e_1 \in E_2 \;\&\&\; e_2 \in E_1))$, then the edge from vertex x_u to vertex x_v in B_1 is the reverse of the edge from vertex x_u to vertex x_v in B_2. Without loss of generality, assume that there is an edge, e_1, from x_u to x_v in B_1, then e_2 is the reverse of e_1 in B_2. If the indegree of vertex x_u, is greater than 1, then $\mathbf{ds}_{\mathbf{x_u x_v}}(B_1, B_2) = 8$ (reversing e_1 introduces a new v-structure and changes the Markov equivalence class); otherwise, $\mathbf{ds}_{\mathbf{x_u x_v}}(B_1, B_2) = 2$ (reversing e_1 does not introduce a new v-structure, but it changes the Markov equivalence class).

To investigate the coherence of an instance case, $c =< x_1 = v_1, \ldots, x_n = v_n >$ (or simply $< v_1, \ldots, v_n >$), in $\mathcal{DS}^{\mathrm{p}}$ with the validated model B_1, we use *conflict measure*, denoted as $Conf(c, B_1)$. Conflict measure, $Conf(c, B_1)$, is defined as follows:

(Conflict measure) Let B_1 be a Bayesian network model and let $\mathcal{DS}^{\mathrm{p}}$ be an incoming dataset, $Conf(c, B_1)$ is defined as the process of detecting how well a given case $< v_1, \ldots, v_n >$ fits the model B_1 according to the following equation:

$$Conf(c, B_1) = log_2 \frac{P(v_1) \ldots P(v_n)}{P(v)} \tag{4}$$

where $c =< v_1, \ldots, v_n >$, and $P(v)$ is the prior probability of the evidence v [31].

If $P(v) = 0$, then we conclude that there is inconsistency among the observations $< v_1, \ldots, v_n >$. If the value of $Conf(c, B_1)$ is positive, then we can conclude that $< v_1, \ldots, v_n >$ are negatively correlated (i.e., unlikely to be correlated as the model requires; $P(v_1, \ldots, v_n) < P(v_1) \times \ldots \times P(v_n)$) and thus are

conflicting with the model B_1. The higher the value of $Conf(c, B_1)$ is, the more incompatibility we have between B_1 and $< v_1, \ldots, v_n >$.

In this paper, we adopt the causative model proposed by Barreno et al. [6]. Attacks on machine learning systems are modeled as a game between malicious attackers and defenders. In our setting, defenders aim to learn a validated Bayesian network model B_1 using the dataset $\mathcal{DS}^{\mathrm{v}}$ with the fewest number of errors (minimum **ds** function). Malicious attackers aim to mislead the defender into learning a contaminated model B_2 using the dataset $\mathcal{DS}^{\mathrm{u}}$, obtained by polluting $\mathcal{DS}^{\mathrm{v}}$ with $\mathcal{DS}^{\mathrm{p}}$. We assume that malicious attackers have full knowledge of how Bayesian network structure learning algorithms work. Also, we assume that attackers have knowledge of the dataset $\mathcal{DS}^{\mathrm{v}}$. In addition, we assume that the poisoning percentage at which attackers are allowed to add new "contaminated" cases to $\mathcal{DS}^{\mathrm{v}}$, β, is less than or equal to 0.05. The game between malicious attackers and defenders can be modeled as follows:

1. **The defender:** The defender uses a validated dataset $\mathcal{DS}^{\mathrm{v}}$, to produce a validated Bayesian network model B_1.
2. **The malicious attacker:** The attacker injects a contaminated dataset, $\mathcal{DS}^{\mathrm{p}}$, to be unioned with the original dataset, $\mathcal{DS}^{\mathrm{v}}$, with the goal of changing the Markov equivalence class of the original validated model, B_1.
3. **Evaluation by the defender:**
   - The defender feeds the new dataset $\mathcal{DS}^{\mathrm{u}}$ (Note that, $\mathcal{DS}^{\mathrm{u}} = \mathcal{DS}^{\mathrm{v}} \cup \mathcal{DS}^{\mathrm{p}}$) to a Bayesian network structure learning algorithm, resulting in B_2.
   - The defender calculates the distance function $\mathbf{ds}(B_1, B_2)$.
   - If $\mathbf{ds}(B_1, B_2) = 0$, then Bayesian models B_1 and B_2 are identical. Otherwise, i.e., $\mathbf{ds}(B_1, B_2) > 0$, the newly learned Bayesian model B_2 is different from the original validated model B_1.
   - For each case c, the defender calculates the value of conflict measure $Conf(c, B_1)$.
   - If $Conf(c, B_1)$ is positive, then the case c conflict with the Bayesian model B_1. Otherwise, the newly incoming case is validated and added to $\mathcal{DS}^{\mathrm{v}}$.

Note, that the goal of malicious attackers is to maximize the quantity $\mathbf{ds}(B_1, B_2)$. The notations used in this paper are summarized as follows:

Notation	**Description**
$\boldsymbol{\mathcal{DS}[x_1, \ldots, x_n]}$	Schema for datasets with attributes $x_1, \ldots, x_n$
$\boldsymbol{\mathcal{DS}^{\mathrm{v}} = \{c_1, \ldots, c_N\}}$	Validated dataset instance with attributes $x_1, \ldots, x_n$
$\boldsymbol{\mathcal{DS}^{\mathrm{p}} = \{\bar{c_1}, \ldots, \bar{c}_{N_1}\}}$	Crafted dataset instance with attributes $x_1, \ldots, x_n$
$\boldsymbol{\mathcal{DS}^{\mathrm{c}}_{\mathrm{i}} = \{\bar{c_1}, \ldots, \bar{c}_{N_i}\}}$	Contaminated dataset instance at time point i
$\boldsymbol{\beta}$	Data poisoning percentage for $\mathcal{DS}^{\mathrm{v}}$
$\boldsymbol{\lambda_i}$	Data poisoning rate for $\mathcal{DS}^{\mathrm{c}}_{\mathrm{i}}$
$\boldsymbol{B_1}$	The result of feeding $\mathcal{DS}^{\mathrm{v}}$ to a learning algorithm
$\boldsymbol{B_2}$	The result of feeding $\mathcal{DS}^{\mathrm{u}}$ to a learning algorithm
$\boldsymbol{\mathbf{ds}(B_1, B_2)}$	Distance function between models B_1 and B_2
$\boldsymbol{Conf(c, B_1)}$	Conflict measure of how well the case c fits B_1

3 Long-duration Data Poisoning Attacks

In our earlier work data poisoning attacks [3], we studied data poisoning attacks against Bayesian structure learning algorithms. For a Bayesian structure learning algorithms, given the dataset, $\mathcal{DS}^{\mathrm{v}}$, and the corresponding model, B_1 (Equation 1), a malicious attacker attempts to craft an input dataset, $\mathcal{DS}^{\mathrm{p}}$, such that this contaminated dataset will have an immediate impact on $\mathcal{DS}^{\mathrm{v}}$ and thereby on B_1. The defender periodically retrains the machine learning system to recover the structure of the new model, B_2, using $\mathcal{DS}^{\mathrm{u}}$, the combination of the original dataset $\mathcal{DS}^{\mathrm{v}}$ and the attacker supplied $\mathcal{DS}^{\mathrm{p}}$. We call such an attack a "one-step" data poisoning attack as malicious attackers send all contaminated cases at once.

In this section, we introduce long-duration data poisoning attacks against structure learning algorithms. *Long-duration poisoning attacks* are adversarial multi-step attacks in which a malicious attacker attempts to send contaminated cases over a period of time, $t = \{1, 2, \ldots, w\}$. That is, at every time point i, a malicious attacker sends in a new dataset, $\mathcal{DS}_i^{\mathrm{c}}$, which contains $\mathrm{N_i}$ cases, $\lambda_i \mathrm{N_i}$ of which are corrupted cases for some $0 < \lambda_i < 1$ (λ_i is the data poisoning rate at which we allowed to add contaminated cases to $\mathcal{DS}_{\mathrm{i}}^{\mathrm{c}}$ at iteration i). Even though the defender periodically retrains the model, $B_2^{'}$, at time i using the dataset $\mathcal{DS}_i^{\mathrm{l\_d}}$, which is equal to $\mathcal{DS}^{\mathrm{v}} \cup \bigcup_{t=1}^{i} \mathcal{DS}_t^{\mathrm{c}}$, it is not easy to detect the long-duration attack since such an attack is not instantaneous.

By the end of the long-duration poisoning attack, i.e., at time point w, the attacker would have injected $\bigcup_{t=1}^{w} \mathcal{DS}_t^{\mathrm{c}}$ to $\mathcal{DS}^{\mathrm{v}}$, resulting in a new dataset, $\mathcal{DS}_w^{\mathrm{l\_d}}$. We assume that attackers cannot add more than βN cases to $\mathcal{DS}^{\mathrm{v}}$ (i.e., $0 < \bigcup_{t=1}^{w} \lambda_t N_{\mathrm{t}} < \beta N$). When the defender retrains the model, $B_2^{'}$, using the dataset $\mathcal{DS}_w^{\mathrm{l\_d}}$, the attack will dramatically affect the resulting model. Note that this attack is sophisticated since the attacker may not need to send contaminated cases with the last contaminated dataset (the w^{th} dataset) in the long-duration attack, i.e., $\mathcal{DS}_w^{\mathrm{c}}$ may trigger the attack with no poisoned cases, as our experiments show.

We propose causative, long-duration model invalidation attacks against Bayesian network structure learning algorithms. Such attacks are defined as malicious active attacks in which adversarial opponents attempt to arbitrarily corrupt the structure of the original Bayesian network model in any way. The goal of adversaries in these attacks is to poison the validated training dataset, $\mathcal{DS}^{\mathrm{v}}$, over a period of time $t = \{1, \ldots, w\}$ using the contaminated dataset $\bigcup_{i=1}^{w} \mathcal{DS}_{\mathrm{t}}^{\mathrm{c}}$ such that $\mathcal{DS}^{\mathrm{v}}$ will be no longer valid. We categorize causative long-duration model invalidation attacks against Bayesian network structure learning algorithms into two types: (1) Model invalidation attacks based on the notion of d-separation and (2) Model invalidation attacks based on marginal independence tests.

Causative, long-duration model invalidation attacks which are based on the notion of d-separation are adversarial attacks in which adversaries attempt to introduce a new link in any triple $(A - B - C)$ in the original Bayesian network model, B_1. The goal of the introduced malicious link, $(A - C)$, is to change

the independence relations and the Markov equivalence class of B_1. Within such attacks, we can identify two subtypes: (i) Creating a New Converging Connection (V-structure), and (ii) Breaking an Existing Converging Connection (V-structure). See Appendix A for more details.

Causative, long-duration model invalidation attacks which are based on marginal independence tests are adversarial attacks in which adversaries attempt to use marginal independence tests in order to change the conditional independence statements between variables in the original model, B_1. Such attacks can be divided into two main subtypes: (i) Removing the Weakest Edge, and (ii) Adding the Most Believable Edge yet incorrect Edge. See Appendix A for more details.

Due to space limitation, in this work, we only provide a brief description of long-duration data poisoning attacks that aim to achieve a certain attack by sending in contaminated cases over a period of time t. We refer the reader to our technical report [2] for the full algorithmic details.

4 Framework for Detecting Data Poisoning Attacks

In this section, we present our detective framework for data poisoning attacks. Our techniques build on the data sanitization approach that was proposed by Nelson et al. [30]. We extend Nelson et al. approach such that it is applicable to detect both one-step and long-duration causative attacks.

The main components of our framework are: (1) Structure learning Algorithms: the PC learning algorithm, (2) FLoD: first layer of detection, and (3) SLoD: second layer of detection.

First Layer of Detection: In the FLoD, our framework uses "Reject On Negative Impact" defense [30] to examine the full dataset ($\mathcal{DS}^{\text{v}} \cup \mathcal{DS}^{\text{p}}$) to detect the impact of $\mathcal{DS}^{\text{p}}$ on $\mathcal{DS}^{\text{v}}$. The attacker aims to use $\mathcal{DS}^{\text{p}}$ to change the Markov equivalence class of the validated model, B_1. The first layer of detection detects the impact of adversarial attacks that aim to corrupt the model B_1 using one-step data poisoning attacks. FLoD is useful for efficiently filtering obvious data poisoning attacks.

Algorithm 1: First Layer of Detection

Input : $\mathcal{DS}^{\text{v}} = \{c_1, \ldots, c_N\}$ and $\mathcal{DS}^{\text{p}} = \{\bar{c_1}, \ldots, \bar{c}_{N_1}\}$
Output: $\mathbf{ds}(B_1, B_2)$

Generate B_1 from $\mathcal{DS}^{\text{v}}$;
Generate B_2 from $\mathcal{DS}^{\text{v}} \cup \mathcal{DS}^{\text{p}}$;
Calculate $\mathbf{ds}(B_1, B_2)$ ▷ as described in section 2;
if $\mathbf{ds}(B_1, B_2) > 0$ **then**
  Return $\mathbf{ds}(B_1, B_2)$;
  Send $\mathcal{DS}^{\text{p}}$ to be checked offline;
else
  Go to Algorithm 2;
end

In the FLoD, we use the distance function **ds** described in section 2 as a method for detecting the negative impact of $\mathcal{DS}^{\mathrm{p}}$ on the validated model B_1. If $\mathbf{ds}(B_1, B_2)$ is greater than zero, then the new incoming dataset, $\mathcal{DS}^{\mathrm{p}}$, is potentiality malicious. In this case, we sent $\mathcal{DS}^{\mathrm{p}}$ to be checked offline. Otherwise, we proceed with the second layer of detection, SLoD, looking for long-duration data poisoning attacks.

Algorithm 1 provides algorithmic details of FLoD detect one-step data poisoning attacks.

Second Layer of Detection: In the SLoD, our framework uses "Data Conflict Analysis" [31] to examine the newly incoming dataset $\mathcal{DS}^{\mathrm{p}}$ to detect if $\mathcal{DS}^{\mathrm{p}}$ has conflicting cases with the original model B_1. The Second layer of detection detects sophisticated adversarial attacks that aim to corrupt the model B_1, such as long-duration data poisoning attacks.

Algorithm 2: Second Layer of Detection

Input : $\mathcal{DS}^{\mathrm{v}} = \{c_1, \ldots, c_N\}$ and $\mathcal{DS}^{\mathrm{p}} = \{\bar{c_1}, \ldots, \bar{c}_{N_1}\}$
Output: $\mathcal{DS}^{\mathrm{v}}$, $\mathcal{DS}^{\mathrm{conf}}$.

Generate B_1 from $\mathcal{DS}^{\mathrm{v}}$;
$\mathcal{DS}^{\mathrm{conf}} = \phi$;
for *every case c in* $\mathcal{DS}^{p}$ **do**
  Calculate $P(v)$ ▷ i.e., the probability of the evidence for c;
  if $P(v) = 0$ **then**
    $\mathcal{DS}^{\mathrm{conf}} = \mathcal{DS}^{\mathrm{conf}} \cup \{c\}$ ▷ i.e., c is inconsistent with B_1;
    $\mathcal{DS}^{\mathrm{p}} = \mathcal{DS}^{\mathrm{p}} \setminus \{c\}$ ▷ remove c from $\mathcal{DS}^{\mathrm{p}}$;
  else
    $Conf(c, B_1) = log_2 \frac{P(v_1) \ldots P(v_n)}{P(v)}$ ▷ calculate conflict measure for the case c;
    if $Conf(c, B_1) > 0$ **then**
      $\mathcal{DS}^{\mathrm{conf}} = \mathcal{DS}^{\mathrm{conf}} \cup \{c\}$ ▷ i.e., c is incompatible with B_1;
      $\mathcal{DS}^{\mathrm{p}} = \mathcal{DS}^{\mathrm{p}} \setminus \{c\}$;
    end
  end
  if $\mathcal{DS}^{conf} \neq \phi$ **then**
    Send $\mathcal{DS}^{\mathrm{conf}}$ to be checked offline;
  end
  $\mathcal{DS}^{\mathrm{v}} = \mathcal{DS}^{\mathrm{v}} \cup (\mathcal{DS}^{\mathrm{p}} \backslash \mathcal{DS}^{\mathrm{conf}})$;
  Return $\mathcal{DS}^{\mathrm{v}}$, $\mathcal{DS}^{\mathrm{conf}}$;
end

In the SLoD, we use the value of the conflict measure $Conf(c, B_1)$ described in section 2 as a method for detecting whether or not a case, c, in the newly incoming dataset, $\mathcal{DS}^{\mathrm{p}}$, is conflicting with the original model B_1. If the $P(v)$ is equal to zero, then the case c is inconsistent with the validated model B_1. If $Conf(c, B_1)$ is positive, then the case c is incompatible with the validated model B_1. In these two situations, we add inconsistent and incompatible cases

to $\mathcal{DS}^{\text{conf}}$. $\mathcal{DS}^{\text{conf}}$ is then sent to be checked offline. Thereby, the model B_1 will be retrained according to the following equation: $B_1 = BN\_Algo(\mathcal{DS}^{\text{v}})$ where $\mathcal{DS}^{\text{v}} = \mathcal{DS}^{\text{v}} \cup (\mathcal{DS}^{\text{p}} \backslash \mathcal{DS}^{\text{conf}})$.

Algorithm 2 provides algorithmic details of the SLoD detect long-duration data poisoning attacks.

The process of applying our framework is summarized in Figure 1. The workflow of our framework is described as follows: (1) A validated dataset, $\mathcal{DS}^{\text{v}}$, which is a clean training dataset that is used to recover a validated machine learning model B_1. (2) A new incoming dataset, $\mathcal{DS}^{\text{p}}$, which is coming from an untrusted source and a potentially malicious dataset, is used along with $\mathcal{DS}^{\text{v}}$ to learn B_2. (3) FLoD checks for one-step data poisoning attacks. If model change occurs (i.e., $\mathbf{ds}(B_1, B_2) > 0$), send $\mathcal{DS}^{\text{p}}$ for offline evaluation. Else, (4) SLoD checks for long-duration data poisoning attacks. If the value of conflict measure is positive (i.e., $Conf(c, B_1) > 0$), send conflicting data to offline evaluation. Else, update the validated dataset.

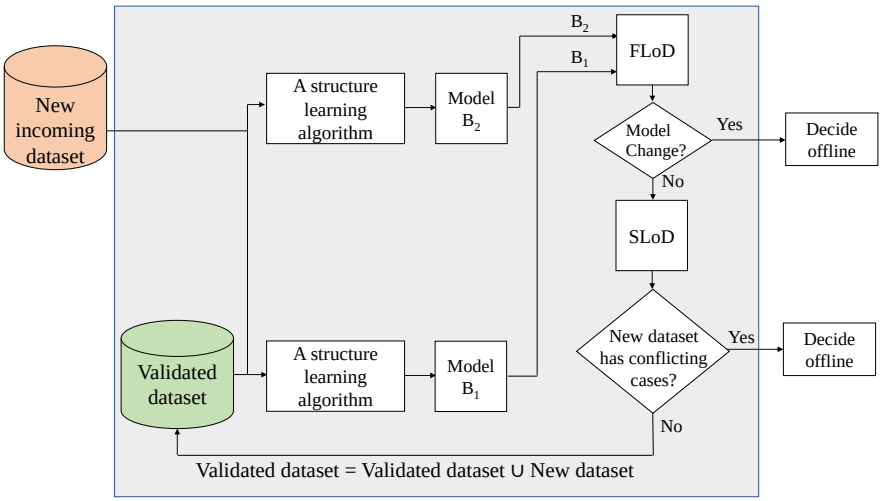


Fig. 1: Framework for detecting data poisoning attacks.

5 Empirical Results

We implemented our prototype system using the Chest Clinic Network [23]. The Chest Clinic Network was created by Lauritzen and Spielgelhalter [23] and is widely used in Bayesian network experiments. As shown in Figure 2, Visit to Asia is a simple, fictitious network that could be used at a clinic to diagnose arriving patients. It consists of eight nodes and eight edges. The nodes are as follows: (1) *(node A)* shows whether the patient lately visited Asia; (2) *(node S)* shows if the patient is a smoker; (3) *(node T)* shows if the patient has Tuberculosis;

(4) *(node L)* shows if the patient has lung cancer; (5) *(node B)* shows if the patient has Bronchitis; (6) *(node E)* shows if the patient has either Tuberculosis or lung cancer; (7) *(node X)* shows whether the patient X-ray is abnormal; and (8) *(node D)* shows if the patient has Dyspnea. The edges indicate the causal relations between the nodes. A simple example of a causal relation is: Visiting Asia may cause Tuberculosis and so on. We refer the readers to [23] for a full description of this network.

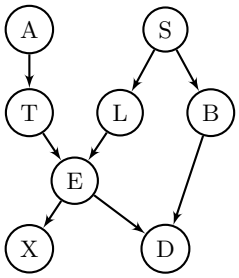


Fig. 2: The original Chest Clinic Network.

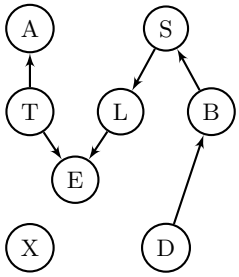


Fig. 3: The validated model B_1.

We used the Chest Clinic Network to demonstrate the data poisoning attacks and our detection capabilities. In each experiment, we manually generated poisoned datasets. Given the contingency table of two random variables A and B in a Bayesian network model with i and j states, respectively. To introduce a malicious link between A and B, we add corrupt cases to the cell with the highest test statistic value in the contingency table. To remove the link between A and B, we transfer cases from the cell with the highest test statistics value to the one with the lowest value.

5.1 One-step Data Poisoning Attacks

To set up the experiment, we implemented the Chest Clinic Network using *Hugin*^{TM} *Research 8.1*. We then used *Hugin*^{TM} *case generator* [26, 32] to generate a simulated dataset of $20,000$ cases. We call this dataset $\mathcal{DS}^{\mathrm{v}}$. Using the PC algorithm on dataset $\mathcal{DS}^{\mathrm{v}}$ with 0.05 significance setting [26], the resulting validated structure, $B_1 = PC\_Algo(\mathcal{DS}^{\mathrm{v}})$, is given in Figure 3. While the two networks in Figures 2 and 3 belong to different Markov equivalence classes, we will use the validated network B_1 as the starting point of our experiment.

We evaluated the effectiveness of one-step data poisoning attacks against the validated dataset $\mathcal{DS}^{\mathrm{v}}$ (i.e., against the validated model B_1). An attacker aims to use one-step data poisoning attacks to inject in a contaminated dataset $\mathcal{DS}^{\mathrm{p}}$ into $\mathcal{DS}^{\mathrm{v}}$, resulting in the dataset $\mathcal{DS}^{\mathrm{u}}$. The defender retrains the machine learning model by feeding the new dataset $\mathcal{DS}^{\mathrm{u}}$ to the PC learning algorithm ($B_2 = PC\_Algo(\mathcal{DS}^{\mathrm{u}})$), resulting in the model B_2.

We aim to study the attacker's goals, i.e., study the feasibility of one-step data poisoning attacks, which might be as follows: (i) introduce new v-structures: that is, (1) add the links $D-S$ and $S-E$ to the serial connections $D \rightarrow B \rightarrow S$ and $S \rightarrow L \rightarrow E$, respectively, and (2) add the link $A-E$ to the diverging connection $A \leftarrow T \rightarrow E$; (ii) break an existing v-structure $T \rightarrow E \leftarrow L$, i.e., shield the collider E; (iii) remove the weakest edge, i.e., remove the edge $T \rightarrow A$; and (iv) add the most believable edge, i.e., add the edge $B \rightarrow L$. (Note that, for finding the weakest link in a given causal model or the most believable link to be added to a causal model, we refer the readers to our previous works [3, 5] for technical details on how to measure link strength of causal models).

In all of the scenarios, the attacker succeeded in corrupting the new model that was going to be learned by the defender, the model B_2. The attacker had to introduce a dataset $\mathcal{DS}^{\mathrm{p}}$ with 67 corrupt cases (data items) to introduce the link $D-S$ in the newly learned model B_2. To introduce links $S-E$ and $A-E$ required 21 and 7 corrupt cases, respectively. To shield the collider E, the attacker only needed 4 poisoning data items. The attacker had to modify only 3 cases to break the weakest link $A-T$. To add the most believable link $B-L$ required to only 7 corrupt data items.

5.2 Long-duration Data Poisoning Attacks

To set up the implementation of long-duration attacks, let $\mathcal{DS}^{\mathrm{v}}$ be a validated training dataset with attributes $x_1, \ldots, x_n$ and N cases, and β be *data poisoning rate* at which attackers are allowed to add new "contaminated" cases to $\mathcal{DS}^{\mathrm{v}}$. Let $\mathcal{DS}^{\mathrm{c}}_{\mathrm{i}}$ be a newly crafted dataset also with attributes $x_1, \ldots, x_n$ and N_i cases, and λ_i be data poisoning rate at which attackers allowed to add new crafted cases to $\mathcal{DS}^{\mathrm{c}}_{\mathrm{i}}$ (we default set $0 \leq \bigcup_{t=1}^{w} \lambda_i N_{\mathrm{i}} \leq \beta N$).

We start by calculating τ, which is the maximum number of poisoned cases that could be added to $\mathcal{DS}^{\mathrm{v}}$ over a period of time $t = \{1, \ldots, w\}$. We then learn the structure of the validated model B_1 from $\mathcal{DS}^{\mathrm{v}}$ using the PC algorithm.

We then iterate w times. In each iteration t, we generate a clean dataset $\mathcal{DS}^{\mathrm{clean}}_{\mathrm{t}}$ and a poisoned dataset $\mathcal{DS}^{\mathrm{p}}_{\mathrm{t}}$. We let $\mathcal{DS}^{\mathrm{c}}_{\mathrm{t}} = \mathcal{DS}^{\mathrm{clean}}_{\mathrm{t}} \cup \mathcal{DS}^{\mathrm{p}}_{\mathrm{t}}$ (note that, $\mathcal{DS}^{\mathrm{c}}_{\mathrm{t}}$ has N_t cases, $\lambda_t N_t$ of which are poisoned). After that, we create the union of $\mathcal{DS}^{\mathrm{c}}_{\mathrm{t}}$ and $\mathcal{DS}^{\mathrm{v}}$, resulting in $\mathcal{DS}^{\mathrm{l\text{-}d}}_{t}$, which is used to learn the structure of model $B_2^{'}$. Note that, in each iteration the number of cases in $\mathcal{DS}^{\mathrm{p}}_{\mathrm{t}}$ should be between 0 (i.e., no poisoned cases) and $\frac{\tau}{w}$, which is the maximum number of poisoned cases that could be added to $\mathcal{DS}^{\mathrm{c}}_{\mathrm{t}}$ in the t^{th} iteration.

We terminate after iteration w. If $\bigcup_{t=1}^{w} \lambda_t N_t \leq \beta N$, we return $\mathcal{DS}^{\mathrm{l\text{-}d}}_{t}$; otherwise, we print a failure message since implementing the long-duration attack on $\mathcal{DS}^{\mathrm{v}}$ is not feasible.

We assumed that $w = 4$, which means that the attacker is allowed to send in four contaminated datasets to achieve the long-duration data poisoning attack. We divided the 20,000 case dataset that was generated for one-step data poisoning attacks in section 5.1 into five datasets as follows: 12,000 cases are used as $\mathcal{DS}^{\mathrm{v}}$; and the rest is divided into four datasets of 2,000 cases each. We call

Table 1: Results of long-duration data poisoning attacks against $\mathcal{DS}^{\text{v}}$.

(a) Introducing the link $A \rightarrow E$ in the diverging connection $A \leftarrow T \rightarrow E$.

Time point $t = \{1, \ldots, w\}$	$t = 1$	$t = 2$	$t = 3$	$t = 4$
Number of clean cases at time point t ($\mathcal{DS}_{\text{t}}^{\text{Clean}}$)	2,000	2,000	2,000	2,000
Number of crafted cases at time point t ($\mathcal{DS}_{\text{t}}^{\text{Crafted}}$)	3	1	3	0
$\mathcal{DS}_{\text{t}}^{\text{c}} = \mathcal{DS}_{\text{t}}^{\text{Clean}} \cup \mathcal{DS}_{\text{t}}^{\text{Crafted}}$	2,003	2,004	2,007	2,007
$\mathcal{DS}_{t}^{\text{I-d}} = \mathcal{DS}^{\text{v}} \cup \bigcup_{t=1}^{w} \mathcal{DS}_{\text{t}}^{\text{c}}$	14,003	16,004	18,007	20,007
Model Change	No	No	No	Yes

(b) Breaking the v-structure $T \rightarrow E \leftarrow L$.

Time point $t = \{1, \ldots, w\}$	$t = 1$	$t = 2$	$t = 3$	$t = 4$
Number of clean cases at time point t ($\mathcal{DS}_{\text{t}}^{\text{Clean}}$)	2,000	2,000	2,000	2,000
Number of crafted cases at time point t ($\mathcal{DS}_{\text{t}}^{\text{Crafted}}$)	2	2	0	0
$\mathcal{DS}_{\text{t}}^{\text{c}} = \mathcal{DS}_{\text{t}}^{\text{Clean}} \cup \mathcal{DS}_{\text{t}}^{\text{Crafted}}$	2,002	2,002	2,000	2,000
$\mathcal{DS}_{t}^{\text{I-d}} = \mathcal{DS}^{\text{v}} \cup \bigcup_{t=1}^{w} \mathcal{DS}_{\text{t}}^{\text{c}}$	14,002	16,004	18,004	20,004
Model Change	No	No	No	Yes

(c) Add the most believable edge, $B \rightarrow L$, to the causal model B_1.

Time point $t = \{1, \ldots, w\}$	$t = 1$	$t = 2$	$t = 3$	$t = 4$
Number of clean cases at time point t ($\mathcal{DS}_{\text{t}}^{\text{Clean}}$)	2,000	2,000	2,000	2,000
Number of crafted cases at time point t ($\mathcal{DS}_{\text{t}}^{\text{Crafted}}$)	2	2	1	2
$\mathcal{DS}_{\text{t}}^{\text{c}} = \mathcal{DS}_{\text{t}}^{\text{Clean}} \cup \mathcal{DS}_{\text{t}}^{\text{Crafted}}$	2,002	2,002	2,001	2,002
$\mathcal{DS}_{t}^{\text{I-d}} = \mathcal{DS}^{\text{v}} \cup \bigcup_{t=1}^{w} \mathcal{DS}_{\text{t}}^{\text{c}}$	14,002	16,004	18,005	20,007
Model Change	No	No	No	Yes

(d) Adding the link $D \rightarrow S$ to the serial connection $D \rightarrow B \rightarrow S$.

Time point $t = \{1, \ldots, w\}$	$t = 1$	$t = 2$	$t = 3$	$t = 4$
Number of clean cases at time point t ($\mathcal{DS}_{\text{t}}^{\text{Clean}}$)	2,000	2,000	2,000	2,000
Number of crafted cases at time point t ($\mathcal{DS}_{\text{t}}^{\text{Crafted}}$)	20	20	23	4
$\mathcal{DS}_{\text{t}}^{\text{c}} = \mathcal{DS}_{\text{t}}^{\text{Clean}} \cup \mathcal{DS}_{\text{t}}^{\text{Crafted}}$	2,020	2,020	2,023	2,004
$\mathcal{DS}_{t}^{\text{I-d}} = \mathcal{DS}^{\text{v}} \cup \bigcup_{t=1}^{w} \mathcal{DS}_{\text{t}}^{\text{c}}$	14,020	16,040	18,063	20,067
Model Change	No	No	No	Yes

(e) Adding the link $S \rightarrow E$ to the serial connection $S \rightarrow L \rightarrow E$.

Time point $t = \{1, \ldots, w\}$	$t = 1$	$t = 2$	$t = 3$	$t = 4$
Number of clean cases at time point t ($\mathcal{DS}_{\text{t}}^{\text{Clean}}$)	2,000	2,000	2,000	2,000
Number of crafted cases at time point t ($\mathcal{DS}_{\text{t}}^{\text{Crafted}}$)	7	8	5	1
$\mathcal{DS}_{\text{t}}^{\text{c}} = \mathcal{DS}_{\text{t}}^{\text{Clean}} \cup \mathcal{DS}_{\text{t}}^{\text{Crafted}}$	2,007	2,008	2,005	2,001
$\mathcal{DS}_{t}^{\text{I-d}} = \mathcal{DS}^{\text{v}} \cup \bigcup_{t=1}^{w} \mathcal{DS}_{\text{t}}^{\text{c}}$	14,007	16,015	18,020	20,021
Model Change	No	No	No	Yes

(f) Removing the weakest link, $T \rightarrow A$, from the causal model B_1.

Time point $t = \{1, \ldots, w\}$	$t = 1$	$t = 2$	$t = 3$	$t = 4$
Number of clean cases at time point t ($\mathcal{DS}_{\text{t}}^{\text{Clean}}$)	2,000	2,000	2,000	2,000
Number of crafted cases at time point t ($\mathcal{DS}_{\text{t}}^{\text{Crafted}}$)	1	1	1	0
$\mathcal{DS}_{\text{t}}^{\text{c}} = \mathcal{DS}_{\text{t}}^{\text{Clean}} \cup \mathcal{DS}_{\text{t}}^{\text{Crafted}}$	2,001	2,001	2,001	2,000
$\mathcal{DS}_{t}^{\text{I-d}} = \mathcal{DS}^{\text{v}} \cup \bigcup_{t=1}^{w} \mathcal{DS}_{\text{t}}^{\text{c}}$	14,001	16,002	18,003	20,003
Model Change	No	No	No	Yes

these four datasets $\mathcal{DS}_1^{\text{Clean}}$, $\mathcal{DS}_2^{\text{Clean}}$, $\mathcal{DS}_3^{\text{Clean}}$, and $\mathcal{DS}_4^{\text{Clean}}$. Using the PC algorithm on dataset $\mathcal{DS}^{\text{v}}$ with 0.05 significance setting [26], the resulting validated structure, $B_1 = PC\_Algo(\mathcal{DS}^{\text{v}})$, is given in Figure 3, which is the starting point of this experiment.

We evaluated the effectiveness of long-duration data poisoning attacks against the validated dataset $\mathcal{DS}^{\text{v}}$ (i.e., against the validated model B_1). At every time point $t = \{1, \ldots, w\}$, the attacker injects a contaminated dataset $\mathcal{DS}_{\text{t}}^{\text{Crafted}}$ into $\mathcal{DS}_{\text{t}}^{\text{Clean}}$, resulting in the dataset $\mathcal{DS}_{\text{t}}^{\text{c}}$. This resulting dataset is then sent in as a new source of information. The defender receives $\mathcal{DS}_{\text{t}}^{\text{c}}$ and retrains the validated model, B_1, by creating the union of $\mathcal{DS}^{\text{v}}$ and the new incoming dataset $\mathcal{DS}_{\text{t}}^{\text{c}}$ and feeding them to the PC algorithm, resulting in the model $B_2^{'}$ (i.e., $B_2^{'} = PC\_Algo(\mathcal{DS}^{\text{v}} \cup \mathcal{DS}_{\text{t}}^{\text{c}})$).

The results of our experiments are presented in Table 1. In all of the scenarios, the attacker succeeded in achieving the desired modification. In our experiments, we assumed that $t = \{1, \ldots, 4\}$. For every one of the studied long-duration attacks on the dataset $\mathcal{DS}^{\text{v}}$ (Tables 1a, 1b, 1c, 1d, 1e, and 1f), the adversary had to send in the attack over 4 datasets. That is, at every time point t (for $t = 1, \ldots, 4$), the attacker had to create the union of $\mathcal{DS}_{\text{t}}^{\text{Clean}}$ and $\mathcal{DS}_{\text{t}}^{\text{Crafted}}$ resulting in $\mathcal{DS}_{\text{t}}^{\text{c}}$, which was going to be sent to the targeted machine learning system as a new source of information. The defender, on the other hand, retrained the machine learning model every time a new incoming dataset $\mathcal{DS}_{\text{t}}^{\text{c}}$ arrived.

Note that, in our experiments, long-duration attacks require the same number of contaminated cases as the one-step data poisoning attacks. An important observation is that the malicious attacker does not always have to send poisoned cases in the last dataset that will trigger the attack. For instance, in our experiments, when introducing the link $A \rightarrow E$ (Table 1a), shielding collider E (Table 1b), and removing the weakest edge (Table 1f), the last contaminated dataset, $\mathcal{DS}_4^{\text{c}}$, had no contaminated cases, which makes it impossible for the defender to find what caused a change in the newly learned model.

5.3 Discussion: Detecting Data Poisoning Attacks

The results of using our framework to detect one-step data poisoning attacks are presented in Table 2. Algorithm 1 succeeded to detect the negative impact (i.e., the change in the Markov equivalence class) of the new incoming dataset $\mathcal{DS}^{\text{p}}$ on the validated model B_1.

Table 2: Results of using FLoD to detect one-step poisoning attacks.

Attack	Attack's class	$\mathbf{ds}(B_1, B_2)$ score
Introduce the link $A \rightarrow E$	New v-structure	12
Introduce the link $D \rightarrow S$	New v-structure	24
Introduce the link $S \rightarrow E$	New v-structure	54
Introduce the link $T \rightarrow L$	Shield an existing collider	16
Remove the link $A \rightarrow T$	Delete the weakest link	4
Introduce the link $B \rightarrow L$	Add the most believable link	32

The results using our framework to detect long-duration data poisoning attacks are summarized in Table 3. Algorithm 2 succeeded to detect the long-duration impact of $\mathcal{DS}^{c}$ on the validated dataset $\mathcal{DS}^{v}$. Note, that FLoD using traditional reject on negative impact was not able to detect long-duration attacks. However, when using the SLoD, we were able to detect the conflicting cases, which are either inconsistent or incompatible with the original validated model B_1 (A detailed experiment is presented in Figure 4). Such cases might be exploited by a malicious adversary to trigger the long-duration attack at a later time. Also, in some attacks no poisoned cases are even required to be sent with $\mathcal{DS}^{c}$ to trigger the long-duration attack, which is very hard to detect.

Table 3: Results of using SLoD to detect long-duration data poisoning attacks.

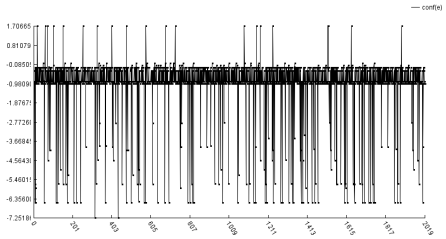
Attack	Attack's class	Algorithm 2 decision
Introduce $A \rightarrow E$	New v-structure	Inconsistent observations
Introduce $D \rightarrow S$	New v-structure	Incompatible observations
Introduce $S \rightarrow E$	New v-structure	Inconsistent observations
Introduce $T \rightarrow L$	Shield an existing collider	Inconsistent observations
Remove $A \rightarrow T$	Delete weakest link	Inconsistent\Incompatible observations
Introduce $B \rightarrow L$	Add most believable link	Inconsistent observations

In summary, our 2-layered approach was able to detect both one-step and long-duration attacks. Moreover, our solution did not lose all the incoming datasets; we only send conflicting cases to be checked offline. We have carried out over 200 experiments for long-duration attacks. A comprehensive description of these experiments is given in [2].

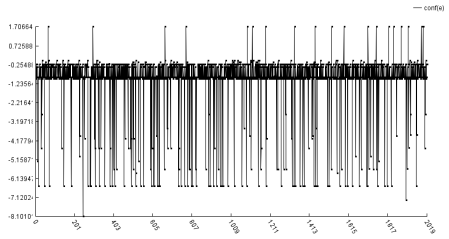
6 Related Work

In this section, we will give a brief overview of adversarial machine learning research; focusing on data poisoning.Recent surveys on adversarial machine learning can be found in [6, 16, 24].

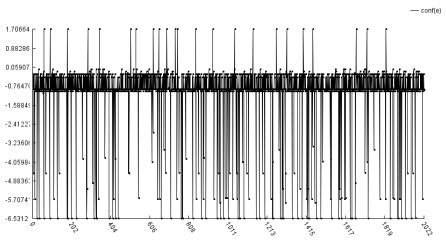
Data Poisoning Attacks: As machine learning algorithms have been widely used in security-critical settings such as spam filtering and intrusion detection, adversarial machine learning has become an emerging field of study. Attacks against machine learning systems have been organized by [6, 7, 18] according to three features: Influence, Security Violation, and Specificity. Influence of the attacks on machine learning models can be either causative or exploratory. Causative attacks aim to corrupt the training data whereas exploratory attacks aim to corrupt the classifier at test time. Security violation of machine learning models can be a violation of integrity, availability, or privacy. Specificity of the attacks can be either targeted or indiscriminate. Targeted attacks aim to corrupt machine learning models to misclassify a particular class of false positives whereas indiscriminate attacks have the goal of misclassifying all false positives.



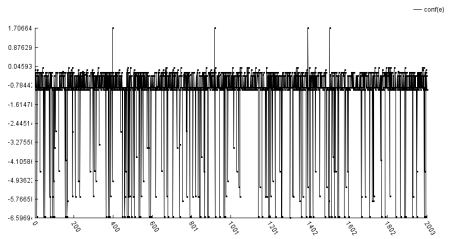
(a) $\mathcal{DS}_1^{\mathrm{c}}$ has 20 incompatible cases.



(b) $\mathcal{DS}_2^{\mathrm{c}}$ has 20 incompatible cases.



(c) $\mathcal{DS}_3^{\mathrm{c}}$ has 23 incompatible cases.



(d) $\mathcal{DS}_4^{\mathrm{c}}$ has 4 incompatible cases.

Fig. 4: The result of using **SLoD** to detect a long-duration attack that aims to introduce the link $D \rightarrow S$ in the Chest Clinic dataset, $\mathcal{DS}^{\mathrm{v}}$. We present the case number in $\mathcal{DS}_{\mathrm{t}}^{\mathrm{c}}$ as the variable on the X-axis and the value of our conflict measure $Conf(c, B_1)$ as the variable on the Y-axis. A case is incompatible (conflicting) with the validated model B_1 if $Conf(c, B_1) > 0$.

Evasion attacks and Data poisoning attacks are two of the most common attacks on machine learning systems [18]. Evasion attacks [17, 20, 22] are exploratory attacks at the testing phase. In an evasion attack, an adversary attempts to pollute the data for testing the machine learning classifier; thus causing the classifier to misclassify adversarial examples as legitimate ones. Data poisoning attacks [1, 11, 21, 27, 28, 36] are causative attacks, in which adversaries attempt to corrupt the machine learning classifier itself by contaminating the data in the training phase.

Data poisoning attacks are studied extensively during the last decade [3, 8, 9, 10, 11, 12, 21, 28, 29, 36]. However, attacks against Bayesian network algorithm are limited. In our previous work, we were addressed data poisoning attacks against Bayesian network algorithms [3, 4, 5]. We studied how an adversary could corrupt the Bayesian network structure learning algorithms by inserting contaminated data into the training phase. We showed how our novel measure of strengths of links for Bayesian networks [5] can be used to do a security analysis of attacks against Bayesian network structure learning algorithms. However, our approach did not consider long-duration attacks.

Defenses and Countermeasures: Detecting adversarial input is a challenging problem. Recent research [13, 15, 25] illustrate these challenges. Our work ad-

dresses these issues in the specific context of Bayesian network structure learning algorithms. Data sanitization is a best practice for security optimization in the adversarial machine learning context [14]. It is often impossible to validate every data source. In the event of a poisoning attack, data sanitization adds a layer of protection for training data by removing contaminated samples from the targeted training data set prior to training a classifier. Reject on Negative Impact is one of the widely used method for data sanitization [6,14,24]. Reject on Negative Impact defense assesses the impact of new training sample additions, opting to remover or discard samples that yield significant, negative effects on the observed learning outcomes or classification accuracy [6,14]. The base training set is used to train a classifier, after which, the new training instance is added and a second classifier is trained [6]. In this approach, classification performance is evaluated by comparing error rates (accuracy) between the original and the new, retrained classifier resulting from new sample integration [24]. As such, if new classification errors are substantially higher compared to the original or baseline classifier, it is assumed that the newly added samples are malicious or contaminated and are therefore removed in order to maximize and protect classification accuracy [6].

7 Conclusion and Future Work

Data integrity is vital for effective machine learning. In this paper, we studied data poisoning attacks against Bayesian network structure learning algorithms. We demonstrated the vulnerability of the PC algorithm against one-step and long-duration data poisoning attacks. We proposed a 2-layered framework for detecting data poisoning attacks. We implemented our prototype system using the Chest Clinic Network which is a widely used network in Bayesian networks. Our results indicate that Bayesian network structure learning algorithms are vulnerable to one-step and long-duration data poisoning attacks. Our framework is effective in detecting both one-step and long-duration data poisoning attacks, as it thoroughly validates and verifies training data before such data is being incorporated into the model.

Our ongoing work focuses on offline validation of potentially malicious datasets. Currently, our approach detects datasets that either change the Bayesian network structure (distance measure) or in conflict with the validated model (conflict measure). We are investigating methods for 1) distinguishing actual model shift from model enrichment, i.e., our initial model was based on data that was not fully representative of the "true" distribution, and 2) determining if cases are truly conflicting or again if the initial model poorly approximates the "true" distribution. We are also investigating the applicability of Wisdom of the Crowd (WoC) [37]. Rather than human experts, we plan to use an ensemble of classifiers, i.e., take the votes of competing algorithms instead of the votes of humans. In the case of an ensemble of classifiers, one could investigate the likelihood of unexpected cases and adjust the sensitivity to anomalies by how much perturbation it causes in the model.

References

1. Alfeld, S., Zhu, X., Barford, P.: Data poisoning attacks against autoregressive models. In: AAAI. pp. 1452–1458 (2016)
2. Alsuwat, E., Alsuwat, H., Rose, J., Valtorta, M., Farkas, C.: Long duration data poisoning attacks on Bayesian networks. Tech. rep., University of South Carolina, SC, USA (2019)
3. Alsuwat, E., Alsuwat, H., Valtorta, M., Farkas, C.: Cyber attacks against the pc learning algorithm. In: ECML PKDD 2018 Workshops. pp. 159–176. Springer International Publishing, Cham (2019)
4. Alsuwat, E., Valtorta, M., Farkas, C.: Bayesian structure learning attacks. Tech. rep., University of South Carolina, SC, USA (2018)
5. Alsuwat, E., Valtorta, M., Farkas, C.: How to generate the network you want with the pc learning algorithm. In: Proceedings of the 11th Workshop on Uncertainty Processing (WUPES'18). pp. 1 – 12 (2018)
6. Barreno, M., Nelson, B., Joseph, A.D., Tygar, J.D.: The security of machine learning. Machine Learning **81**(2), 121–148 (Nov 2010)
7. Barreno, M., Nelson, B., Sears, R., Joseph, A.D., Tygar, J.D.: Can machine learning be secure? In: Proceedings of the 2006 ACM Symposium on Information, computer and communications security. pp. 16–25. ACM (2006)
8. Biggio, B., Bulò, S.R., Pillai, I., Mura, M., Mequanint, E.Z., Pelillo, M., Roli, F.: Poisoning complete-linkage hierarchical clustering. In: Joint IAPR International Workshops on Statistical Techniques in Pattern Recognition (SPR) and Structural and Syntactic Pattern Recognition (SSPR). pp. 42–52. Springer (2014)
9. Biggio, B., Didaci, L., Fumera, G., Roli, F.: Poisoning attacks to compromise face templates. In: Biometrics (ICB), 2013 International Conference on. pp. 1–7. IEEE (2013)
10. Biggio, B., Fumera, G., Roli, F., Didaci, L.: Poisoning adaptive biometric systems. In: Joint IAPR International Workshops on Statistical Techniques in Pattern Recognition (SPR) and Structural and Syntactic Pattern Recognition (SSPR). pp. 417–425. Springer (2012)
11. Biggio, B., Nelson, B., Laskov, P.: Poisoning attacks against support vector machines. In: Proceedings of the 29th International Coference on International Conference on Machine Learning. pp. 1467–1474. Omnipress (2012)
12. Biggio, B., Pillai, I., Rota Bulò, S., Ariu, D., Pelillo, M., Roli, F.: Is data clustering in adversarial settings secure? In: Proceedings of the 2013 ACM workshop on Artificial intelligence and security. pp. 87–98. ACM (2013)
13. Carlini, N., Wagner, D.: Adversarial examples are not easily detected: Bypassing ten detection methods. In: Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security. pp. 3–14. ACM (2017)
14. Chan, P.P., He, Z.M., Li, H., Hsu, C.C.: Data sanitization against adversarial label contamination based on data complexity. International Journal of Machine Learning and Cybernetics **9**(6), 1039–1052 (2018)
15. Feinman, R., Curtin, R.R., Shintre, S., Gardner, A.B.: Detecting adversarial samples from artifacts. CoRR **abs/1703.00410** (2017)
16. Gardiner, J., Nagaraja, S.: On the security of machine learning in malware c&c detection: A survey. ACM Computing Surveys (CSUR) **49**(3), 59 (2016)
17. Goodfellow, I.J., Shlens, J., Szegedy, C.: Explaining and harnessing adversarial examples. arXiv preprint arXiv:1412.6572 (2014)

18. Huang, L., Joseph, A.D., Nelson, B., Rubinstein, B.I., Tygar, J.: Adversarial machine learning. In: Proceedings of the 4th ACM workshop on Security and artificial intelligence. pp. 43–58. ACM (2011)
19. de Jongh, M., Druzdzel, M.J.: A comparison of structural distance measures for causal bayesian network models. Recent Advances in Intelligent Information Systems, Challenging Problems of Science, Computer Science series pp. 443–456 (2009)
20. Kantchelian, A., Tygar, J., Joseph, A.: Evasion and hardening of tree ensemble classifiers. In: International Conference on Machine Learning. pp. 2387–2396 (2016)
21. Koh, P.W., Liang, P.: Understanding black-box predictions via influence functions. In: International Conference on Machine Learning. pp. 1885–1894 (2017)
22. Laskov, P., et al.: Practical evasion of a learning-based classifier: A case study. In: Security and Privacy (SP), 2014 IEEE Symposium on. pp. 197–211. IEEE (2014)
23. Lauritzen, S.L., Spiegelhalter, D.J.: Local computations with probabilities on graphical structures and their application to expert systems. Journal of the Royal Statistical Society. Series B (Methodological) pp. 157–224 (1988)
24. Liu, Q., Li, P., Zhao, W., Cai, W., Yu, S., Leung, V.C.: A survey on security threats and defensive techniques of machine learning: a data driven view. IEEE access **6**, 12103–12117 (2018)
25. Lu, J., Issaranon, T., Forsyth, D.: Safetynet: Detecting and rejecting adversarial examples robustly. In: 2017 IEEE International Conference on Computer Vision (ICCV). pp. 446–454 (Oct 2017). https://doi.org/10.1109/ICCV.2017.56
26. Madsen, A.L., Jensen, F., Kjaerulff, U.B., Lang, M.: The hugin tool for probabilistic graphical models. International Journal on Artificial Intelligence Tools **14**(03), 507–543 (2005)
27. Mei, S., Zhu, X.: The security of latent dirichlet allocation. In: Artificial Intelligence and Statistics. pp. 681–689 (2015)
28. Mei, S., Zhu, X.: Using machine teaching to identify optimal training-set attacks on machine learners. In: AAAI. pp. 2871–2877 (2015)
29. Muñoz-González, L., Biggio, B., Demontis, A., Paudice, A., Wongrassamee, V., Lupu, E.C., Roli, F.: Towards poisoning of deep learning algorithms with back-gradient optimization. In: Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security. pp. 27–38. ACM (2017)
30. Nelson, B., Barreno, M., Chi, F.J., Joseph, A.D., Rubinstein, B.I., Saini, U., Sutton, C., Tygar, J., Xia, K.: Misleading learners: Co-opting your spam filter. In: Machine learning in cyber trust, pp. 17–51. Springer (2009)
31. Nielsen, T.D., Jensen, F.V.: Bayesian networks and decision graphs. Springer Science & Business Media (2009)
32. Olesen, K.G., Lauritzen, S.L., Jensen, F.V.: ahugin: A system creating adaptive causal probabilistic networks. In: Uncertainty in Artificial Intelligence, 1992, pp. 223–229. Elsevier (1992)
33. Paudice, A., Muñoz-González, L., Gyorgy, A., Lupu, E.C.: Detection of adversarial training examples in poisoning attacks through anomaly detection. arXiv preprint arXiv:1802.03041 (2018)
34. Spirtes, P., Glymour, C.N., Scheines, R.: Causation, prediction, and search. MIT press (2000)
35. Wang, Y., Chaudhuri, K.: Data poisoning attacks against online learning. arXiv preprint arXiv:1808.08994 (2018)
36. Yang, C., Wu, Q., Li, H., Chen, Y.: Generative poisoning attack method against neural networks. arXiv preprint arXiv:1703.01340 (2017)
37. Yi, S.K.M., Steyvers, M., Lee, M.D., Dry, M.J.: The wisdom of the crowd in combinatorial problems. Cognitive science **36**(3), 452–470 (2012)

A Causative, Long-duration Model Invalidation Attacks

In this Appendix, we explain the two subtypes of each of the causative long-duration attacks which are based on the notion of d-separation and marginal independence tests.

The causative long-duration attacks which are based on the notion of d-separation are divided into two main subtypes as follows:

(i) Creating a new converging connection (v-structure) attacks, in which adversaries attempt to corrupt the original Bayesian network model, B_1, by poisoning the validated dataset, $\mathcal{DS}^{\mathrm{v}}$, using contaminated datasets $\bigcup_{t=1}^{w} \mathcal{DS}_{\mathrm{t}}^{\mathrm{c}}$. Attackers aim to introduce a new v-structure by adding the link $A \rightarrow C$ to the serial connection $A \rightarrow B \rightarrow C$, link $C \rightarrow A$ to the serial connection $A \leftarrow B \leftarrow C$, or either one of the links $A \rightarrow C$ or $C \rightarrow A$ to the diverging connection $A \leftarrow B \rightarrow C$ in B_1.

(ii) Breaking an existing converging connection (v-structure) attacks, in which malicious attackers attempt to corrupt the original model, B_1, by shielding existing colliders (v-structures). Such adversarial attacks can be performed by poisoning the dataset, $\mathcal{DS}^{\mathrm{v}}$, over time using the poisoned datasets $\bigcup_{t=1}^{w} \mathcal{DS}_{\mathrm{t}}^{\mathrm{c}}$ such that new links are introduced to marry the parents of unshielded colliders in B_1 (i.e., add the link $A \rightarrow C$ to the converging connection $A \rightarrow B \leftarrow C$).

We divide the causative long-duration attacks which are based on marginal independence tests into two main subtypes:

(i) Removing the weakest edge attacks, in which adversarial opponents attempt to poison the validated learning dataset, $\mathcal{DS}^{\mathrm{v}}$, using contaminated datasets, $\bigcup_{t=1}^{w} \mathcal{DS}_{\mathrm{t}}^{\mathrm{c}}$, over a period of time t with the ultimate goal of removing weak edges. Note that, a weak edge in a Bayesian model, B_1, is the easiest edge to be removed from B_1. We use our previously defined link strength measure to determine such edges [5].

(ii) Adding the most believable yet incorrect edge attacks, in which adversaries can cleverly craft their input datasets, $\bigcup_{t=1}^{w} \mathcal{DS}_{\mathrm{t}}^{\mathrm{c}}$, over a period of time t to poison $\mathcal{DS}^{\mathrm{v}}$ so that adding the most believable yet incorrect edge is viable. The most believable yet incorrect edge is a newly added edge to model, B_1, with the maximum amount of belief. We use our link strength measure defined in [5] to determine such edges.