Poisoning Network Flow Classifiers

Giorgio Severi*, Simona Boboila*, Alina Oprea*,
John Holodnak§, Kendra Kratkiewicz§, Jason Matterer§
Northeastern University* MIT Lincoln Laboratory§

ABSTRACT

As machine learning (ML) classifiers increasingly oversee the automated monitoring of network traffic, studying their resilience against adversarial attacks becomes critical. This paper focuses on poisoning attacks, specifically backdoor attacks, against network traffic flow classifiers. We investigate the challenging scenario of clean-label poisoning where the adversary's capabilities are constrained to tampering only with the training data — without the ability to arbitrarily modify the training labels or any other component of the training process. We describe a trigger crafting strategy that leverages model interpretability techniques to generate trigger patterns that are effective even at very low poisoning rates. Finally, we design novel strategies to generate stealthy triggers, including an approach based on generative Bayesian network models, with the goal of minimizing the conspicuousness of the trigger, and thus making detection of an ongoing poisoning campaign more challenging. Our findings provide significant insights into the feasibility of poisoning attacks on network traffic classifiers used in multiple scenarios, including detecting malicious communication and application classification.

1 INTRODUCTION

Automated monitoring of network traffic plays a critical role in the security posture of many companies and institutions. The large volumes of data involved, and the necessity for rapid decision-making, have led to solutions that increasingly rely on machine learning (ML) classifiers to provide timely warnings of potentially malicious behaviors on the network. Given the relevance of this task, undiminished despite being studied for quite a long time [53], a number of machine learning based systems have been proposed in recent years [28, 51, 59, 60, 87] to classify network traffic.

The same conditions that spurred the development of new automated network traffic analysis systems, have also led researchers to develop adversarial machine learning attacks against them, targeting both deployed models [5, 9, 13, 24, 63] (*evasion* attacks) and, albeit to a lesser extent, their training process [4, 29, 40, 57] (*poisoning* attacks). We believe this second category is particularly interesting, both from an academic perspective as well as a practical one. Recent research on perceived security risks of companies deploying machine learning models repeatedly highlighted poisoning attacks as a critical threat to operational ML systems [22, 77]. Yet, much of the prior research on poisoning attacks in this domain tends to adopt threat models primarily formulated in the sphere of image classification, such as assuming that the victim would accept a pre-trained model from a third party [57], thus allowing adversarial control over the entire training phase, or granting the adversary the ability to tamper with the training labels [4]. As awareness of poisoning attacks permeates more extensively, it is reasonable to assume that companies developing this type of systems will exhibit an increased wariness to trust third parties providing pre-trained classifiers, and will likely spend resources and effort to control or vet both code and infrastructure used during training. For this reason, we believe it is particularly interesting to focus on the less studied scenario of an adversary who is restricted to tampering only with the training data (*data-only* attack) by disseminating a small quantity of maliciously crafted points, and without the ability to modify the labels assigned to training data (*clean-label*) or any other component of the training process.

Our aim is to investigate the feasibility and effects of poisoning attacks, and in particular backdoor attacks —where an association is induced between a trigger pattern and an adversarially chosen output of the model—, on network traffic flow classifiers. Our approach focuses on the manipulation of aggregated traffic flow features rather than packet-level content, as they are common in traffic classification applications [52, 60, 87]. We will focus on systems that compute aggregated features starting from the outputs of the network monitoring tool Zeek[1], because of its large user base. It is important to note that, despite the perceived relevance of poisoning attacks, it is often remarkably difficult for an adversary to successfully run a poisoning campaign against classifiers operating on constraint-heavy tabular data, such as cybersecurity data — like network flows or malware samples [72]. This is a well known issue in adversarial ML, illustrated in detail by [67] and often referred to as *problem-space* mapping. It stems from the complexity of crafting perturbations of the data points (in feature space) that induce the desired behavior in the victim model without damaging the structure of the underlying data object (problem space) necessary for it to be generated, parsed, or executed correctly. When dealing with aggregated network flow data, these difficulties compound with the inherent complexity of handling multivariate tabular data consisting of heterogeneous fields. To address these challenges, we design a novel methodology based on ML explanation methods to determine important features for backdoor creation, and map them back into the problem space. Our methods handle complex dependencies in feature space, generalize to different models and feature representations, are effective at low poisoning rates (as low as 0.1%), and generate stealthy poisoning attacks.

In summary, we make the following contributions: (i) We develop a new strategy to craft clean-label, data-only, backdoor poisoning attacks against network traffic classifiers that are effective at low poisoning rates. (ii) We show that our poisoning attacks work across different model types, classification tasks, and feature representations, and we comprehensively evaluate the techniques on several network traffic datasets used for malware detection and application classification. (iii) We propose different strategies, including generative approaches based on Bayesian networks, to make the attacks inconspicuous and blend the poisoned data with the underlying

[1] https://zeek.org/ Previously known as Bro.

training set. To ensure reproducibility, we evaluate our techniques on publicly available datasets, and all the code used to run the experiments in the paper will be released upon publication.

2 BACKGROUND AND RELATED WORK

Machine Learning for Threat Detection. Machine learning methods have been successfully used to detect several cyber security threats, including: malicious domains [2, 3, 58, 61, 68], command-and-control communication between attackers and compromised hosts [54, 61], or malicious binaries used by adversaries for distributing exploit code and botnet commands [32, 78]. Several endpoint protection products [30, 31, 49, 50] are now integrating ML tools to proactively detect the rapidly increasing number of threats.

Adversarial Machine Learning. We can identify two major categories of integrity attacks against ML classifiers: (1) evasion attacks, which occur at test time and consist in applying an imperceptible perturbation to test samples in order to have them misclassified, and (2) poisoning attacks, which influence the training process (either through tampering with the training dataset or by modifying other components of the training procedure) to induce wrong predictions during inference. For details on other adversarial ML techniques, we direct the reader to the standardized taxonomy presented in [62].

In this study, we are focusing on backdoor poisoning attacks, a particularly insidious technique in which the attacker forces the learner to associate a specific pattern to a desired target objective — usually the benign class in cybersecurity applications. While backdoor poisoning does not impact the model's performance on typical test data, it leads to misclassification of test samples that present the adversarial pattern. Backdoor poisoning attacks against modern ML models were introduced by Gu et al. [23] in BadNets, where a small patch of bright pixels (the trigger pattern) was added to a subset of images at training time together with an altered label, to induce the prediction of a target class. Subsequently, Turner et al. [81] and Shafahi et al. [73] devised clean-label backdoor attacks which require more poisoning data samples to be effective, but relax some strong assumptions of previous threat models, making them significantly more applicable in security scenarios.

In cybersecurity, the earliest poisoning attacks were designed against worm signature generation [56, 65] and spam detectors [55]. More recently, a few studies have looked at packet-level poisoning via padding [29, 57], feature-space poisoning in intrusion detection [4, 41], and label flipping attacks for IoT [64]. Severi et al. [72] proposed to use model interpretation techniques to generate clean-label poisoning attacks against malware classifiers. Their strategies are applicable to security datasets whose records are independent such as individual files or Android applications, which present a direct mapping from feature space to problem space. In contrast, our study explores attacks trained on network traffic, where multiple sequential connections are translated into one single feature-space data point; in this setting, inverting triggers from feature to problem space becomes particularly difficult due to data dependencies.

Model Interpretation Techniques. With the proliferation and increase in complexity of ML models the field of explainable machine learning, focused on understanding and interpreting their predictions, has seen a substantial increase in popularity over recent years. We are particularly interested in model-agnostic interpretability techniques, which can be applied to any model. Linardatos et al. [43] provide a comprehensive taxonomy of these methods, and conclude that, among the black-box techniques presented, Shapley Additive explanations (SHAP) [47, 48] is the most complete, providing explanations for any model and data type both at a global and local scale. SHAP is a game-theory inspired method, which attempts to quantify how important each feature is for a classifier's predictions. SHAP improves on other model interpretation techniques like LIME [71], DeepLIFT [76] and Layer-Wise Relevance Propagation [6], by introducing a unified measure of feature importance that is able to differentiate better among output classes.

In this study, we also experiment with Gini index [21] and information gain [37, 39] – two of the most popular splitting algorithms in decision trees. A decision tree is built recursively, by choosing at each step the feature that provides the best split. Thus, the tree offers a natural interpretability, and a straightforward way to compute the importance of each feature towards the model's predictions.

Preserving Domain Constraints. Functionality-preserving attacks on network traffic have mostly looked at evasion during test time, rather than poisoning. For instance, Wu et al. [83] proposed a packet-level evasion attack against botnet detection, using reinforcement learning to guide updates to adversarial samples in a way that maintains the original functionality. Sheatsley et al. [75] study the challenges associated with the generation of valid adversarial examples that abide domain constraints and develop techniques to learn these constraints from data. Chernikova et al. [13] design evasion attacks against neural networks in constrained environments, using an iterative optimization method based on gradient descent to ensure valid numerical domain values. With our constraint-aware problem-space mapping, which also takes into account dependencies in network traffic, we delve one step further into the challenging issue of designing functionality-preserving attacks.

Significant advances have been made recently with respect to generating multivariate data. Modern tabular data synthesizers of mixed data types leverage the power of generative adversarial networks [11, 18, 19, 85, 90] and diffusion models [38] to create realistic content from the same distribution as the original data. Among the different frameworks, FakeTables [11] is the only attempt at preserving functional dependencies in relational tables. However, its evaluation is limited to Census and Air Carrier Statistics datasets, and its ability to capture more complex relationships between variables is unclear.

In this work, we model conditional dependencies in the traffic using Bayesian networks – a common choice for generating synthetic relational tables [15, 26, 35, 69, 89]. Bayesian networks offer increased transparency and computational efficiency over more complex generative models like generative adversarial networks [35]. We believe this is an important advantage is our setting, which deals with large volumes of network traffic featuring multiple variables (e.g., log fields). In cybersecurity, Bayesian networks have also been used to learn traffic patterns and flag potentially malicious attempts in intrusion detection systems [16, 33, 82, 84].

3 THREAT MODEL

Adversary's Capabilities. Recent work analysing the training time robustness of malware classifiers [72, 88] pointed out that the use of ever larger quantities of data to train effective security classifiers inherently opens up the doors to data-only poisoning attacks, especially in their more stealthy clean-label [73, 80] variants where the adversary does not control the label of the poisoned samples. Thus, in this work, we constrain the adversary to clean-label data-only attacks. This type of setup moves beyond the classic threat model proposed by Gu et al. [23] and adopted by other research [12, 46, 57], where the adversary was able to tamper with not only the content of the training points but also the corresponding ground-truth labels. Here, instead, by disseminating innocuous looking —but adversarially crafted— data, the adversary is able to indirectly tamper with a small, yet effective, percentage of the training set and induce the desired behavior in the learned model. To design the trigger, the adversary requires access to a small amount of clean labeled data, D_a, from a similar distribution as the victim's training data. In our experiments, we partition the test set in two disjoint sets of 85% and 15% of the points respectively, and supply the adversary with the smaller one.

We consider an adversary who has query-only access to the machine learning classifier. This allows the attacker to use the SHAP explanation technique to compute feature importance coefficients, but it prevents any form of inspection of model weights or hidden states. This scenario is very common for deployed models, as they often undergo periodical re-training but are only accessible behind controlled APIs. Interacting with a victim system, however, always imposes a cost on the attacker, whether in terms of actual monetary expenses for API quotas, or by increasing the risk of being discovered. Motivated by this observation, we also explore the use of model interpretation methods that do not require any access to the classifier, but instead leverage proxy models on local data (i.e., information gain and Gini coefficients), and can be used even when the model is not subject to re-training cycles. Several previous studies on training time attacks [46, 57] relax the model access constraints, assuming an adversary can train a ML classifier and provide it to the victim through third-party platforms such as Machine Learning as a Service (MLaaS) [70]. However, we believe that this threat model is rapidly becoming obsolete, at least in the cybersecurity domain, due to the push for stricter cyber hygiene practices from security vendors, including the reluctance to trust third-party model providers and MLaaS platforms [1, 66].

Adversary's Objective. The main objective of the adversary is to acquire the ability to consistently trigger desired behavior, or output, from the victim model, after the latter has been trained on the poisoned data. In this study, we focus on the binary class scenario (0/1), where the goal is reified into having points of a chosen *victim* class being mis-labeled as belonging to the *target* class, when carrying a backdoor pattern that does not violate the constraints of the data domain. For instance, in the benign/malicious case, the adversary attempts to have malicious data points misclassified as benign, where "benign" represents the target class.

Adversary's Target. We select two representative ML classifier models as target for our attacks: Gradient Boosting decision trees, and Feed-forward Neural Networks. Both of these models have been widely-used in intrusion detection for classifying malicious network traffic, with decision trees often preferred in security contexts due to their easier interpretation [34]. We study two use cases of network traffic classifiers: (1) detection of malicious activities, and (2) application classification.

Table 1: Network data format. Our data is represented by connection logs ("conn.log" files) extracted with the Zeek monitoring tool from publicly-available packet-level PCAP files.

Name	Description
orig_ip, resp_ip	Source and destination IP address
orig_p, resp_p	Source and destination port
proto	Transport Protocol (e.g., TCP, UDP, or ICMP)
service	Application protocol (e.g., ssh, dns, etc.)
ts	Timestamp – the connection start time
duration	Duration of connection
orig_pkts, resp_pkts	Number of transmitted packets
orig_bytes, resp_bytes	Number of payload bytes
conn_state	Connection state, assessing whether the connection was established and terminated normally (13 different states)

Data Format. In our threat model, network traffic consists of connection logs ("conn.log" files), which are extracted from packet-level PCAP files using the Zeek monitoring tool. The Zeek log fields used in our study are described in Table 1, and include port, IP address, protocol, service, timestamp, duration, packets, payload bytes and connection state. Thus, the input data is tabular and multivariate, consisting of multiple log fields in either numeric format (e.g., bytes, packets, etc.) or categorical format (e.g., connection state, protocol, etc.). A data point in this domain is represented by a *sequence* of raw log records grouped together. This *problem-space* data point is mapped into a corresponding *feature-space* data point through various aggregation techniques applied over the log field values.

Feature Representation. We study two standard and widely adopted feature mapping techniques: (1) aggregation, to produce statistical features, and (2) embeddings— using auto-encoders to automatically generate feature vectors. Traffic statistics have multiple applications in network monitoring and security [52, 87], which require dealing with large volumes of data. For instance, distinct count metrics are used to identify scanning attacks, while volume metrics or traffic distributions over port numbers and IP address ranges are utilized in anomaly detection [8]. We use similar aggregation methods with previous works [8, 60], to derive statistics of connections. The statistical features used in our study are described in Table 2, and include traffic volume by internal IP (in bytes and packets) within a 30-sec time window, connection counts by transport protocol, connection counts by state, etc.

Recent literature also features a variety of approaches for network traffic classification based on auto-encoders [14, 25, 51, 87]. Auto-encoders are unsupervised models that learn to reconstruct the training data. They are often used either for anomaly detection or to learn high level features to use in downstream classifiers.

Table 2: Statistical features aggregated over connection logs within each data point grouping. The grouping is comprised of connections within 30-sec time windows, aggregated separately for each *internal* IP and destination port within the time window. Note that the internal IP versus external IP distinction pertains to the subnet, not to the two ends of the connection (source/destination).

Field	Description
Aggregation Key: (time window, internal IP, destination port)	
proto	Count of connections per transport protocol
conn_state	Count of connections for each conn_state
orig_pkts, resp_pkts	Sum, min, max over packets
orig_bytes, resp_bytes	Sum, min, max over bytes
duration	Sum, min, max over duration
Aggregation Key: (time window, internal IP)	
ip	Count of distinct *external* IPs
resp_p	Count of distinct destination ports

4 ATTACK STRATEGY

The formulation of an appropriate trigger pattern is a fundamental aspect of backdoor poisoning attacks The inherent intricacies of network traffic —feature dependencies, multiple data modalities— makes it particularly challenging to ensure that the trigger is mapped correctly to realizable actions in problem space [67]. This is a stark difference with the image domain, where the backdoor trigger can be extremely simplistic, such as a bright colored square [23].

There are three key requirements that characterize a feasible poisoning attack: (i) To be effective, the trigger should be easy to associate to the target class by the victim model. (ii) The injected pattern should appear inconspicuous, so as to avoid detection by potential human or automated observers. (iii) The perturbations induced by the injection of the trigger pattern should not affect data validity. While the first two requirements are generic to any backdoor attack, the third one translates to additional constraints on adversarial actions in the network domain, specifically: (i) The adversary can only insert traffic, but not modify or remove existing traffic. (ii) Data semantics and dependencies need to be preserved, such as value restrictions on specific fields (e.g., upper/lower bounds on packet length), feature correlations (e.g., protocols use specific ports), etc. (iii) The injected pattern needs to handle multiple data types, i.e., numeric and categorical.

4.1 Crafting the Poisoning Data

To address the above challenges, we design a novel methodology that leverages insights from explanation-based methods to determine important features in feature space, then map them back to constraint-aware triggers in problem space. The mapping can be done via: (i) poisoning attacks using connections directly extracted from malicious traffic; (ii) poisoning attacks with reduced footprint; (iii) generative Bayesian models to increase attack stealthiness. Our attack strategy, illustrated in Figure 1, consists of five main phases:

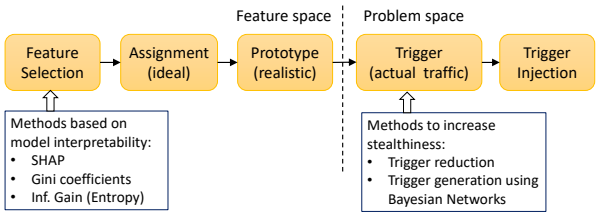


Figure 1: Pipeline for poisoning network flow classifiers.

(I) Select a subset of features that are most important for the class that the adversary wishes to misclassify using model explanation techniques;
(II) Find an ideal trigger in feature space — we call this an *assignment*;
(III) Find a data point that best approximate the ideal trigger values — this will be our *prototype* trigger;
(IV) Identify a set of real connections that induce the values observed in the prototype — this set of connections will be our actual *trigger*;
(V) Inject the trigger in points of the target class, potentially trying to minimize its conspicuousness.

Phase I. We first identify the most relevant features for the class to be misclassified. Our goal is to leverage highly informative features to coerce the model into associating the trigger pattern with the target class. There are a variety of techniques from the field of model interpretability used to estimate the effect of specific features towards the classifier's decision. We start by adapting the SHAP-based technique from [72] to the network domain. Here, SHAP values are computed for a subset of points to which the adversary has access to, and their contributions summed per-feature, to identify the ones most contributing to each class. This approach has the advantage of being model agnostic, allowing us to estimate feature importance coefficients for any possible victim model. Unfortunately, it also assumes the adversary is able to perform a possibly large number of queries against the victim model. To address this potential limitation, we also evaluate the effect of selecting the important features through more indirect ways. In particular we can leverage the *information gain* and *Gini coefficient* metrics used in training decision trees, to estimate the global contributions of each feature. The attentive reader will notice here that the approaches we mentioned to estimate feature importance are quite different. This is intentional, and it highlights the modularity of this component. As long as the adversary is capable of obtaining global estimates of feature importance scores, they can use them to guide the attack. Moreover, with potential future discoveries in the, extremely active, field of model interpretation, novel methods could be used to improve the effectiveness of this attack.

Phase II. Once the subset of important features is selected, we can proceed to find a suitable *assignment* of values. To be consistent with real traffic constraints, we need to ensure that the values that we select represent information that can be easily added to data points of the non-target class, by injecting new connections, without having to remove existing connections. Thus, we select values

that correspond to the top t^{th} percentile of the corresponding features for non-target class points; in practice, setting this parameter to 95^{th} percentile performed well in our experiments. Note that the non-target class points are generated by software under the control of the adversary, and therefore we assume they have access to a collection of log rows that represent those connections.

Phase III. Armed with the desired *assignment* for the selected features, we can proceed to identify an existing data point that approximates these ideal trigger values. To find it, in our first attack we leverage a *mimicry* method to scan the non-target (e.g., malicious) class samples and isolate the one with the lowest Euclidean distance from the assignment, in the subspace of the selected features. We call this point in feature space the trigger *prototype*.

Phase IV. Up until this point, the process was working completely in feature space. Our explicit goal, however, is to run the attack in problem space. So the next step in the attack chain is to identify, in the attacker's dataset, a contiguous subset of log connections that best approximate the *prototype*. Enforcing that the selected subset is contiguous ensures that temporal dependencies across log records are preserved. This subset of connections represents the actual *trigger* that we will use to poison the target-class training data.

Phase V. Finally, it is time to inject the trigger in the training data. This step is quite straightforward, as it only requires the software under control of the adversary, to execute the *trigger* connections in the specified order. We next describe two strategies for increasing trigger stealthiness before injection.

4.2 Increasing Attack Stealthiness

Beyond the basic objective of maximizing attack success, the adversary may have the additional goal of minimizing the chance of being detected. To achieve this secondary goal, the adversary may wish to slightly alter the trigger before injecting it in the training data. In particular, we study two strategies: (1) trigger size reduction and (2) trigger generation using Bayesian models.

Trigger size reduction. The first strategy consists of minimizing the trigger footprint, by removing all the connections that are not strictly necessary to achieve the values specified in the *prototype* for the subset of important features (such as connections on other ports). We then select the smallest subset of contiguous connections that would produce the desired values for the selected features.

Trigger generation using Bayesian networks. The second strategy aims at reducing the conspicuousness of the trigger by blending it with the set of connections underlying the data point where it is embedded. To this end, we generate the values of the log fields corresponding to *non-selected* features in the backdoor to make them appear closer to values common in the target-class natural data $\in D_a$. Note that fields influencing the selected (important) features will *not* be modified, because they carry the backdoor pattern associated with the target class. Our generative approach leverages Bayesian networks, a widely-used probabilistic graphical model for encoding conditional dependencies among a set of variables, and deriving realistic samples of data [15, 26, 69]. Bayesian networks consist of two parts: (1) structure – a directed acyclic graph

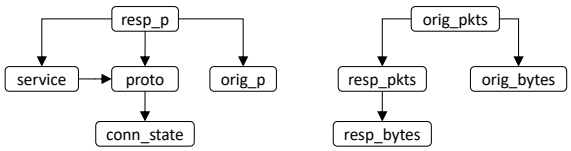


Figure 2: Directed Acyclic Graph (DAG) representing the inter-dependencies between log connection fields.

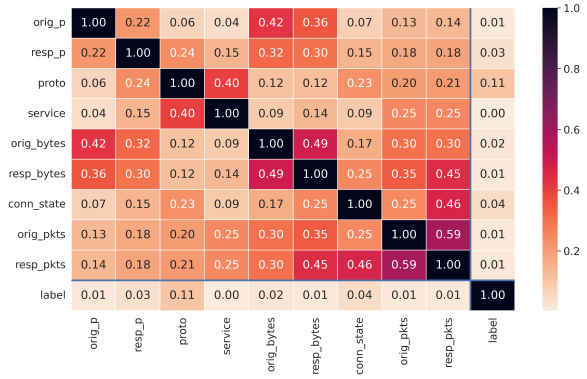
(DAG) that expresses dependencies among the random variables associated with the nodes, and (2) parameters – represented by conditional probability distributions associated with each node.

Structure. Given our objective to synthesize realistic log connections (in problem space) that lead to the feature-space prototype, we construct a directed acyclic graph $G = (V, E)$ where the nodes $x_i \in V$ correspond to fields of interest in the connection log and the edges $e_{ij} \in E$ model the inter-dependencies between them. We explore field-level correlations in connection logs using two statistical methods that have been previously used to study the degree of association between variables [36]: the correlation matrix and the pairwise normalized mutual information. In our experiments, both methods discover similar relationships in D_a, with the mutual information approach bringing out additional inter-dependencies. Note that we are not interested in the actual coefficients, rather, in the associational relationships between variables. Thus, we extract the strongest pairwise associations, and use them in addition to domain expertise to guide the design of the DAG structure. For instance, there is a strong relationship between the number of response packets and source packets (resp_pkts ↔ orig_pkts); between the protocol and the response port (proto ↔ resp_p); between the connection state and protocol (conn_state ↔ proto), etc.

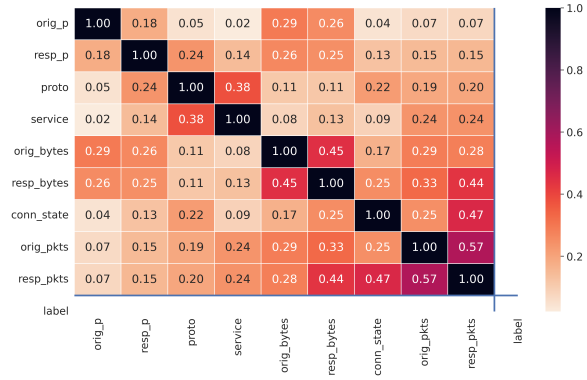
There is a large body of literature on learning the DAG structure directly from data. We point the interested reader to a recent survey by Kitson et al. [36]. However, computing the graphical structure remains a major challenge, as this is an NP-hard problem, where the solution space grows super-exponentially with the number of variables. Resorting to a hybrid approach [36] that incorporates expert knowledge is a common practice that alleviates this issue. The survey also highlights the additional complexity in modeling the DAG when continuous variables are parents of discrete ones, and when there are more than two dependency levels in the graph.

Based on the above considerations, we design the direct acyclic graph presented in Figure 2. For practical reasons, we filter out some associations that incur a high complexity when modeling the conditional probability distributions. To ensure that the generated traffic still reflects the inter-dependency patterns seen in the data, we inspect the poisoned training dataset using the same statistical techniques (correlation matrix and mutual information). We include the mutual information matrix on the clean adversarial dataset (Figure 3a) and on the training dataset poisoned with the Generated trigger method (Figure 3b), to show that the associational relationships between variables are preserved after poisoning (though the actual coefficients may vary).

Parameters. Bayesian networks follow the local Markov property, where the probability distribution of each node, modeled as a



(a) Mutual information on *clean data*, computed on the adversary's dataset.



(b) Mutual information on the *poisoned training dataset*

Figure 3: Mutual information comparison on clean and poisoned data. Showing associations between relevant fields of the *conn.log* file for CTU-13.

random variable x_i, depends only on the probability distributions of its parents. Thus, the joint probability distribution of a Bayesian network consisting of n nodes is represented as: $p(x_1, x_2, \cdots, x_n) = \prod_{i=1}^{n} p(x_i|x_{P_i})$, where P_i is the set of parents for node i, and the conditional probability of node i is expressed as $p(x_i|x_{P_i})$.

Sampling. The DAG is traversed in a hierarchical manner, one step at a time, as a sequential decision problem based on probabilities derived from the data, with the goal of generating a realistic set of field-value assignments. The value assignments for nodes at the top of the hierarchy are sampled independently, from the corresponding probability distribution, while the nodes on lower levels are conditioned on parent values during sampling. We compute the conditional probabilities of categorical fields (e.g., ports, service, protocol, connection state), and model numerical fields (e.g., originator/responder packets and bytes) through Gaussian kernel density estimation (KDE). An example of the KDE learned from the data, and used to estimate the number of exchanged bytes between a source (originator) and a destination (responder), given the number of packets, is presented in Figure 4.

Given the complexity of sampling from hybrid Bayesian networks, we approximate the conditional sampling process with a

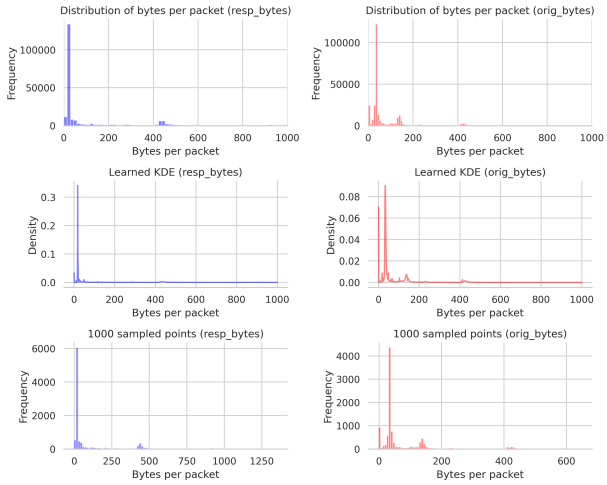


Figure 4: Modeling the bytes distribution for responder (left side) and originator (right side): From top to bottom, the figures show: distribution of byte counts per packet, learned KDEs, and sampled data from the learned distributions.

heuristic, described in Table 3. We consider an example where the log fields corresponding to the most important features have been set to the TCP protocol and responder port 80. Our generative method synthesizes values for the rest of the fields, in an attempt to make the trigger blend in with the target class. We show in our evaluation that the synthesized poisoning traffic is a good approximation of clean network traffic, both in terms of Jensen-Shannon distance between distributions (Section 5.3) and preservation of field-level dependencies.

5 EXPERIMENTAL RESULTS

5.1 Experimental Setup

In this section, we describe the datasets and performance metrics used in our evaluation. We also present the baseline performance of the target classifiers (without poisoning).

Datasets. We used three public datasets commonly used in cybersecurity research for intrusion detection and application classification.

CTU-13 Neris Botnet: We started our experimentation with the Neris botnet scenario of the well-known CTU-13 dataset [20]. This dataset offers a window into the world of botnet traffic, captured within a university network and featuring a blend of both malicious and benign traffic. Despite the sizeable number of connections ($\approx 9{*}10^6$), the classes are extremely imbalanced, with a significantly larger number of benign than malicious data points. Note that the class imbalance is a common characteristic of security applications. The Neris botnet scenario unfolds over three capture periods. We use two of these periods for training our models, and we partition the last one in two subsets, keeping 85% of the connections for the test set, and 15% for the adversarial set, D_a.

Table 3: Sampling method for each dependency described in the DAG from Figure 2. In this example, we assume that the most important features correspond to protocol and port; their values (TCP protocol on port 80) have been determined in Phase II of our strategy. Here, our generative method samples the rest of the log field values. D_a represents the attacker's dataset.

Dependency	Sampling method
1. resp_p → service	Select subset from attacker's data, D_a, with resp_p = 80. Sample a value for service (S) according to the observed probabilities.
2. service → conn_state	Subset D_a with proto = TCP and service = S. Sample conn_state according to the observed probabilities.
3. resp_p → orig_p	Subset D_a with resp_p = 80. Sample orig_p according to the observed probabilities.
4. orig_pkts	Sample a value for orig_pkts from the KDE learned on D_a.
5. orig_pkts → resp_pkts	Subset D_a based on orig_pkts. Learn the KDE for resp_pkts from the subset. Sample resp_pkts from the KDE.
6. orig_pkts → orig_bytes	Learn the KDE distribution D_O of originator bytes-per-packet from D_a. Given previously sampled value for number of packets, orig_pkts = m, sample and sum up $1, \cdots, m$ values from the distribution D_O.
7. resp_pkts → resp_bytes	Learn the KDE distribution D_R of responder bytes-per-packet from D_a. Given previously sampled value for number of packets, resp_pkts = n, sample and sum up $1, \cdots, n$ values from the distribution D_R.

CIC IDS 2018 Botnet: From CTU-13, we moved to a recent dataset for intrusion detection systcheems, the Canadian Institute for Cybersecurity (CIC) IDS 2018 dataset [74]. We experimented with the botnet scenario, in which the adversary uses the Zeus and Ares malware packages to infect victim machines and perform exfiltration actions. This dataset includes a mixture of malicious and benign samples and is also heavily imbalanced.

CIC ISCX 2016 dataset: This dataset contains several application traffic categories, such as chat, video, file transfer. We leverage the CIC ISCX 2016 dataset [17] to explore another scenario where an adversary may affect the outcome via poisoning: detection of banned applications. For instance, to comply with company policies, an organization monitors its internal network to identify usage of prohibited applications. An adversary may attempt to disguise traffic originating from a banned application as another type of traffic. We study two examples of classification tasks on the non-vpn traffic of this dataset: (1) *File vs Video*, where we induce the learner to mistake video traffic flows as file transfer, and (2) *Chat vs Video*, where the classifier mis-labels video traffic as chat communication.

Performance Metrics. Similar to previous work in this area [57, 72], we are interested in the following indicators of performance for the backdoored model:

- *Attack Success Rate (ASR).* This is the fraction of test data points which are mis-classified as belonging to the target class. We evaluate this metric on a subset of points that have been *previously correctly classified* by a clean model trained with the same original training data and random seed.
- *Performance degradation on clean data.* This metric captures the side effects of poisoning, by evaluating the ability of the backdoored model to maintain its predictive performance on clean samples. Let F_1^p be the F1 score of the poisoned model on the clean test set, and F_1^c the test score of a non-poisoned model trained equally, the performance degradation on clean data at runtime is: $\Delta F_1 = |F_1^p - F_1^c|$.

Unless otherwise noted, all the results shown in the following sections are averages of five experiments with different random seeds, reported with their relative standard deviations.

Table 4: Base performance of the classifiers, avg. over 5 runs.

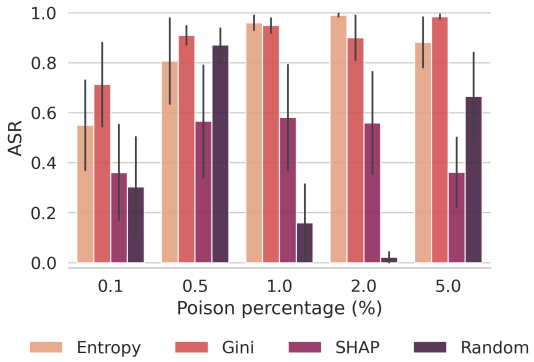
Model	Accuracy	F1 score	Precision	Recall
CTU-13 Neris Botnet				
GB	0.999	0.959	0.996	0.925
FFNN	0.999	0.927	0.971	0.887
CIC-IDS 2018 Botnet				
GB	0.999	0.994	0.993	0.995
FFNN	0.999	0.995	0.999	0.991
ISCX 2016 File/Video				
GB	0.962	0.800	0.799	0.802
FFNN	0.941	0.719	0.666	0.780
ISCX 2016 Chat/Video				
GB	0.936	0.901	0.928	0.875
FFNN	0.947	0.919	0.939	0.900

Parameters. We define $p\%$ as the percentage of feature-space points of the training dataset that have been compromised by an adversary. Since the amount of poisoned points is generally a critical parameter of any poisoning attack, we measure the attack performance across multiple poison percentage values $p\%$. At runtime, we randomly select a subset of test points to inject the trigger. Specifically, we select 200 points for the CTU-13 and CIC IDS 2018 datasets, and 80 for the CIC ISCX 2016 dataset (due its smaller size).

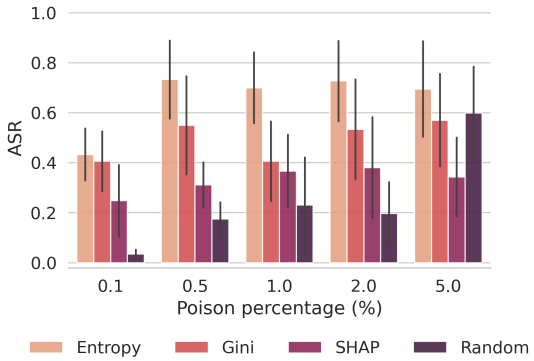
Baseline Model Performance. As mentioned in our threat model, we consider two representative classifiers: a Gradient Boosting Decision Tree (GB), and a Feed Forward Neural Network (FFNN). Note that we are not interested in finding the most effective possible learner for the classification task at hand, instead our focus is on selecting generic and widely adopted classifiers to showcase the adaptability of our attack strategy. Baseline values for accuracy, F1 score, precision and recall of the classifiers are reported in Table 4.

5.2 Impact of Feature Selection

Similar to the procedure reported in [72], our initial feature selection strategy revolved around computing local feature importance scores with SHAP and then aggregating them to obtain global indicators for each feature of the magnitude and direction of impact for each feature. As mentioned in Section 4.1, however, this approach has an important drawback: it requires to perform a potentially large



(a) Gradient Boosting model



(b) Feed-forward Neural Network model

Figure 5: Attack success rate (ASR) for the CTU-13 Neris Botnet scenario with different models and feature selection strategies.

number of queries against the victim classifier. To obviate this issue, we also considered ways in which the adversary can extract feature importance estimates directly from their data subset, D_a. In practice, we experimented with fitting a Decision Tree on D_a, following either the Gini impurity (*Gini*) or the information gain (*Entropy*) criteria, and using the importance estimate given by the reduction of the criterion induced by the feature[2].

The three feature selection strategies implemented (Entropy, Gini, SHAP) use the top eight most important features to design the trigger pattern, and are compared against Random, a baseline strategy that chooses the same number of features uniformly at random. Looking at the features selected by the different strategies, we generally observe that Entropy and Gini tend to assign scores that are strongly positive only for a very small number of features (typically 1-3), while SHAP scores are distributed more evenly. This observation, together with the desire to minimize the trigger footprint, informed our decision to select eight most relevant features. We also experimented with different values of this parameter, halving and doubling the number of selected features, but we found that eight were sufficient to achieve satisfying success rates.

[2]Using the implementation in Scikit-Learn https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html

Attack Success Rate: We show the results of these experiments in Figure 5. On average, we found the Entropy strategy to be the most successful against both classifiers on this dataset. The Random strategy leads to inconsistent results: occasionally, it stumbles upon useful features, but overall attacks relying on Random selection perform worse than attacks guided by the other feature selection methods. Figure 5 also illustrates a major finding – our attacks perform well even at very small poisoning rates such as 0.1%, where they reach an attack success rate of up to 0.7 against the Gradient Boosting classifier. As expected, increasing the poisoning percentage leads to an increase in attack success rate; for instance, an ASR of 0.95 is obtained with Entropy at 1.0% poisoning. This is interesting considering that previous works only considered larger poisoning rates (e.g, 2% to 20% in [40], 20% samples from nine (out of ten) non-target classes in [57]). We also notice that some of the variance in the ASR results can be attributed to a somewhat bimodal distribution. This can be partially explained with differences in the resulting trigger sizes, with Figure 6b highlighting the correlation between larger triggers and higher ASR. We leave a more detailed analysis of the distribution of the ASR scores for future work. The second interesting observation we can make, is that the SHAP strategy, while working well in some scenarios (especially for the application classification tasks in Section 5.5) does not, on average, lead to better results than estimating feature importance through proxy models (Entropy and Gini). This makes the attack quite easier to run in practice, as it circumvents the necessity to run multiple, potentially expensive, queries to the victim model.

Performance degradation on clean data: While these results show that the poisoned model is able to successfully misclassify *poisoned* data, we also want make sure that the performance on *clean* data is maintained. The average ΔF_1 across poisoning rates and feature selection strategies in our experiments was below 0.037, demonstrating that the side effects of the attack are minimal. The neural network model exhibits on average a slightly larger decrease when compared against the Gradient Boosting classifier, especially when the Entropy and Gini feature selection strategies are used.

5.3 Attack Stealthiness

Remaining undetected is an important factor in running a successful poisoning campaign. Here, we study the impact of our two approaches for increasing attack stealthiness described in Section 4.2: reducing the trigger size (*Reduced* trigger) and generating the trigger connections using Bayesian networks (*Generated* trigger). We start by analyzing the attack success with the different types of triggers, followed by a quantitative comparison of their stealthiness in feature space (via anomaly detection), and in problem space (via the Jensen-Shannon distance).

Evaluation of attack success. Figure 6a shows the attack success rate as a function of the poisoning percentage for the three different types of triggers: Full, Reduced, and Generated. We observe that all triggers are able to mount effective attacks against the Gradient Boosting classifier, with attack success rates over 0.8 when 0.5% or more of the training data is poisoned. The Feed-forward Neural Network, is generally more resilient to our attacks: the Full trigger and Reduced trigger deliver an attack success rate of about 0.7 and

Table 5: Area under the Precision-Recall Curve and F1 score obtained by performing anomaly detection on the poisoned data with an Isolation Forest model trained on a clean subset of the training data. CTU-13 Neris, at 1% poisoning rate.

Strategy	Model	Trigger	PR AUC	F_1 score
Entropy	Any	Full	0.056	0.013
		Reduced	0.045	0.012
		Generated	0.078	0.018
SHAP	Gradient Boosting	Full	0.099	0.015
		Reduced	0.070	0.013
		Generated	0.099	0.019
	Feed-forward NN	Full	0.061	0.015
		Reduced	0.047	0.014
		Generated	0.052	0.012

0.4, respectively, while the Generative trigger is able to synthesize more effective triggers, which lead to attack success rates over 0.7.

Figure 6b studies the correlation between trigger size (measured in number of connections) and attack success rate for each type of trigger. Each data point represented in the figure constitutes a separate experiment, while the regression lines capture the trend (how ASR changes as the trigger size changes). These figures show that the generative method leads to consistently smaller triggers than the other two methods, without sacrificing attack success. This result is indicative of the power of generative models in knowledge discovery, and, in our case, their ability to synthesize a small set of realistic log connections that lead to the feature-space prototype. Figure 6b also shows that the size reduction strategy is able to create triggers (Reduced trigger) that are smaller than the Full trigger, but at the expense of the attack success rate.

Evaluation of attack stealthiness in feature space. Next, we evaluate the attack stealthiness in feature space, using the Isolation Forest [44] algorithm for anomaly detection. The objective of this experiment is to see whether a standard technique for anomaly detection can identify and flag the poisoned samples as anomalies. The anomaly detector is trained on a clean subset of data, which is completely disjoint from the poisoned data points and consists of 10% of the entire training dataset.

Table 5 presents the anomaly detection results on the poisoned data obtained with each trigger type (Full, Reduced, and Generated). For comparison, we evaluate both the entropy-based and the SHAP-based feature selection strategies used to craft the injected pattern. Since SHAP queries the model to compute feature relevance scores, we present the anomaly detection results separately for a SHAP-guided attack against a Gradient Boosting classifier and against a Feed-forward Neural Network. Across the board, we observe very low Precision-Recall area under the curve (AUC) scores (in the 0.045 – 0.099 range), as well as very low F_1 scores (in the 0.012 – 0.019 range). These results demonstrate the difficulty of differentiating the poisoned data points from the clean data points, and indicate that the poisoning attacks are highly inconspicuous in feature space.

Evaluation of attack stealthiness in problem space. We also evaluate attack stealthiness in problem space, in terms of how close the poisoned data is to the target class, here represented by the benign class (normal traffic). We leverage the Jensen-Shannon divergence [42], a normalized and symmetrical scoring method for measuring the similarity between two probability distributions, and in particular we use the distance formulation defined as the square root of the divergence, which assumes value of zero for identical distributions. We compute the distance for each field in the connection logs (e.g., bytes, port, connection state, etc.), and report the average across all fields. As a baseline, we compute the average Jensen-Shannon distance between the target class points (benign log connections only) of the training and test datasets, capturing the distribution shift between train and test data. For the CTU-13 Neris Botnet dataset, we evaluated this reference distance as being D_REF = JS(TRAIN, TEST) = 0.24.

Figure 7 shows the Jensen-Shannon distance between the poisoned and clean training dataset for each of the trigger types. The figure illustrates that all three strategies produce stealthy attacks, characterized by average Jensen-Shannon distances that are comfortably lower than D_REF. Furthermore, the generative method (Generated trigger) constructs the most inconspicuous triggers, followed by the trigger size reduction method (Reduced trigger).

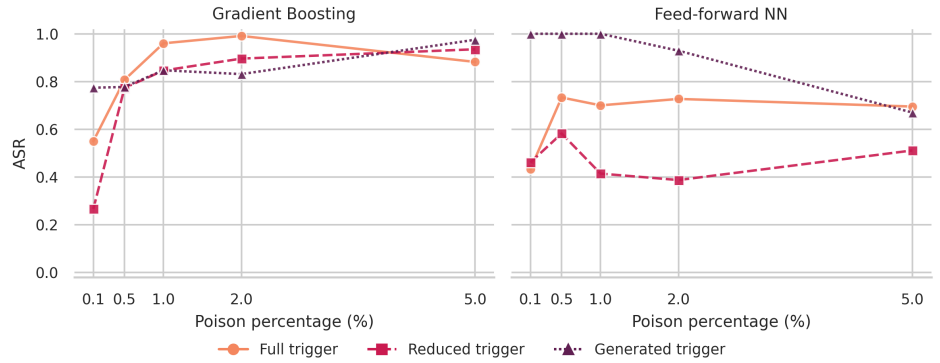
Table 6: Results on the CTU-13 Neris Botnet scenario, where the victim model uses an auto-encoder to learn the feature representation. Entropy strategy.

Poison budget	0.5%	1%	2%	4%	5%	10%
ASR	0.013	0.066	0.166	0.362	0.406	0.634
Stand. dev.	0.009	0.045	0.134	0.100	0.140	0.109
ΔF_1 **Test**	0.002	0.003	0.005	0.009	0.011	0.007

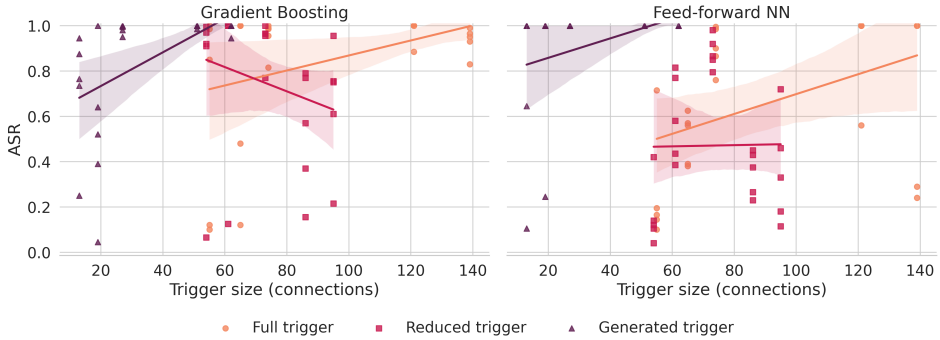
5.4 Impact of Feature Representation

The feature representation used by the learning task can strongly influence the attack success. In the next set of experiments, we study feature encodings, which are automatically learned with an auto-encoder architecture. Together with statistical features, encoded features are representative in network traffic classification, and auto-encoder models have been widely adopted for this task by previous works [14, 25, 51, 87]. To generate these features, we first train an auto-encoder model in an unsupervised manner, with the goal of minimizing the reconstruction error. Then the encoder portion of the model is run on the same training data to extract the high-level features used to train the feed-forward neural network architecture considered in previous experiments. Since the auto-encoder requires its inputs to be of a consistent shape, instead of features extracted from 30-second time windows, here the model is provided with an input representation consisting of contiguous blocks of 100 connections. Given that features are extracted from connection blocks of a fixed size, we also fix the trigger size to be 50 connections long. We found this value empirically by experimenting with different trigger sizes, and noticed that smaller ones would lead to unsatisfying attack results. While the trigger is relatively large compared to the unit block size, it is worth noting that the total number of connections introduced by the attack is still very limited when compared to the size of the the training set.

Table 6 reports the results of the Entropy strategy when applied in this setup, at different poison percentages, together with its standard deviation across 5 experiments and the average degradation



(a) Comparison of attack success rates (ASR) as a function of poisoning percentage.



(b) Correlation between the number of connections composing the trigger and the attack success rate (ASR). Each point represents a separate experiment. Curve fitting illustrating the trend is performed using linear regression.

Figure 6: Analysis of trigger selection strategy. CTU-13 Neris Botnet scenario, with the Entropy feature selection strategy.

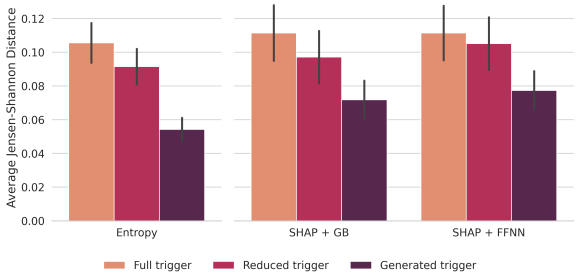


Figure 7: Jensen-Shannon distance between the poisoned and clean training dataset, averaged over all considered *conn.log* fields. For reference, the average JS distance value between the original training data and test data is 0.24. CTU-13 Neris Botnet experiments, at 1% poisoning rate.

in performance of the victim model on clean data. Since the auto-encoder was trained in an unsupervised fashion to minimize the reconstruction loss, we expect this training loss to impact negatively the overall success of the attack. In fact, we do observe a general reduction of the success rate compared to the simple neural network model, especially for limited poisoning budgets ($\leq$ 1%). However, if the adversary is allowed to increase the poisoning rate beyond 1%, we observe that the attack scales nicely with larger poisoning budgets. At the same time, the ΔF_1 values remain generally low even at larger poison percentages.

5.5 Other datasets

In the previous sections, we carried out an in-depth evaluation of various attack characteristics and their impact on the attack success. In this section, we investigate how generalizable this poisoning approach is by testing it on different datasets and other classification tasks. We evaluate here a second cybersecurity task on the CIC IDS 2018 dataset, and two application classification scenarios on CIC ISCX 2016. For all of these case studies, we use the statistical features (see Table 2) and the full trigger strategy.

We report the attack success rate at different poisoning percentages in Figure 8. Due to the much smaller size of the ISCX dataset, we test up to slightly larger poison percentage values — for instance in the Chat/Video scenario, 0.1% of the training set would amount to a single poisoning point. In general, we observe similar trends as in previous experiments, with the SHAP and Entropy strategies performing similarly, and achieving significant attack success rates even with very limited poison budgets.

We also evaluated the poisoned model on clean test data, to verify whether the poisoned model is still able to classify clean test data correctly. We obtained very limited reductions in F_1 scores:

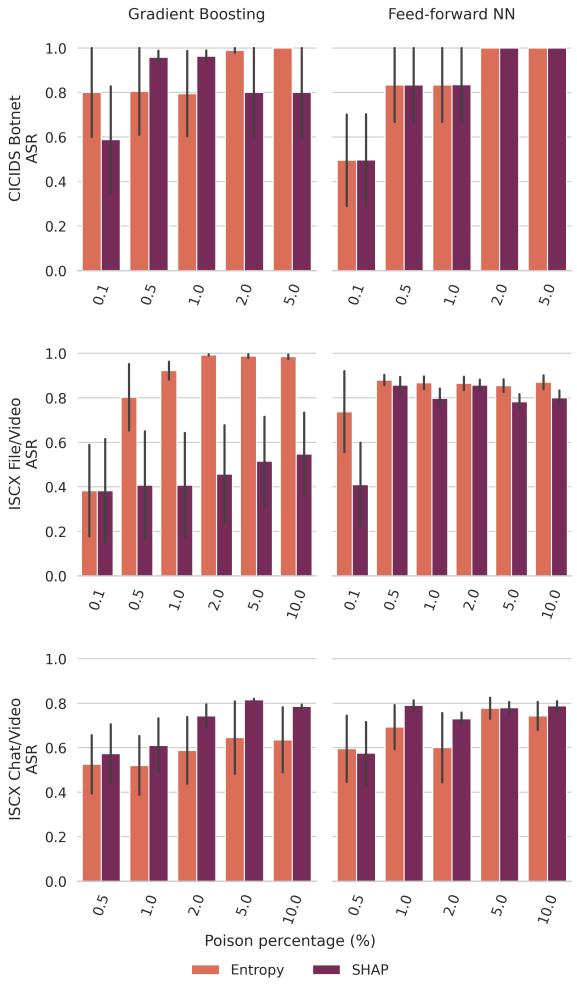


Figure 8: Attack success rate (ASR) on the CIC IDS 2018 Botnet and the CIC ISCX 2016 dataset, full trigger.

ΔF_1 is between 0.002 and 0.046, with the SHAP strategy resulting in slightly larger shifts than the other feature selection methods.

6 DISCUSSION AND LIMITATIONS

Despite our efforts towards the practical feasibility of the attack we propose, poisoning complex data is still a challenging task, and there are some elements that could increase the difficulty of deploying this attack on an arbitrary victim network. Regarding the problem-space mapping of the triggers, the adversary may experience a situation where two connection events are inter-dependent, due to the internal state of Zeek, but the trigger does not include both of them simultaneously — this could occur if the connections happen across the border of two time windows. For instance, inter-dependent connection events may take place in the case of hosts running the FTP protocol. Documentation on this type of connections for Zeek is quite scarce, but a dedicated attacker could allocate time and resources to enumerate all possible corner cases and explicitly avoid them during the trigger creation phase.

Another potential source of issues could arise when using the generated trigger approach. This method leads to a generally good attack success with a small footprint, however, it could in principle generate connections that are not feasible in practice for stateful protocols (TCP). There are two possible ways to address this potential issue. First, given the relentless pace of improvements in generative models, including those targeting tabular data [7, 86], we expect that the ability of generative models to infer the inter-features constraints that characterize this data modality will increase significantly in the very short term. In parallel, the adversary could attempt to verify the correctness of the generated connections using a model checker and a formal model of the TCP protocol, and simply reject the non-conforming ones. Both approaches are exciting avenues for future research, and we leave their in-depth analysis to future work.

Finally, we designed methods to hide the poisoning campaign, and showed that our poisoning points are difficult to identify both in feature space, by using anomaly detection techniques, and in problem space, by analysing the distributional distance of poisoned data. Defending ML models from backdoor attacks is an open, and extremely complex, research problem. Many of the current proposed solutions are designed to operate in the computer vision domain [10], or on specific model architectures [45, 79]. In contrast, our attack method generalizes to different model typologies. Moreover, initial research on defending classifiers from backdoor attacks in the security domain [27] highlighted potential trade-offs between robustness and utility (e.g., defenses that rely on data sanitization may mistakenly remove a high number of benign samples in an attempt to prune out potentially poisoned samples). By releasing new attack strategies, we hope to encourage future research in the challenging direction of defending against backdoor attacks on network traffic.

7 CONCLUSIONS

With this work we investigated the possibility of carrying out data-only, clean-label, poisoning attacks against network flow classifiers. We believe this threat model holds substantial significance for the security community, due to its closer alignment with the capabilities exhibited by sophisticated adversaries observed in the wild, and the current best practices in secure ML deployments, in contrast to other prevailing models frequently employed.

The attack strategy we introduce can effectively forge consistent associations between the trigger pattern and the target class even at extremely low poisoning rates (0.1-0.5% of the training set size). This results in notable attack success rates, despite the constrained nature of the attacker. While the attack is effective, it has minimal impacts on the victim model's generalization abilities when dealing with clean test data. Additionally, the detectability of the trigger can be lessened through different strategies to decrease the likelihood of a defender discovering an ongoing poisoning campaign.

Furthermore, we demonstrated that this form of poisoning has a relatively wide applicability for various objectives across different types of classification tasks. The implications of these findings extend our understanding of ML security in practical contexts, and prompt further investigation into effective defense strategies against these refined attack methodologies.

ACKNOWLEDGEMENTS

This research was sponsored by MIT Lincoln Laboratory, the U.S. Army Combat Capabilities Development Command Army Research Laboratory (DEVCOM ARL) under Cooperative Agreement Number W911NF-13-2-0045, and the Department of Defense Multidisciplinary Research Program of the University Research Initiative (MURI) under contract W911NF-21-1-0322.

REFERENCES

[1] Emre Kiciman Andrew Marshall, Jugal Parikh and Ram Shankar Siva Kumar. 2022. Threat Modeling AI/ML Systems and Dependencies. https://learn.microsoft.com/en-us/security/engineering/threat-modeling-aiml.

[2] Manos Antonakakis, Roberto Perdisci, David Dagon, Wenke Lee, and Nick Feamster. 2010. Building a Dynamic Reputation System for DNS. In *Proceedings of the 19th USENIX Conference on Security* (Washington, DC) *(USENIX Security'10)*. USENIX Association, USA, 18.

[3] Manos Antonakakis, Roberto Perdisci, Wenke Lee, Nikolaos Vasiloglou, and David Dagon. 2011. Detecting Malware Domains at the Upper DNS Hierarchy. In *Proceedings of the 20th USENIX Conference on Security* (San Francisco, CA) *(SEC'11)*. USENIX Association, USA, 27.

[4] Giovanni Apruzzese, Michele Colajanni, Luca Ferretti, and Mirco Marchetti. 2019. Addressing Adversarial Attacks Against Security Systems Based on Machine Learning. In *2019 11th International Conference on Cyber Conflict (CyCon)*, Vol. 900. 1–18. https://doi.org/10.23919/CYCON.2019.8756865

[5] Md. Ahsan Ayub, William A. Johnson, Douglas A. Talbert, and Ambareen Siraj. 2020. Model Evasion Attack on Intrusion Detection Systems using Adversarial Machine Learning. In *2020 54th Annual Conference on Information Sciences and Systems (CISS)*. 1–6. https://doi.org/10.1109/CISS48834.2020.1570617116

[6] Alexander Binder, Grégoire Montavon, Sebastian Lapuschkin, Klaus-Robert Müller, and Wojciech Samek. 2016. Layer-wise relevance propagation for neural networks with local renormalization layers. In *Artificial Neural Networks and Machine Learning–ICANN 2016: 25th International Conference on Artificial Neural Networks, Barcelona, Spain, September 6-9, 2016, Proceedings, Part II 25*. Springer, 63–71.

[7] Stavroula Bourou, Andreas El Saer, Terpsichori-Helen Velivassaki, Artemis Voulkidis, and Theodore Zahariadis. 2021. A Review of Tabular Data Synthesis Using GANs on an IDS Dataset. *Information* 12, 9 (Sept. 2021), 375. https://doi.org/10.3390/info12090375

[8] Martin Burkhart, Mario Strasser, Dilip Many, and Xenofontas Dimitropoulos. 2010. SEPIA: Privacy-Preserving Aggregation of Multi-Domain Network Events and Statistics. In *19th USENIX Security Symposium (USENIX Security 10)*. USENIX Association, Washington, DC. https://www.usenix.org/conference/usenixsecurity10/sepia-privacy-preserving-aggregation-multi-domain-network-events-and

[9] Xiaoyu Cao and Neil Zhenqiang Gong. 2017. Mitigating Evasion Attacks to Deep Neural Networks via Region-Based Classification. In *Proceedings of the 33rd Annual Computer Security Applications Conference* (Orlando, FL, USA) *(ACSAC '17)*. Association for Computing Machinery, New York, NY, USA, 278–287. https://doi.org/10.1145/3134600.3134606

[10] Bryant Chen, Wilka Carvalho, Nathalie Baracaldo, Heiko Ludwig, Benjamin Edwards, Taesung Lee, Ian Molloy, and Biplav Srivastava. 2019. Detecting Backdoor Attacks on Deep Neural Networks by Activation Clustering. In *Workshop on Artificial Intelligence Safety*. CEUR-WS.

[11] Haipeng Chen, Sushil Jajodia, Jing Liu, Noseong Park, Vadim Sokolov, and V. S. Subrahmanian. 2019. FakeTables: Using GANs to Generate Functional Dependency Preserving Tables with Bounded Real Data. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*. International Joint Conferences on Artificial Intelligence Organization, Macao China, 2074–2080. https://doi.org/10.24963/ijcai.2019/287

[12] Xinyun Chen, Chang Liu, Bo Li, Kimberly Lu, and Dawn Song. 2017. Targeted Backdoor Attacks on Deep Learning Systems Using Data Poisoning. *CoRR* abs/1712.05526 (2017). arXiv:1712.05526 http://arxiv.org/abs/1712.05526

[13] Alesia Chernikova and Alina Oprea. 2022. FENCE: Feasible Evasion Attacks on Neural Networks in Constrained Environments. *ACM Trans. Priv. Secur.* 25, 4, Article 34 (jul 2022), 34 pages. https://doi.org/10.1145/3544746

[14] Gianni D'Angelo and Francesco Palmieri. 2021. Network Traffic Classification Using Deep Convolutional Recurrent Autoencoder Neural Networks for Spatial–Temporal Features Extraction. *Journal of Network and Computer Applications* 173 (Jan. 2021), 102890. https://doi.org/10.1016/j.jnca.2020.102890

[15] Tristan Deleu, António Góis, Chris Chinenye Emezue, Mansi Rankawat, Simon Lacoste-Julien, Stefan Bauer, and Yoshua Bengio. 2022. Bayesian Structure Learning with Generative Flow Networks. In *The 38th Conference on Uncertainty in Artificial Intelligence*.

[16] Nagaraju Devarakonda, Srinivasulu Pamidi, V. Valli Kumari, and A. Govardhan. 2012. Intrusion Detection System using Bayesian Network and Hidden Markov Model. *Procedia Technology* 4 (2012), 506–514. https://doi.org/10.1016/j.protcy.2012.05.081 2nd International Conference on Computer, Communication, Control and Information Technology(C3IT-2012) on February 25 - 26, 2012.

[17] Gerard Draper-Gil, Arash Habibi Lashkari, Mohammad Saiful Islam Mamun, and Ali A. Ghorbani. 2016. Characterization of Encrypted and VPN Traffic Using Time-related Features:. In *Proceedings of the 2nd International Conference on Information Systems Security and Privacy*. SCITEPRESS - Science and and Technology Publications, Rome, Italy, 407–414. https://doi.org/10.5220/0005740704070414

[18] Justin Engelmann and Stefan Lessmann. 2021. Conditional Wasserstein GAN-based Oversampling of Tabular Data for Imbalanced Learning. *Expert Systems with Applications* 174 (01 2021), 114582. https://doi.org/10.1016/j.eswa.2021.114582

[19] Ju Fan, Junyou Chen, Tongyu Liu, Yuwei Shen, Guoliang Li, and Xiaoyong Du. 2020. Relational data synthesis using generative adversarial networks: a design space exploration. *Proceedings of the VLDB Endowment* 13 (08 2020), 1962–1975. https://doi.org/10.14778/3407790.3407802

[20] S. García, M. Grill, J. Stiborek, and A. Zunino. 2014. An Empirical Comparison of Botnet Detection Methods. *Computers and Security* 45 (Sept. 2014), 100–123. https://doi.org/10.1016/j.cose.2014.05.011

[21] Joseph Gastwirth. 1972. The Estimation of the Lorenz Curve and Gini Index. *The Review of Economics and Statistics* 54 (02 1972), 306–16. https://doi.org/10.2307/1937992

[22] Kathrin Grosse, Lukas Bieringer, Tarek R. Besold, Battista Biggio, and Katharina Krombholz. 2023. Machine Learning Security in Industry: A Quantitative Survey. *IEEE Transactions on Information Forensics and Security* 18 (2023), 1749–1762. https://doi.org/10.1109/TIFS.2023.3251842

[23] T. Gu, K. Liu, B. Dolan-Gavitt, and S. Garg. 2019. BadNets: Evaluating Backdooring Attacks on Deep Neural Networks. *IEEE Access* 7 (2019), 47230–47244. https://doi.org/10.1109/ACCESS.2019.2909068

[24] Mark Handley, Vern Paxson, and Christian Kreibich. 2001. Network Intrusion Detection: Evasion, Traffic Normalization, and End-to-End Protocol Semantics. In *10th USENIX Security Symposium (USENIX Security 01)*. USENIX Association, Washington, D.C. https://www.usenix.org/conference/10th-usenix-security-symposium/network-intrusion-detection-evasion-traffic-normalization

[25] Mingshu He, Xiaojuan Wang, Junhua Zhou, Yuanyuan Xi, Lei Jin, and Xinlei Wang. 2021. Deep-Feature-Based Autoencoder Network for Few-Shot Malicious Traffic Detection. *Security and Communication Networks* 2021 (March 2021), e6659022. https://doi.org/10.1155/2021/6659022

[26] David Heckerman. 2008. *A Tutorial on Learning with Bayesian Networks*. Springer Berlin Heidelberg, Berlin, Heidelberg, 33–82. https://doi.org/10.1007/978-3-540-85066-3_3

[27] Samson Ho, Achyut Reddy, Sridhar Venkatesan, Rauf Izmailov, Ritu Chadha, and Alina Oprea. 2022. Data Sanitization Approach to Mitigate Clean-Label Attacks Against Malware Detection Systems. In *MILCOM 2022 - 2022 IEEE Military Communications Conference (MILCOM)*. 993–998. https://doi.org/10.1109/MILCOM55135.2022.10017768

[28] Jordan Holland, Paul Schmitt, Prateek Mittal, and Nick Feamster. 2022. Towards Reproducible Network Traffic Analysis. arXiv:2203.12410 [cs]

[29] John T. Holodnak, Olivia Brown, Jason Matterer, and Andrew Lemke. 2022. Backdoor Poisoning of Encrypted Traffic Classifiers. In *2022 IEEE International Conference on Data Mining Workshops (ICDMW)*. 577–585. https://doi.org/10.1109/ICDMW58026.2022.00080

[30] IBM. 2023. IBM Security QRadar XDR. https://www.ibm.com/qradar.

[31] Sam Ingalls. 2021. Top XDR Security Solutions for 2022. https://www.esecurityplanet.com/products/xdr-security-solutions/.

[32] Luca Invernizzi, Sung ju Lee, Stanislav Miskovic, Marco Mellia, Ruben Torres, Christopher Kruegel, Sabyasachi Saha, and Giovanni Vigna. 2014. Nazca: Detecting Malware Distribution in Large-Scale Networks. In *NDSS*.

[33] M.A. Jabbar, Rajanikanth Aluvalu, and S. Sai Satyanarayana Reddy. 2017. Intrusion Detection System Using Bayesian Network and Feature Subset Selection. In *2017 IEEE International Conference on Computational Intelligence and Computing Research (ICCIC)*. 1–5. https://doi.org/10.1109/ICCIC.2017.8524381

[34] Arthur S. Jacobs, Roman Beltiukov, Walter Willinger, Ronaldo A. Ferreira, Arpit Gupta, and Lisandro Z. Granville. 2022. AI/ML for Network Security: The Emperor Has No Clothes. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security* (Los Angeles, CA, USA) *(CCS '22)*. Association for Computing Machinery, New York, NY, USA, 1537–1551. https://doi.org/10.1145/3548606.3560609

[35] Dhamanpreet Kaur, Matthew Sobiesk, Shubham Patil, Jin Liu, Puran Bhagat, Amar Gupta, and Natasha Markuzon. 2020. Application of Bayesian networks to generate synthetic health data. *Journal of the American Medical Informatics Association* 28 (12 2020). https://doi.org/10.1093/jamia/ocaa303

[36] Neville Kenneth Kitson, Anthony C. Constantinou, Zhigao Guo, Yang Liu, and Kiattikun Chobtham. 2023. A survey of Bayesian Network structure learning. *Artificial Intelligence Review* (2023). https://doi.org/10.1007/s10462-022-10351-w

[37] Daphne Koller and Mehran Sahami. 1996. Toward Optimal Feature Selection. In *Proceedings of the Thirteenth International Conference on International Conference on Machine Learning* (Bari, Italy) *(ICML'96)*. Morgan Kaufmann Publishers Inc.,

San Francisco, CA, USA, 284–292.

[38] Akim Kotelnikov, Dmitry Baranchuk, Ivan Rubachev, and Artem Babenko. 2022. TabDDPM: Modelling Tabular Data with Diffusion Models. https://doi.org/10.48550/arXiv.2209.15421 arXiv:2209.15421 [cs]

[39] Changki Lee and Gary Geunbae Lee. 2006. Information gain and divergence-based feature selection for machine learning-based text categorization. *Information Processing & Management* 42, 1 (2006), 155–165. https://doi.org/10.1016/j.ipm.2004.08.006 Formal Methods for Information Retrieval.

[40] Pan Li, Qiang Liu, Wentao Zhao, Dongxu Wang, and Siqi Wang. 2018. Chronic Poisoning against Machine Learning Based IDSs Using Edge Pattern Detection. In *2018 IEEE International Conference on Communications (ICC)*. 1–7. https://doi.org/10.1109/ICC.2018.8422328

[41] Pan Li, Qiang Liu, Wentao Zhao, Dongxu Wang, and Siqi Wang. 2018. Chronic Poisoning against Machine Learning Based IDSs Using Edge Pattern Detection. In *2018 IEEE International Conference on Communications (ICC)*. 1–7. https://doi.org/10.1109/ICC.2018.8422328

[42] J. Lin. 1991. Divergence measures based on the Shannon entropy. *IEEE Transactions on Information Theory* 37, 1 (1991), 145–151. https://doi.org/10.1109/18.61115

[43] Pantelis Linardatos, Vasilis Papastefanopoulos, and Sotiris Kotsiantis. 2021. Explainable AI: A Review of Machine Learning Interpretability Methods. *Entropy* 23, 1 (2021). https://doi.org/10.3390/e23010018

[44] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. 2008. Isolation Forest. In *2008 Eighth IEEE International Conference on Data Mining*. IEEE, Pisa, Italy, 413–422. https://doi.org/10.1109/ICDM.2008.17

[45] Kang Liu, Brendan Dolan-Gavitt, and Siddharth Garg. 2018. Fine-Pruning: Defending Against Backdooring Attacks on Deep Neural Networks. In *Research in Attacks, Intrusions, and Defenses (Lecture Notes in Computer Science)*, Michael Bailey, Thorsten Holz, Manolis Stamatogiannakis, and Sotiris Ioannidis (Eds.). Springer International Publishing, Cham, 273–294. https://doi.org/10.1007/978-3-030-00470-5_13

[46] Yingqi Liu, Shiqing Ma, Yousra Aafer, Wen-Chuan Lee, Juan Zhai, Weihang Wang, and Xiangyu Zhang. 2018. Trojaning Attack on Neural Networks. In *25th Annual Network and Distributed System Security Symposium, NDSS 2018, San Diego, California, USA, February 18-21, 2018*. The Internet Society. http://wp.internetsociety.org/ndss/wp-content/uploads/sites/25/2018/02/ndss2018_03A-5_Liu_paper.pdf

[47] Scott M. Lundberg, Gabriel Erion, Hugh Chen, Alex DeGrave, Jordan M. Prutkin, Bala Nair, Ronit Katz, Jonathan Himmelfarb, Nisha Bansal, and Su-In Lee. 2020. From local explanations to global understanding with explainable AI for trees. *Nature Machine Intelligence* 2, 1 (2020), 56–67. https://doi.org/10.1038/s42256-019-0138-9

[48] Scott M Lundberg and Su-In Lee. 2017. A Unified Approach to Interpreting Model Predictions. In *Advances in Neural Information Processing Systems*, I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (Eds.), Vol. 30. Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2017/file/8a20a8621978632d76c43dfd28b67767-Paper.pdf

[49] MalwareGuard FireEye 2020. MalwareGuard: FireEye's Machine Learning Model to Detect and Prevent Malware. https://www.fireeye.com/blog/products-and-services/2018/07/malwareguard-fireeye-machine-learning-model-to-detect-and-prevent-malware.html.

[50] Microsoft. 2021. Microsoft Defender for Endpoint | Microsoft Security. https://www.microsoft.com/en-us/security/business/threat-protection/endpoint-defender

[51] Yisroel Mirsky, Tomer Doitshman, Yuval Elovici, and Asaf Shabtai. 2018. Kitsune: An Ensemble of Autoencoders for Online Network Intrusion Detection. In *Proceedings 2018 Network and Distributed System Security Symposium*. Internet Society, San Diego, CA. https://doi.org/10.14722/ndss.2018.23204

[52] Andrew Moore, Denis Zuev, and Michael Crogan. 2005. *Discriminators for Use in Flow-Based Classification*. Technical Report. Queen Mary and Westfield College, Department of Computer Science.

[53] B. Mukherjee, L.T. Heberlein, and K.N. Levitt. 1994. Network Intrusion Detection. *IEEE Network* 8, 3 (May 1994), 26–41. https://doi.org/10.1109/65.283931

[54] Terry Nelms, Roberto Perdisci, and Mustaque Ahamad. 2013. ExecScent: Mining for New C&C Domains in Live Networks with Adaptive Control Protocol Templates. In *Proceedings of the 22nd USENIX Conf. on Security*. USENIX Association, USA, 589–604.

[55] Blaine Nelson, Marco Barreno, Fuching Jack Chi, Anthony D. Joseph, Benjamin I. P. Rubinstein, Udam Saini, Charles Sutton, J. D. Tygar, and Kai Xia. 2008. Exploiting Machine Learning to Subvert Your Spam Filter. In *Proceedings of the 1st Usenix Workshop on Large-Scale Exploits and Emergent Threats* (San Francisco, California) *(LEET'08)*. USENIX Association, USA, Article 7, 9 pages.

[56] James Newsome, Brad Karp, and Dawn Song. 2006. Paragraph: Thwarting Signature Learning by Training Maliciously. In *Recent Advances in Intrusion Detection*, Diego Zamboni and Christopher Kruegel (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 81–105.

[57] Rui Ning, Chunsheng Xin, and Hongyi Wu. 2022. TrojanFlow: A Neural Backdoor Attack to Deep Learning-based Network Traffic Classifiers. In *IEEE INFOCOM 2022 - IEEE Conference on Computer Communications*. 1429–1438. https://doi.org/10.1109/INFOCOM48880.2022.9796878

[58] Talha Ongun, Simona Boboila, Alina Oprea, Tina Eliassi-Rad, Jason Hiser, and Jack W. Davidson. 2022. CELEST: Federated Learning for Globally Coordinated Threat Detection. *CoRR* abs/2205.11459 (2022). https://doi.org/10.48550/arXiv.2205.11459 arXiv:2205.11459

[59] Talha Ongun, Timothy Sakharaov, Simona Boboila, Alina Oprea, and Tina Eliassi-Rad. 2019. On Designing Machine Learning Models for Malicious Network Traffic Classification. arXiv:1907.04846 [cs, stat]

[60] Talha Ongun, Oliver Spohngellert, Benjamin Miller, Simona Boboila, Alina Oprea, Tina Eliassi-Rad, Jason Hiser, Alastair Nottingham, Jack Davidson, and Malathi Veeraraghavan. 2021. PORTFILER: Port-Level Network Profiling for Self-Propagating Malware Detection. In *2021 IEEE Conference on Communications and Network Security (CNS)*. 182–190. https://doi.org/10.1109/CNS53000.2021.9705045

[61] Alina Oprea, Zhou Li, Robin Norris, and Kevin Bowers. 2018. MADE: Security Analytics for Enterprise Threat Detection. In *Proceedings of Annual Computer Security Applications Conference (ACSAC)*. https://doi.org/10.1145/3274694.3274710

[62] Alina Oprea and Apostol Vassilev. 2023. *Adversarial Machine Learning: A Taxonomy and Terminology of Attacks and Mitigations (Draft)*. Technical Report NIST AI 100-2e2023 ipd. National Institute of Standards and Technology.

[63] Pavlos Papadopoulos, Oliver Thornewill von Essen, Nikolaos Pitropakis, Christos Chrysoulas, Alexios Mylonas, and William J. Buchanan. 2021. Launching Adversarial Attacks against Network Intrusion Detection Systems for IoT. *Journal of Cybersecurity and Privacy* 1, 2 (June 2021), 252–273. https://doi.org/10.3390/jcp1020014

[64] Pavlos Papadopoulos, Oliver Thornewill von Essen, Nikolaos Pitropakis, Christos Chrysoulas, Alexios Mylonas, and William J. Buchanan. 2021. Launching Adversarial Attacks against Network Intrusion Detection Systems for IoT. *Journal of Cybersecurity and Privacy* 1, 2 (2021), 252–273. https://doi.org/10.3390/jcp1020014

[65] R. Perdisci, M. Sharif, P. Fogla, W. Lee, and D. Dagon. 2006. Misleading Worm Signature Generators Using Deliberate Noise Injection. In *2012 IEEE Symposium on Security and Privacy*. IEEE Computer Society, Los Alamitos, CA, USA, 17–31. https://doi.org/10.1109/SP.2006.26

[66] Robert Philipp, Andreas Mladenow, Christine Strauss, and Alexander Völz. 2021. Machine Learning as a Service: Challenges in Research and Applications. In *Proceedings of the 22nd International Conference on Information Integration and Web-Based Applications & Services* (Chiang Mai, Thailand) *(iiWAS '20)*. Association for Computing Machinery, New York, NY, USA, 396–406. https://doi.org/10.1145/3428757.3429152

[67] Fabio Pierazzi, Feargus Pendlebury, Jacopo Cortellazzi, and Lorenzo Cavallaro. 2020. Intriguing Properties of Adversarial ML Attacks in the Problem Space. In *2020 IEEE Symposium on Security and Privacy (SP)*. IEEE, San Francisco, CA, USA, 1332–1349. https://doi.org/10.1109/SP40000.2020.00073

[68] Babak Rahbarinia, Roberto Perdisci, and Manos Antonakakis. 2015. Segugio: Efficient Behavior-Based Tracking of Malware-Control Domains in Large ISP Networks. In *2015 45th Annual IEEE/IFIP Int'l. Conf. on Dependable Systems and Networks*. IEEE, 403–414.

[69] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. 2014. Stochastic Backpropagation and Approximate Inference in Deep Generative Models. In *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32* (Beijing, China) *(ICML'14)*. JMLR.org, II–1278–II–1286.

[70] Mauro Ribeiro, Katarina Grolinger, and Miriam A.M. Capretz. 2015. MLaaS: Machine Learning as a Service. In *2015 IEEE 14th International Conference on Machine Learning and Applications (ICMLA)*. 896–902. https://doi.org/10.1109/ICMLA.2015.152

[71] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. " Why should i trust you?" Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*. 1135–1144.

[72] Giorgio Severi, Jim Meyer, Scott Coull, and Alina Oprea. 2021. Explanation-Guided Backdoor Poisoning Attacks Against Malware Classifiers. In *30th USENIX Security Symposium (USENIX Security 21)*. 1487–1504.

[73] Ali Shafahi, W. Ronny Huang, Mahyar Najibi, Octavian Suciu, Christoph Studer, Tudor Dumitras, and Tom Goldstein. 2018. Poison Frogs! Targeted Clean-Label Poisoning Attacks on Neural Networks. In *Advances in Neural Information Processing Systems*.

[74] Iman Sharafaldin, Arash Habibi Lashkari, and Ali A. Ghorbani. 2018. Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization:. In *Proceedings of the 4th International Conference on Information Systems Security and Privacy*. SCITEPRESS - Science and Technology Publications, Funchal, Madeira, Portugal, 108–116. https://doi.org/10.5220/0006639801080116

[75] Ryan Sheatsley, Blaine Hoak, Eric Pauley, Yohan Beugin, Michael J. Weisman, and Patrick McDaniel. 2021. On the Robustness of Domain Constraints. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security* (Virtual Event, Republic of Korea) *(CCS '21)*. Association for Computing Machinery, New York, NY, USA, 495–515. https://doi.org/10.1145/3460120.3484570

[76] Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. 2017. Learning important features through propagating activation differences. In *International conference on machine learning*. PMLR, 3145–3153.

[77] Ram Shankar Siva Kumar, Magnus Nyström, John Lambert, Andrew Marshall, Mario Goertzel, Andi Comissoneru, Matt Swann, and Sharon Xia. 2020. Adversarial Machine Learning-Industry Perspectives. In *2020 IEEE Security and Privacy Workshops (SPW)*. 69–75. https://doi.org/10.1109/SPW50608.2020.00028

[78] Acar Tamersoy, Kevin Roundy, and Duen Horng Chau. 2014. Guilt by Association: Large Scale Malware Detection by Mining File-Relation Graphs. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (New York, New York, USA) *(KDD '14)*. Association for Computing Machinery, New York, NY, USA, 1524–1533. https://doi.org/10.1145/2623330.2623342

[79] Brandon Tran, Jerry Li, and Aleksander Mądry. 2018. Spectral Signatures in Backdoor Attacks. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems (NIPS'18)*. Curran Associates Inc., Montréal, Canada, 8011–8021.

[80] Alexander Turner, Dimitris Tsipras, and Aleksander Mądry. 2018. Clean-Label Backdoor Attacks. *Manuscript submitted for publication* (2018), 21.

[81] Alexander Turner, Dimitris Tsipras, and Aleksander Madry. 2019. Label-Consistent Backdoor Attacks. arXiv:1912.02771 [stat.ML]

[82] María Vargas Muñoz, Rafael Martínez-Peláez, Pablo Velarde Alvarado, Efraín Moreno-Garcia, Deni Torres-Roman, and José Ceballos-Mejia. 2018. Classification of network anomalies in flow level network traffic using Bayesian networks. In *2018 International Conference on Electronics, Communications and Computers (CONIELECOMP)*. IEEE, 238–243. https://doi.org/10.1109/CONIELECOMP.2018.8327205

[83] Di Wu, Binxing Fang, Junnan Wang, Qixu Liu, and Xiang Cui. 2019. Evading Machine Learning Botnet Detection Models via Deep Reinforcement Learning. In *ICC 2019 - 2019 IEEE International Conference on Communications (ICC)*. IEEE, Shanghai, China, 1–6. https://doi.org/10.1109/ICC.2019.8761337

[84] Jing Xu and Christian R. Shelton. 2010. Intrusion Detection Using Continuous Time Bayesian Networks. *J. Artif. Int. Res.* 39, 1 (sep 2010), 745–774.

[85] Lei Xu, Maria Skoularidou, Alfredo Cuesta-Infante, and Kalyan Veeramachaneni. 2019. Modeling Tabular Data Using Conditional GAN. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems*. Curran Associates Inc., Red Hook, NY, USA, Article 659, 11 pages.

[86] Lei Xu, Maria Skoularidou, Alfredo Cuesta-Infante, and Kalyan Veeramachaneni. 2019. Modeling Tabular Data Using Conditional GAN. In *Advances in Neural Information Processing Systems*, Vol. 32. Curran Associates, Inc.

[87] Kun Yang, Samory Kpotufe, and Nick Feamster. 2021. Feature Extraction for Novelty Detection in Network Traffic. https://doi.org/10.48550/arXiv.2006.16993 arXiv:2006.16993 [cs]

[88] Limin Yang, Zhi Chen, Jacopo Cortellazzi, Feargus Pendlebury, Kevin Tu, Fabio Pierazzi, Lorenzo Cavallaro, and Gang Wang. 2023. Jigsaw Puzzle: Selective Backdoor Attack to Subvert Malware Classifiers. In *IEEE Symposium on Security & Privacy*.

[89] Jim Young, Patrick Graham, and Richard Penny. 2009. Using Bayesian Networks to Create Synthetic Data. *Journal of Official Statistics* 25 (12 2009), 549–567.

[90] Zilong Zhao, Aditya Kunar, Robert Birke, and Lydia Y. Chen. 2021. CTAB-GAN: Effective Table Data Synthesizing. In *Proceedings of The 13th Asian Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 157)*, Vineeth N. Balasubramanian and Ivor Tsang (Eds.). PMLR, 97–112. https://proceedings.mlr.press/v157/zhao21a.html