IXP Scrubber: Learning from Blackholing Traffic for ML-Driven DDoS Detection at Scale

Matthias Wichtlhuber¹, Eric Strehle², Daniel Kopp¹, Lars Prepens¹, Stefan Stegmueller¹
Alina Rubina¹, Christoph Dietzel¹, Oliver Hohlfeld²
¹DE-CIX  ²Brandenburg University of Technology

ABSTRACT
Distributed Denial of Service (DDoS) attacks are among the most critical cybersecurity threats, jeopardizing the stability of even the largest networks and services. The existing range of mitigation services predominantly filters at the edge of the Internet, thus creating unnecessary burden for network infrastructures. Consequently, we present IXP Scrubber, a Machine Learning (ML) based system for detecting and filtering DDoS traffic at the core of the Internet. It utilizes BGP signals to drop traffic for certain routes (blackholing) to sample DDoS and thus learn new attack vectors without the operator’s intervention and on unprecedented amounts of training data. We present three major contributions: i) a method to semi-automatically generate arbitrarily large amounts of labeled DDoS training data from IXPs’ sampled packet traces, ii) the novel, controllable, locally explainable and highly precise two-step IXP Scrubber ML model, and iii) an evaluation of the IXP Scrubber ML model, including its temporal and geographical drift, based on data from 5 IXPs covering a time span of up to two years.

CCS CONCEPTS
* Security and privacy → Denial-of-service attacks; * Networks → Wide area networks; Network monitoring; Public Internet.

KEYWORDS
Machine Learning, Traffic Classification, Denial of Service

1 INTRODUCTION
With our societies increasingly relying on online services, cyberattacks are becoming more frequent and devastating [17, 22, 43, 57].

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

SIGCOMM '22, August 22–26, 2022, Amsterdam, Netherlands
© 2022 Association for Computing Machinery.
ACM ISBN 978-1-4503-9420-8/22/08...
https://doi.org/10.1145/3544216.3544268

Figure 1: IXP Scrubber applies an ML DDoS classifier at IXPs at the Internet’s core and filters DDoS traffic for connected networks. It learns continuously from ASes (A-D) marking unwanted traffic (blackholing).

One of the most prevalent threats to online services to date are DDoS attacks [17, 35, 39, 46, 47, 52, 59]. DDoS attacks aim at consuming more critical resources than available to a service, e.g., network bandwidth, which makes protection against DDoS hard for victim operators. They are frequent (e.g., thousands of attacks can be observed at certain vantage points every single day [16, 37]), they can be conducted without technical expertise [38], and can generate attack volumes (e.g., of up to 3.5 Tbit/s observed in late 2021 [53]) that can threaten even the largest networks [15, 30, 53, 59]. The motivation to conduct criminal activities are manifold and include financial gain through ransom [21] or political motivation [42].

Current DDoS mitigation approaches detect and drop attack traffic close to the edge—with the downside that attack traffic is carried over the Internet before filtering. They can be roughly divided into the two categories of filtering (1) inside or (2) outside a victim’s network. The first category comprises solutions that are directly employed on the victim’s side (e.g., mitigation appliances or software stacks [6]). By locally monitoring and dropping incoming traffic, they protect against DDoS attacks, but are limited to the bandwidth that connects the victim’s network to the Internet. The second category comprises external services, i.e., inspection and dropping by external Traffic Scrubbing Services [44] (TSSes) as offered by TSS providers and CDNs; this requires a rerouting of traffic through the TSS/CDN infrastructure. In principle DDoS attacks could overload the bandwidth available at the scrubber and thus harm other customers. For both categories, DDoS traffic must traverse the Internet from edge to edge to be filtered.

Contribution. Consequently, this paper proposes IXP Scrubber (Figure 1). IXP Scrubber is a Machine Learning (ML) based system designed for detecting and filtering DDoS traffic at the core of...
the Internet at scale. It is suitable for large traffic hubs like Internet Exchange Points (IXPs), transferring traffic of anywhere up to thousands of Autonomous Systems (ASes). These traffic hubs see large amounts and varieties of DDoS traffic on a daily basis [37]. IXP Scrubber is continuously self-learning based on neighboring ASes’ input. It utilizes BGP signals to filter traffic on certain routes (blackholing traffic [36]) to collect DDoS samples and learns their properties. Thus, IXP Scrubber can learn about attack vectors without intervention from operators and can do so on unprecedented amounts of training data. At the same time, the IXP Scrubber ML algorithm is designed to be controllable and locally explainable for operators, which addresses a major issue for practical deployment.

In greater detail, our contributions are as follows.

- We design a method to semi-automatically generate arbitrarily large amounts of labeled DDoS training data from IXPs’ sampled packet traces. It is based on blackholing announcements, indicating unwanted traffic. Blackholed traffic contains both benign and malicious traffic and is unbalanced relative to non-blackholed, thus not directly usable for ML modeling. For the first time, we show that blackholed traffic represents a rich dataset once balanced and filtered.
- We construct a two-step IXP Scrubber machine learning approach to identify DDoS attacks at the core. Our approach is interpretable and controllable by network operators. Moreover, our approach abstracts local knowledge from the classifier to enable model transfer among different vantage points.
- We evaluate IXP Scrubber at a previously unseen scale at 5 IXPs over three months to two years. We provide insights on model drift, re-training frequencies, the transferability of IXP Scrubber models between geographically diverse vantage points, and demonstrate how the IXP Scrubber can learn new DDoS vectors without the IXP operators’ intervention.

Scope, Threat Model, and Data Limitations. IXP Scrubber learns from sampled packet headers of blackholing traffic at IXPs and aims at detecting any DDoS attack vectors that cause characteristic signatures in terms of L2-4 headers appearing in blackholing data (i.e., attacks that were labeled by some networks as unwanted). As we will show, this encompasses mainly reflection and amplification attacks (see § 4.2). While other types of attacks (e.g., TCP middlebox attacks [20], pulse wave attacks [40]) are likely detectable with our approach, they are not in the scope of this work.

2 RELATED WORK

A plethora of machine learning based methods for DDoS detection and mitigation appeared in the last decade (see [16] for a comprehensive survey). We limit our review of the state-of-the-art to three criteria: datasets, models/explainability, DDoS mitigation at the core.

Dataset size. Dataset properties impact the performance of ML-algorithms. Recent works [25, 34, 58, 61] have mostly used artificial and small-sized sets, e.g., the public datasets shown in Table 1. These evaluation datasets are no longer than a few weeks, whereas our datasets generated from IXPs’ blackholing traffic span up to two years. Other works resort to artificially generated data [27, 49, 51], either from simulating attacks or supersampling shorter existing datasets [54, 60]. The availability of labeled DDoS datasets that are of sufficient size still remains to be a challenge. In the absence of a large enough public dataset, we base our work on longitudinal data obtained from an online analysis at IXPs.

ML Models. A large set of off-the-shelf ML models were used to detect DDoS (e.g., [25, 26, 31, 34, 58, 61]). [16] reviews > 120 such models, including their classification performance and used datasets. Related work shows a comparable classification performance to ours (see § 6.1), but on less diverse and smaller datasets. We show that the chosen model is less relevant than i) training data quality and ii) the ML pipeline itself, consisting of data pre-processing, clustering, and feature encoding. Showing a novel way to generate training data and the pipeline is our main contribution.

Model Explainability. Operators are reluctant to deploy blackbox solutions that make decisions that cannot be explained. In [18], a lack of interpretability of ML models for networking experts is identified as an open problem (with DDoS as a use case). Explainable AI (XAI) aims at making ML systems more understandable to humans [32, 48]—an open challenge in ML research—, starting from global explainability (i.e., explaining the model) to local explainability (i.e., explaining model decisions for certain inputs). This work aims at local explainability, a useful property for typical troubleshooting processes at network providers (i.e., “debugging” a model’s decision upon customer request).

DDoS mitigation at the Internet’s core. Recent works have measured DDoS at large-scale infrastructures such as IXPs and Internet Service Providers (ISP) [19, 28, 37, 38, 45], in some cases through the lens of blackholing traffic [19, 28] or high volume DDoS events [38]. The authors of [19] investigate common blackholing practices and quantify the filtering issues of blackholing at IXPs as discussed in § 3. A recent work [37] proposed a heuristic to identify high traffic rate DDoS attacks, which was later used [55] to study the potential for joint DDoS filtering at IXPs. The latter work relies on a simple traffic thresholding approach to identify volumetric attacks, thus achieving high precision but low recall by ignoring low-volume attacks. Our work is the first work providing a precise and locally explainable ML-driven DDoS identification approach that is directly applicable to IXPs. It is not based on heuristics or limited to high volume attacks and aims to broadly classify DDoS.

Table 1: Related work dataset overview.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Year</th>
<th>Size</th>
<th>Time span</th>
<th>Used in:</th>
</tr>
</thead>
<tbody>
<tr>
<td>KDD99 [1]</td>
<td>1999</td>
<td>1.2 GB</td>
<td>9 weeks</td>
<td>[61]</td>
</tr>
<tr>
<td>ISCX2012 [8]</td>
<td>2012</td>
<td>84.42 GB</td>
<td>7 days</td>
<td>[58]</td>
</tr>
<tr>
<td>CICIDS2017 [50]</td>
<td>2017</td>
<td>51.1 GB</td>
<td>5 days</td>
<td>[26, 31]</td>
</tr>
<tr>
<td>Our aggregated dataset</td>
<td>2019-2021</td>
<td>3 month up to 2 years of online</td>
<td>from 5 IXPs (see § 4.1)</td>
<td></td>
</tr>
</tbody>
</table>

We now describe how we obtain arbitrarily large amounts of training data from blackholing data—a data source that is noisy, unbalanced, and that has never been used this way.

Blackholing as a crowdsourced labeling source. Blackholing is a standardized operational practice [36] enabling network operators to signal neighboring networks (routers) to drop traffic directed
to the announced IP prefix. Triggering a blackhole announcement may happen \( i \) manually in a router’s BGP configuration, or \( ii \) automatically by traffic analysis software or DDoS mitigation appliances. Thus, blackholing is a strong and explicit signal by network operators that enables the labeling of IP traffic as unwanted. Yet blackholed IPs can also receive benign traffic [19, 29] and thus blackholing does not lead to clean labels (i.e., blackholed traffic is not entirely malicious). However, as we will show later on, the vast majority of blackholing traffic is DDoS (§ 4.2) and is more prevalent in the data collection. Hence, we need to employ a sampling approach that maintains the mentioned traffic properties of the blackholed traffic. This gives us a balanced dataset for training ML models with a roughly 50:50 share of benign traffic and blackholed traffic. We validate the balancing procedure in § 4.2.

### Security considerations

For a discussion on possible attacks on our labeling and learning approach, see Appendix E.

## 4 DATASETS FROM FIVE IXPS

Next, we present the datasets processed for this work. In particular, we used two types of datasets: \( i \) the ML training set predominantly used for training of ML models, and \( ii \) the self-attack set predominantly used for validation.

### 4.1 Dataset Overview

**ML training set.** We partnered with five IXPs providing sampled flow and BGP blackholing data. The IXPs are located in central Europe (IXP-CE1, IXP-CE2), southern Europe (IXP-SE), the east coast (IXP-US1) and in the south (IXP-US2) of the United States. We processed 3 months' of sampled flow data (23/07/2021 to 23/10/2021) for all IXPs except IXP-SE (24 months; 23/10/2019 to 23/10/2021).

The IXPs show a large variance in terms of connected ASes and traffic peak. We show the network- and traffic-level details of each IXP in Table 2. The span of ASes and peak traffic shows that the data represents a broad spectrum of very small to very large IXPs from Europe and the USA. We thus argue that our dataset is representative for the typical IXPs found in practice. The data reduction induced by balancing (rightmost column) is at least 99.6% with > 225 million flow records remaining. All IXPs are operated by the same entity and use the same generic blackholing service that this entity offers to the members of the IXPs. Thus, any differences observed between the different IXPs’ datasets are due to differences in the traffic and not due to artifacts of data collection (e.g., different management policies or operational procedures).

**Self-attack set (SAS).** To validate our approach, we obtain a second dataset of labeled ground truth attack data. One IXP provided us with flow data of self-initiated, controlled DDoS attacks. It was

---

### Figure 2: Blackholing is implemented between IXP members, i.e., member networks are responsible for dropping traffic. Members not accepting blackholing routes send unfiltered traffic visible at the IXP.

<table>
<thead>
<tr>
<th>Member AS A</th>
<th>Member AS C (accepting blackholing routes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Control plane, blackholing route announcement via BGP</td>
<td>DDoS traffic</td>
</tr>
<tr>
<td>Member AS B (accepting blackholing routes)</td>
<td></td>
</tr>
</tbody>
</table>

\[\text{DDoS traffic marked for dropping via BGP, but traverses IXP unfiltered.}\]
captured over 9 days in spring 2021 at IXP-CE1 with a setup designed for DDoS monitoring for the local authorities. During the self-attacks, a total of around 5 TB raw data was transmitted, resulting in about 338K sampled flows. We balanced the SAS similar to the remaining datasets with benign data from the same time frame, which doubles the records after balancing in Table 2.

The SAS is close to what can be expected in a live setting and collected using a different method than sampling from blackholing data. Thus, the SAS is useful to reduce the potential for bias. In particular, this covers i) sampling bias in the training data introduced by the sampling method (i.e., hidden correlations with the blackholing label [41]) and ii) inductive bias (i.e., underspecified models [24]). Both types of bias are expected to lead to a noticeable loss of classification performance when training ML algorithms on the ML set and cross validating them on the SAS set; this is not the case, we demonstrate a comparable performance in § 6.1.

4.2 Dataset Validation

Quality of balancing in ML training set. We begin by validating our balancing approach that ensures an equal share of benign and blackholed traffic for all IXPs and the self-attack dataset is shown in Table 2. The number of flows is balanced in all datasets at around 50% with a maximum deviation of 5% for IXP-SE. We further show the number of flows per unique IP per bucket for both traffic classes in Figure 3c. As expected, both classes are clearly correlated (Pearson’s $r = 0.77$ at $p < 0.01$). This verifies our approach generates the balanced data sets needed for training. A positive side effect of balancing is a data reduction of at least 99.6% (see Table 2, rightmost column).

Service distribution. Figure 4a shows the share of well-known DDoS ports across three classes: benign and blackholing data of the ML training set (across all IXPs) and the SAS. Recall the SAS acts as a baseline containing DDoS traffic only. While the benign class contains $\sim 7.5\%$ of traffic from well-known DDoS ports such as NTP, SNMP, or LDAP, the blackhole class contains more than $\sim 87.5\%$ of traffic from well-known DDoS ports. The blackholing class and self-attack class contain an order of magnitude more UDP fragments than the benign class. The service distribution of the blackholing class is close to the self-attack class, thus exhibiting a high share of DDoS, but is not purely DDoS. This is expected, as blackholing is an IP-based filter mechanism used to block entire destination IPs. Attacked IPs typically receive both benign and attack traffic [29]. In case of an attack, benign and DDoS traffic is blackholed. Thus, we introduce a pre-filtering step (§ 5.1) in our approach.

Packet size characteristics. Moreover, we validate our data by comparing the packet sizes of well-known DDoS ports of the blackholing and self-attack class (Figure 4b). Many DDoS vectors produce characteristic packet sizes (e.g., NTP DDoS commonly uses 500 bytes monlist replies [38]). We find similar packet sizes for all DDoS vectors except WS-Discovery, which is hardly present in the blackholing class. This strongly indicates that the bulk of the traffic captured by blackholing is indeed DDoS traffic.

Takeaway. The balancing procedure creates a well-balanced dataset and maintains privacy by reducing the raw data by more than 99.6%. The validation of the data indicates that blackholing data is a useful source of DDoS data samples. The characteristics are similar to the baseline self-attack data. Nevertheless, blackholed data contains up to 12.5% of possibly benign data, which has to be considered for ML.

Table 2: Dataset overview (*=sum recorded online, unbalanced part of data was discarded early).
We immediately obfuscate sensitive data, i.e., IP addresses and MAC addresses are hashed with a secret salt before storage and analysis. Moreover, the data is sampled, aggregated on a flow-level and does not contain payload information. Capturing the data is compliant with the local legal regulations. The resulting attack data is not privacy-critical. The attacked systems were hosted by the IXP within a dedicated AS and IP space, <7 Gbps, duration <5 minutes).

The resulting attack data is not privacy-critical. The attacked systems were hosted by the IXP within a dedicated AS and IP space, the attacking systems can be found by scanning the IP space (e.g., DNS servers) and are contained in public datasets (e.g., Censys, Rapid 7). The experiments were tightly controlled to immediately stop them in case IXP members experience side effects, i.e., by immediately withdrawing the dedicated IP space. There were no complaints by IXP members during the experiments.

4.3 Ethical Considerations
We carefully take a number of steps to ensure all data processed was recorded and is used in compliance with ethical standards.

Traffic data for ML-training set. The data in the ML training set was recorded online, i.e., flow records not chosen by the balancing procedure were discarded after balancing, thus immediately reducing the recorded data by more than 99.6%. Moreover, the data is sampled, aggregated on a flow-level and does not contain payload information. Capturing the data is compliant with the local legal regulations. We immediately obfuscate sensitive data, i.e., IP addresses and MAC addresses are hashed with a secret salt before storage and analysis. Traffic data for self-attack set. To obtain the ground truth traffic data containing DDoS attacks for validation, one IXP provided us with traffic data from previous self-attacks using DDoS-for-hire services. The experimental setup to obtain this data was designed in collaboration with a government agency. Contracting a DDoS-for-hire service is a sensitive matter. Thus, the data was obtained by purchasing the smallest service package (15$), which also limits possible side effects (volume <7 Gbps, duration <5 minutes).

5.1 Step 1: Rule Tagging
The goal of the rule tagging step is to automatically compile a small list of tagging rules, which are interpretable for humans, that tag each flow as benign or malicious. The rule tags are preserved through the aggregation step for two reasons: i) they represent filter definitions that can be applied directly to the hardware as an Access Control List (ACL) filtering rule later on, and ii) they are helpful to explain the per-target IP classification in Step 2 by explaining problematic header combinations in the traffic. Each rule can be reviewed and manually enabled/disabled by network operators, thereby making this step fully interpretable and controllable. Moreover, by repeatedly applying this step over time to new data, a growing set of rule tags can be accumulated. With our rule minimization approach, we ensure that the list can be curated manually in a reasonable time frame.

5.1.1 Association Rule Mining. For generating tagging rule candidates, we use association rule mining (ARM) [14], which is a well-known data mining technique originating from ecommerce recommender systems (e.g., ‘customers buying milk also buy eggs’). ARM can learn association rules on structured data in the form of $A \rightarrow C$, where $A$ is a set of items called the antecedent and $C$ is a

5 ML-DESIGN OF THE IXP SCRUBBER
The goal of the IXP Scrubber is to identify DDoS attacks at IXPs using flow-level traffic data. To tackle this challenge, IXP Scrubber uses a two-step ML approach shown in Figure 5. As a result, IXP Scrubber generates filters (ACLs) to classify DDoS traffic, which can be used for dropping, shaping, monitoring or re-routing.

Step 1 introduces rule tagging to tag individual flows as benign or malicious (macroscopic level). We automatically generate a few promising rule tags out of large volumes of balanced training data. These rule tags are comparable to firewall rules and are easily interpretable by network operators. Operators can validate them in a web interface supporting the selection process, as shown in Figure 6. This way, domain knowledge of network operators is included in the model, as they can review and manually enable/disable each filtering rule—a practical and feasible approach given our minimization approach that generates minimal filtering rule sets. As we will show, in addition to being inherently interpretable and controllable, this approach also achieves high accuracy.

Step 2 aggregates information from individual flows to a per-target IP perspective (macroscopic level). To do so, we first derive features for learning from the flow headers in an aggregation step. Afterwards, we apply five common ML models to the derived features. These models are less intuitive to understand than the tagging rules, depending on the algorithm (see challenges in XAI, § 2). Yet local explainability methods can be applied to these models because the use of Weight of Evidence encoding (introduced in § 5.2) and tagging rules preserve the meaning carried by the models’ features.
Algorithm 1 minimize association rules

1: function MinimizeAssociationRules(\(L_c, L_s\), 
2: \(R = [(A_0, c_0, s_0), (A_1, c_1, s_1), \ldots, (A_n, c_n, s_n)]\))
3: while true do
4: \(D \leftarrow \emptyset\)
5: for \(i = 0\) to \(n\) do
6: \(\triangleright \) rule indices to delete
7: for \(j = 0\) to \(n\) do
8: \(\triangleright \) pair-wise iteration of rules in \(R\)
9: if \(i \neq j\) then
10: if \(A_i \subseteq A_j\) then
11: \(\triangleright \) rule i’s antecedent is in j’s
12: if \((c_i - c_j < L_c) \land (s_i - s_j < L_s)\) then
13: \(D \leftarrow \{i\}\)
14: \(\triangleright \) remove i; limited loss in conf./supp.
15: if \(|D| = 0\) then
16: break
17: \(\triangleright \) no more dispensable rules in \(R\)
18: for \(k = 0\) to \(n\) do
19: \(\triangleright \) remove rules from \(R\)
20: if \(k \in D\) then
21: \(R \leftarrow R - \{R[k]\}\)
22: return \(R\)

The test subjects generated rule sets of high quality in a short time, suggesting the approach to be applicable in network operation. On average, the subjects correctly drop 76.73% of the ground truth DDoS traffic, while only dropping 0.43% of the benign data. For curating the 38 rules, they only needed 6.62 minutes on average. While the study is small-scale, it has been conducted by domain experts and shows the feasibility of our approach.

5.2 Step 2: Aggregation from Flows to Targets and Classification

In step 2, we aggregate from individual flows (microscopic perspective in step 1) to per-target IP profiles (macroscopic perspective).

Step 1 provides a classification of individual flows only, ignoring any structural information in traffic aggregates beyond a single flow. The macroscopic perspective uses supervised machine learning techniques to learn expected and anomalous traffic profiles per target IP by aggregating flow-level features into a holistic picture.
5.2.1 Feature Construction. To create profiles that infer whether a target is under attack, we first need to derive meaningful features that aggregate flow-level information by target IP. To do so, we aggregate multiple flows into one record to summarize all traffic sent to a target. We next describe our feature and label construction.

Binning and grouping. The aggregation process is depicted in Figure 7. First, traffic flows of the balanced dataset are grouped. Similar to the dataset balancing, flows are separated into time bins. We use the same time bin resolution of one minute as in the balancing procedure. The flows in each time bin are grouped according to the target IP address. Afterwards, all flows are grouped by time range and target IP and are aggregated into a single dataset record.

Ranking categoricals. The categorical flow properties of a set of flows \( C = \{ \text{source IPs, source port, destination port, source MAC address, transport protocol} \} \) are ranked based on the non-categorical flow metrics \( M = \{ \text{mean packet size, sum of bytes, sum of packets} \} \) with a resolution of \( r = 5 \) ranks, e.g., the top 5 source ports by bytes sent to the target (see Figure 7 for an example). This results in \( |M| \times |C| \) rankings with 2 + \( r \) columns each, as we store the categoricals and the aggregated metric per ranking. In our case, the aggregation generates 150 feature columns (excluding the <time bin, target IP> index columns and the label column) while reducing the ML training dataset from all vantage points to 1.2 million records. Notably, we ignore any features related to the network announcing the blackhole (i.e., no target IP's or ASN/path information) to avoid bias (i.e., learning who is blackholing traffic instead of traffic properties). The way we aggregate data by using ranks for the final classification of attacked systems deliberately generates redundant/correlated feature columns in the aggregated dataset to have a broad base of features for feature selection. Appendix B shows the correlation among columns in the aggregated data.

Labels. The last step generates the labels required for learning. If we find at least one of the flows for a certain destination IP marked as blackholed, the corresponding aggregated record is likewise marked as a blackhole and thus DDoS.

5.2.2 Machine Learning Classification. The goal of this step is to classify traffic towards a target IP as malicious or benign by using a supervised machine learning approach. This classification is based on the feature set that aggregates individual flows to a macroscopic per-target IP perspective. While only a single machine learning model is required for this step, we implement and evaluate a broad set of common classification models to identify the one that achieves the best classification performance and requires the least CPU cycles for prediction. We follow a generic 3-step methodology to implement and optimize all classifiers.

1. Data preparation. We carefully review the assumptions regarding input data for each algorithm and implement a data preprocessing pipeline that performs the required transformations on the data before classification. This can include normalization, encoding of null values, and the like.

2. Hyperparameter optimization. We carefully review hyperparameters available to tune the classification performance. For each algorithm, we define a grid of hyperparameters. The grid is tested for parameter combinations providing the best performance on our overall dataset using a 3-fold cross-validation.

3. Feature elimination. Recall that the aggregation approach deliberately introduces correlated features to have a broad base of features for feature selection (also discussed in Appendix B). We reduce this excess dimensionality with common techniques like recursive feature elimination or PCA.

Weight of evidence encoding. Beyond data normalization as common feature pre-processing, a key step in our pipeline is to encode all categorical variables (IPs, transport ports, MAC addresses of IXP members, etc.) as Weight of Evidence (WoE). The WoE concept originates from the context of financial risk assessment [56]. The idea of WoE is to map each possible value \( x_i \) of a categorical feature to \( \text{WoE}(x_i) = \ln \frac{P(X=x_i|y=1)}{P(X=x_i|y=0)} \), where \( y \) is the blackhole label. That is, \( x \) variables (e.g., IPs) are transformed into bins of similar WoE values based on the similarity of the distribution in the blackhole labels. In the case of risk management, \( x_i \) may be the name of a debtor and \( y \) whether the debtor defaulted or not. A

\[ \text{WoE}(x_i) = \ln \frac{P(X=x_i | y=1)}{P(X=x_i | y=0)} \]

\[ \text{WoE}(x_i) = \ln \frac{P(X=x_i | y=1)}{P(X=x_i | y=0)} \]

\[ \text{WoE}(x_i) = \ln \frac{P(X=x_i | y=1)}{P(X=x_i | y=0)} \]

\[ \text{WoE}(x_i) = \ln \frac{P(X=x_i | y=1)}{P(X=x_i | y=0)} \]

\[ \text{WoE}(x_i) = \ln \frac{P(X=x_i | y=1)}{P(X=x_i | y=0)} \]
Weight of Evidence identifies problematic attack traffic in header clusters. The selection of algorithms and their respective data preprocessing is especially beneficial in our case, as WoE can leverage our knowledge. If a destination IP has received a flow matching one of the mined tagging rules, a rule tagging-based classifier (RBC) is applied instead. This would remove the need to apply rule tags to their WoE of appearing in the blackhole (or not in it). We then input WoE($x_i$) to each ML classifier, instead of using $x_i$ directly.

Benefits of WoE encoding. Using WoE encodings has four benefits. (1) As we will show in §6.6, WoE is a useful tool to make model behavior locally explainable. (2) In comparison to methods like one-hot encoding which encodes each possible value of the categorical into a separate binary feature column, WoE is very memory efficient, as it only requires storing the mapping of possible categorical values to their WoE. (3) WoE encoding incorporates a long-term memory on suspicious transport ports, reflector IPs or DDoS prone IXP member ports without the need to have the classifier looking at past records during classification, which would lead to extended training times and more complicated ML architectures. This is especially beneficial in our case, as WoE can leverage our long-term data. (4) WoE encoding encapsulates local knowledge independent of the model, e.g., a local, nearly disjoint set of DDoS reflection hosts is learned at each IXP. As we will show in §6.4, this enables the exchange of trained models between IXPs.

Classifiers. We test five different classifiers: XGBoost (XGB) [23], decision tree (DT), neural networks (NN), linear support vector machine (LSVM), and multinomial/complement/gaussian/bernoulli naive Bayes (NB-M/NB-C/NB-G/NB-B). All fitting of data (including all preprocessing, esp. WoE encoding) is done with a disjoint training and test set. This prevents data leakage distorting results. The choice of algorithms and their respective data preprocessing pipelines are shown in Figure 8, and the results of the hyperparameter optimization and feature reduction are listed in Appendix C. Please note that we classify attacked systems and apply tagging rules as filters afterwards. It might be possible to use multiclass classification to predict the tagging rules and use them as ACLs directly instead. This would remove the need to apply rule tags to flows for prediction, but might lead to a less interpretable model: tagging rules are derived from the raw data, whereas predicted tagging rules would be generated by the ML-model.

Baseline classifiers. In addition to the classifiers mentioned above, we use two baseline classifiers for a comparison. One is a dummy classifier (DC), which randomly guesses a label with equal probability, i.e., the worst conceivable classifier. The second baseline classifier is the rule tagging-based classifier (RBC). The RBC performs a prediction based on the rule tags discussed in Section 5.1. If a destination IP has received a flow matching one of the mined tagging rules, the RBC predicts it as DDoS, otherwise as benign traffic. The RBC constitutes a baseline for the achievable classification performance when relying on association rule mining only.

Local Explainability. Figure 9 shows an example of how misclassifications can be debugged independently of the actual classifier by investigating rule tags and WoE encodings of the pipeline. The example shows a false negative classification. The annotated rule indicates an NTP amplification attack and most of the Weight of Evidence encoded features also point to an attack, but the negative value of the IP source address causes the classifier to mislabel the attack as benign traffic. A viable resolution for the operator would be to blacklist the source IP, e.g., by artificially adding a high WoE to the respective sending host.

6 IXP SCRUBBER EVALUATION

We have prototypically deployed IXP Scrubber at five commercial IXPs to evaluate its prediction performance, temporal stability (how often do models need to be trained?), geographic stability (can models be trained at one IXP and used at other locations?), and local explainability. In these deployments, we train all models with the input data (§4.1) and then evaluate their actions—of course, without actually dropping or filtering traffic or impacting IXP operation.

6.1 ML Model Classification Performance

We begin by evaluating the performance of all models. To do so, we merge traces from all five IXPs and use 2/3 of the ML training set for model training and 1/3 for evaluation.

Model performance. We report the performance for the classical model performance indicators, i.e., true positives/true positive rate ($tp/tpr$), true negatives/true negative rate ($tn/tnr$), false positives/false positive rate ($fp/fpr$), and false negatives/false negative rate ($fn/fnr$) in Table 3. It contains all models except NB-C, NB-M and NB-B due to unacceptable performance ($tnr$ below 0.90, see Appendix D). We also show the harmonic mean of precision and recall expressed as $F_1$ score, where $F_1 = \frac{2tp}{2tp+fp+fn}$.

Classifiers. We compute the weighted ratio $F_\beta$, which we use in the remainder of this work, expressed as $F_\beta = \frac{(1+\beta^2)\cdot tp}{(1+\beta^2)\cdot tp + \beta^2 \cdot fp + \beta \cdot fn}$ for $\beta = 0.5$. Last, we present a performance perspective measured in mega clock cycles per prediction (mcc) that we obtain directly from the CPU during prediction (averaged over 30 runs). Good model performance is indicated by high $F$-scores and $tnr/tpr$ values, low $fnr/fpr$ values, and a low mcc value.

We observe that high prediction performance at low false positive rates is possible with all tested ML models. While the choice seems arbitrary, one should keep in mind that IXP Scrubber has to classify large quantities of IPs per day at IXPs, thus even small relative differences in performance may cause large absolute numbers in false positives/false negatives. The best performance is obtained for the XGB model, which achieves the highest $F_{\beta=0.5}$ score and the lowest false negative rate, the third lowest false positive rate and the third lowest mcc when predicting. For any practical application, the XGB model is thus the recommended model.
We therefore do not further differentiate performance by vector.

Table 3: Classification results (except the last column) are based on a random 2/3 train set 1/3 test set split on the ML training set (all IXPs). The last column applies models learned on 2/3 of the ML training set (all IXPs) to the self-attack set (SAS).

| Model | $F_{\beta=0.5}$ | $F_1$ | mcc | tnr | fnr | tpr | fpr | UDP | DNS | NTP | SNMP | LDAP | SSDP | Apple | $F_{\beta=0.5}$ (all on SAS) |
|-------|-----------------|-------|-----|-----|-----|-----|-----|-----|-----|-----|------|------|------|------|------|-------------------|
| XGB   | 0.998 | 0.988 | 0.015 | 0.988 | 0.012 | 0.988 | 0.012 | 0.994 | 0.994 | 0.993 | 0.996 | 0.993 | 0.971 | 0.993 | 0.961 |
| NN    | 0.985 | 0.976 | 0.043 | 0.990 | 0.039 | 0.961 | 0.101 | 0.994 | 0.993 | 0.990 | 0.996 | 0.991 | 0.959 | 0.991 | 0.631 |
| LSVM  | 0.976 | 0.973 | 0.001 | 0.981 | 0.035 | 0.965 | 0.019 | 0.994 | 0.993 | 0.990 | 0.996 | 0.991 | 0.958 | 0.990 | 0.963 |
| NB-G  | 0.978 | 0.959 | 0.022 | 0.991 | 0.071 | 0.929 | 0.069 | 0.993 | 0.993 | 0.990 | 0.996 | 0.991 | 0.959 | 0.991 | 0.425 |
| DT    | 0.965 | 0.950 | 0.004 | 0.974 | 0.072 | 0.928 | 0.026 | 0.991 | 0.991 | 0.987 | 0.994 | 0.990 | 0.963 | 0.991 | 0.954 |
| RBC   | -     | -     | -    | -    | -    | -    | -    | -    | -    | -    | -    | -    | -    | -    | 0.917 |
| DUM   | 0.511 | 0.506 | -    | 0.501 | 0.498 | 0.500 | 0.499 | -    | -    | -    | -    | -    | -    | -    | 0.530 |

Figure 10: XGB features with highest avg. gain for all splits (notation: categorical/metric/rank, see Figure 7).

Model performance per attack vector. Model performance might differ by attack vector (i.e., the used amplification protocol). To evaluate this aspect, we show the $F_{\beta=0.5}$ score for each model for the top 7 attack vectors in our dataset in Table 3. Irrespective of the attack vector, we do not observe noticeable performance differences and all models perform equally well for all shown attack vectors. We therefore do not further differentiate performance by vector.

Ground truth evaluation. In a second step, we evaluate the models trained on ML training dataset on the self-attack set (SAS). Recall the self-attack data only contains DDoS attacks, was recorded with a different method and is close to production data; thus the set is useful to reduce the possibility of bias when generalizing to a real-world setting (see §4.1). In case of bias, we expect a considerable loss in classification performance. We show the model performance as $F_{\beta=0.5}$ score in Table 3 in the rightmost column. LSVM reaches the highest $F_{\beta=0.5}$ score (0.963) with XGB being second (0.961) which is comparable to the performance on the remaining data.

Moreover, only using the operator-driven rule mining approach (RBC) already yields a high prediction performance — without any further machine learning model applied. Here we observe an $F_{\beta} = 0.5$ score of 0.917 with a tpr/tnr of 0.847/0.938 and fpr/fnr of 0.153/0.0616 (not shown in table); these results show that combining an interpretable approach (rule mining) with a more precise but less interpretable approach like XGB can produce local explainability efforts. We explore the overlap between the RBC and XGB in §6.6. In contrast to the dummy classifier that performs a coin toss for each flow (and thus arrives at tnr=tpr=fpr=fnr=0.5), this is a substantial classification performance that is only slightly improved by the machine learning models in step 2. Note that we can only validate RBC on the self-attack dataset, because the rules were mined on the remaining data. Validating RBC on the same dataset would introduce data leakage.

Takeaway. The evaluation of the performance of different models shows that XGB achieves the best results for IXP Scrubber on the whole ML training set as well as single attack vectors at reasonable cost for prediction. XGB is thus the recommended model for any practical application. Notably, the fully operator interpretable and controllable rule mining approach performs remarkably well, reaching scores not far below XGB. This shows the benefit of our design choice that combines a sophisticated learning algorithm (XGB) with an interpretable approach (rule mining).

6.2 Features
We briefly discuss the features used by the selected XGB model. Figure 10 shows 10 of the model features ranked by their average gain, a measure of the average loss reduction when using a feature for splitting the data in XGB. All features relate to relevant properties of DDoS attack vectors known from literature (e.g., [25, 26, 31, 34, 38, 58, 61]). The features encode temporarily stable properties of the DDoS attack vectors, e.g., the abused protocols, ports, or packet sizes as well as potentially drifting features such as the source IP (possible referrer). Since referrer IPs can change over time, we evaluate the temporal model drift in §6.3 and specifically focus on referrer IPs in Figure 12. We show that temporal stability is no practical problem with modest re-training. We remark that all shown features are automatically identified by the algorithm without a-priori knowledge and thus will also adapt to new attack vectors as we show in Figure 13 later on.

6.3 Temporal Model Drift
Since traffic patterns change over time, the temporal stability of the trained models is of relevance to any practical application. This aspect involves two basic questions. First, the relevance of the length of training period, i.e., for how long should a model be trained before it can be used? Second, the model drift over time, i.e., how often does a model need to be re-trained to maintain performance? We argue that both aspects are crucial for any practical deployment. Notably, the fully operator interpretable and controllable rule mining approach performs remarkably well, reaching scores not far below XGB. This shows the benefit of our design choice that combines a sophisticated learning algorithm (XGB) with an interpretable approach (rule mining).
IOP-US1, IXP-CE1 and a model learned on all five IXPs (ALL) in Figure 11a. We perform a one-shot training using an interval of one i) day, ii) week, and iii) month at the beginning of the data while predicting and scoring the performance on each of all remaining days. The model learned on the first day becomes quickly outdated with an $F_{\beta=0.5}$ dropping below 0.90 while the model learned on the first month always reaches a $F_{\beta=0.5}$ above 0.90 at a median performance of 0.989 at IXP-US1. The longer we train, the better the results, regardless of the training dataset size. Longer one-shot training periods help to reduce outliers in classification performance. We observe this for all IXPs (not shown).

How often do we need to re-train? Once trained, a model may become outdated since traffic patterns change over time (e.g., new attack vectors or new DDoS reflection hosts). We thus evaluate the effect of re-training frequency on model performance and show the performance of the XGB model when trained daily on a sliding window covering the past i) day, ii) week, or iii) month in Figure 11b. There is a clear overall increase of the performance compared to one-shot learning in Figure 11a. While increasing the size of the sliding window does not increase median performance too much, it helps to reduce outliers. The best performance is achieved with daily re-training on the last month for the XGB model. This approach results in a median $F_{\beta=0.5}$ of 0.993 (IXP-US1) and 0.978 (IXP-CE1) while never dropping below 0.95. Note that XGB can in principle learn incrementally, but this may be detrimental in the presence of temporarily shifting features such as reflector IP; these require forgetting information when, e.g., IP’s are repurposed in a legitimate way. Thus, re-training is (currently) the better option.

Takeaway. The more time passes between learning and predicting, the lower the performance, i.e., a trained model becomes outdated quickly. This can be fixed easily, either by continuous re-training or by training over longer time periods (e.g., one week or one month), where re-training with a sliding window of one month is the recommended method according to our results.

6.4 Geographic Model Drift

Given that training is complex and that quality training data is hard to obtain, it might be sufficient to train a model once and then share it. Given the lack of a rich enough training set, this aspect has not yet been investigated. We study this question by training all ML models at each IXP location once and then apply each to all other locations. We show $F_{\beta=0.5}$ for the best performing model as a heatmap in Figure 12 (left). For legibility, we cut off the color bar at $F_{\beta=0.5} = 0.95$.

The results show that XGB can outperform any other algorithm when either training and testing is done with data from the same IXP (diagonal values) or XGB is trained on all available data and tested on the data of any IXP (top row). In these cases, XGB can reach a performance close to a perfect score of 1.0. However, when transferring models between IXPs, performance can be seriously harmed and other algorithms can outperform XGB with no clear winner. Please note the locations are sorted by increasing dataset size with IXP-CE1 being the largest. With the exception of IXP-US2 and IXP-CE2, the models trained on the largest IXP-CE1 can be transferred to other locations with decent performance.

Recall we stated in § 5.2.2 that WoE encoding is useful to separate local information from the classifier. We will substantiate this claim in the following by i) investigating the overlap of WoE encodings between different IXPs and ii) evaluating a transfer of only the classifier between IXPs while keeping the local WoE encoding local. Figure 12 (middle) analyses the overlap of source IP’s appearing in the WoE encoding. In order to restrict this analysis to the knowledge of reflectors, we only consider source IP’s with a WoE $> 2.71$ times more likely to send traffic to a blackhole than not. The overlap analysis plot indicates a very low overlap of DDoS reflection hosts among IXPs. Consequently, in the case of a model transfer across geographies, the ML-models cannot rely on knowing the source IP’s anymore and need to utilize other location independent features. Note that knowledge on reflection
hosts is only one of multiple features exhibiting locality. We have done a similar analysis for transport ports, which have an order of magnitude more overlap; nevertheless the analysis indicates that not all DDoS vectors are visible at all IXPs.

To test the hypothesis that WoE encoding abstracts local knowledge from the classifier, we repeat the transfer of models between IXPs, but this time we only transfer the actual classifier while keeping the local WoE encoding (see Figure 12, right plot). The classification performance increases to more than 98% with XGB being the winning model in almost all cases, except for transfers between very small IXPs like IXP-CE2 and IXP-US2. This shows that i) WoE encoding enables efficient abstraction from local knowledge while ii) it is nearly irrelevant where the classifier on top is learning, but learning on more (WoE encoded) data is helpful (e.g. IXP-CE1 compared to IXP-CE2).

Takeaway. When keeping local information in WoE scores, models are transferable between IXPs with only a very minimal performance penalty. Slightly better performance can be obtained when joining all IXPs to generate a joint XGB model.

### 6.5 Learning new DDoS Vectors

This section demonstrates how IXP Scrubber picks up new attack vectors without intervention by IX operators using the two year dataset of IXP-SE. Figure 13 shows how the WoE and classification performance varies over time for individual attack vectors. We present the WoE of the SNMP, SSDP, memcached DDoS vectors identified by their respective protocol and transport ports. Once these new attack vectors are blackholed by IXP members, their WoE starts to rise, as the attack vectors are predominantly found in blackholing traffic rather than benign traffic. This shows that IXP Scrubber can learn new, previously unknown DDoS attacks.

As a reference, we plot the WoE of HTTP, which has a constantly negative WoE as it is predominantly found outside the blackhole. We additionally display the $F_{\beta=0.5}$ score of XGB for each vector with incremental training in the lower part of Figure 13. The first 9 weeks of the dataset are used to warm up the algorithm and are thus omitted. For the SNMP and SSDP attack vectors, we trained XGB 30 times in the period from week 2020-00 to week 2020-30, using an additional week of data in each iteration. For memcached, we did the same in the period from week 2020-20 to week 2020-50. We validated each of the trained models using a test set consisting of data from week 2020-51 to week 2021-42. It can be seen that as the WoE of an attack vector increases, so does the classification performance of XGB. In particular, for SSDP, this is illustrated by two increases in WoE and $F_{\beta=0.5}$ score at successive points.

Takeaway. IXP Scrubber can pick up new DDoS vectors without intervention of IX operators and converges to high classification performance the more frequently a vector is blackholed.

### 6.6 Local Explainability

One of the major contributions of the IXP Scrubber is local explainability of ML classifications. Classification decisions can be explained with two mechanisms: i) by observing WoE encodings and ii) by the mined rule tags that identify problematic header combinations and may likewise act as ACL definitions to filter DDoS traffic. Remember, rule tags are preserved during aggregation (§ 5) and can be investigated alongside each classification decision similar to the WoE encoding.

Mined rule tags. Figure 14a demonstrates the value of tagging rule mining for local explainability. Each matching tagging rule is annotated to the training set during aggregation (but not used for classification to avoid data leakage). In cases where the classification of XGB and RBC overlap, i.e., in cases where XGB classifies positively and a mined tagging rule matched the traffic, we can use the annotated tagging rules to locally explain the classification result (or use them as ACLs for actual filtering). This is the case in 70.9% of the records in all datasets. In 30% of the cases with coherent decisions, we can provide at least one rule to interpret the classification, in 50% of the cases we can provide up to 3 rules.

Note that the absence of mined rules does not mean that an attack cannot be mitigated; the operator can still use the information to rate limit traffic to attacked target IPs based on the decision of XGB.
Takeaway. WoE encoding of individual features provides strong evidence for the classification of DDoS traffic. The same is true for annotated, mined tagging rules. Both can be used to locally explain and control individual classification decisions made by the ML algorithm.

7 CONCLUSIONS

This paper tackles detecting and filtering DDoS attacks directly at the core of the Internet: at IXPs. As today’s (commercial) solutions filter at the edge, this paper proposes IXP Scrubber, an ML-based system for detecting and filtering DDoS traffic at the core and at scale. IXP Scrubber is based on a two-step ML model learning continuously and without IXPs operators’ intervention. It proposes a method to extract arbitrary large volumes of DDoS training samples from blackholing traffic—a rich data source that has never been used as input for building systems. IXP Scrubber reaches high classification quality (more than 0.98 Fβ=0.5-score on all targets and more than 0.99 Fβ=0.5-score for the largest attack vectors). With reasonable amounts of training data (one month) in a sliding window training setting, the IXP Scrubber ML model maintains a high temporal stability (median Fβ=0.5-score between 0.978 and 0.994, depending on the vantage point). We demonstrate the benefits of WoE encoding for i) making models geographically transferable without performance penalty and ii) for contributing to the models’ local explainability by showing how high WoE values of certain features can be correlated with the classification outcomes. The latter, in combination with our rule tagging approach, was shown to be able to interpret problematic packet header combinations leading to a DDoS classification of traffic.

Acknowledgements

We thank the anonymous reviewers and our shepherd Walter Willinger for their constructive comments. We further thank our colleagues for their ongoing support and our significant others for their tremendous patience. This work was funded by the German Federal Ministry of Education and Research (BMBF) grant AIDOS (grant number 16KIS0975K and 16KIS0976).
Figure 15: Association rules after minimization for different combinations of support loss $L_s$ and confidence loss $L_c$.

Figure 16: Correlation introduced by aggregation.

Appendices are supporting material that has not been peer-reviewed.

A PARAMETER SENSITIVITY STUDY RULE MINIMIZATION

This appendix discusses how to set the $L_c/L_s$ parameters for Algorithm 1 in § 5.1.1. Setting these loss parameters too high might eliminate many but also relevant rules, while setting them too low will result in many but redundant rules. Thus, we conduct a parameter sensitivity study shown in Figure 15. The figure presents the remaining amount of rules for different $L_c/L_s$ settings; the upper right quadrant shows that reducing aggressively beyond $L_c = 0.01$ and $L_s = 0.01$ does not result in a much lower amount of remaining filtering rules, but increases the likelihood on eliminating relevant rules. Consequently, we choose these settings for the experiments conducted in this work.

B CORRELATION INTRODUCED BY FLOW AGGREGATION

In step 2 of IXP Scrubber (§ 5.2), we deliberately generate redundant/correlated feature columns in the aggregated dataset to have a broad base of features to select from. These features are then reduced with feature elimination. We validate that the resulting features are indeed correlated. Figure 16a demonstrates the correlation as CDF of a Spearman correlation matrix’s values. Depending on the columns’ metrics (aggregation by packets, bytes or packet size), 20% of the columns have a correlation $> 0.7$ or $> 0.8$, respectively.

Figure 16b shows the results of a PCA of the aggregated dataset, revealing the first twenty components already explain 0.8 of the total variance in the dataset, whereas 50 components are explaining close to all variance. This shows great potential for reducing the number of input features for classifiers with a single matrix multiplication using the result of a PCA as done for NN in § 5.2.

C ML MODEL OPTIMIZATION

In any ML application, hyperparameter search represents a classical performance tuning step that chooses the optimal set of parameters for each ML algorithm. We next describe the applied tuning of the ML models in step 2 of IXP Scrubber.

We applied a grid search to determine the hyperparameters of the ML models used. The hyperparameters and their considered values are shown in Table 4. Due to the size of the dataset and the long runtime when it is fully used, we sampled 250K records from the data of all IXPs to perform the grid search. Each of the parameter combinations of a model was validated using 3-fold cross-validation. The training of a model was then repeated three times for a parameter combination where each fold was used once for validation, and the rest of the data formed the training set. The performance of model variations was determined using the mean $F_{\beta=0.5}$ score of the three folds. A detailed description of the hyperparameters can be found in the documentation of the model implementations [5, 9–11, 13].

D COMPLETE ML MODEL CLASSIFICATION RESULTS

Table 5 shows the complete classification results of all evaluated models. This represents an extended version of Table 3, in which we omitted alternative naive Bayes variants that did reach comparable performance. This includes Bernoulli (NB-B) and multinomial (NB-M) distributions, as well as the complement naive Bayes classifier (NB-C). All other performance figures are the same as in Table 3.

E IXP SCRUBBER SECURITY

DDoS has always been a game of cat-and-mouse between attackers and mitigation solutions—IXP Scrubber is no different. Thus, it makes sense to anticipate attack scenarios on the IXP Scrubber...
model itself, i.e., attempts to data poisoning to influence classification results. In other words, can the IXP Scrubber be influenced by an attacker injecting manipulated traffic? We argue that performing such an attack is challenging, resource intense, and thus unlikely.

Impacting the classification output requires the attacker to change the WoE encoding of the features. The WoE encoding of one feature shall either be changed if from positive to negative/neutral or ii) from negative/neutral to positive. The objective of i) is to hide attack traffic and of ii) to produce false positives, i.e., the IXP Scrubber classifies benign traffic as malicious. However, for both scenarios a sophisticated attacker must rent ports at IXPs and send traffic with certain patterns to his own IP space. Additionally in case of ii) the attacker has to announce blackhole announcements for his own IP space.

Depending on the encoding to be influenced, substantial amounts of traffic need to be generated. For instance, attacking HTTP(S) would require to send at least as much HTTP(S) traffic as can be found outside the blackhole—over long time frames. At large traffic hubs, this would correspond to multiple terabit of sustained attack traffic (recall that the bulk of the traffic is not blackholed). In any case, the IXP Scrubber operator can still react by setting the WoE encodings of certain feature values to a suitable constant (see § 6.6).

Table 5: Classification results (except the last column) are based on a random 2/3 train set 1/3 test set split on all dataset combined. The last column applies models learned on 2/3 of all datasets to self-attack data.

<table>
<thead>
<tr>
<th>Classifier</th>
<th>Parameter</th>
<th>Parameter Space</th>
</tr>
</thead>
<tbody>
<tr>
<td>Naive Bayes</td>
<td>var. smoothing&lt;sup&gt;1&lt;/sup&gt;</td>
<td>{10⁻⁹, 10⁻⁸, 10⁻⁷, 10⁻⁶, 10⁻⁵, 10⁻⁴, 10⁻³, 0.01, 0.1, 1}</td>
</tr>
<tr>
<td>Decision Tree</td>
<td>ccp alpha</td>
<td>{10⁻⁹, 10⁻⁷, 10⁻⁵, 0}</td>
</tr>
<tr>
<td>LSVC</td>
<td>regularization (C)</td>
<td>{10⁻⁵, 10⁻⁴, 10⁻⁳, 0.01, 0.1, 1, 10, 100, 1000, 10000}</td>
</tr>
<tr>
<td>XGBoost</td>
<td># estimators</td>
<td>[2, 4, 8, 16, 24]</td>
</tr>
<tr>
<td>Neural Network</td>
<td># PCA components</td>
<td>[25, 50, 75]</td>
</tr>
</tbody>
</table>

Table 4: Overview of the classifiers’ hyperparameter space. The selected parameters are marked in bold letters.

<table>
<thead>
<tr>
<th>Classifier</th>
<th>Parameter</th>
<th>Parameter Space</th>
</tr>
</thead>
<tbody>
<tr>
<td>Naive Bayes</td>
<td>additive smoothing&lt;sup&gt;2&lt;/sup&gt;</td>
<td>{10⁻⁹, 10⁻⁸, 10⁻⁷, 10⁻⁵, 10⁻⁵, 0.01, 0.1, 0.5, 1.0, 2.0, 10.0}</td>
</tr>
<tr>
<td>Decision Tree</td>
<td>min. impurity decrease</td>
<td>{10⁻³, 10⁻⁵}</td>
</tr>
<tr>
<td>LSVC</td>
<td>class weight</td>
<td>[primal, dual]</td>
</tr>
<tr>
<td>XGBoost</td>
<td>max. depth</td>
<td>[2, 100]</td>
</tr>
<tr>
<td>Neural Network</td>
<td># neurons hidden layer</td>
<td>[4, 8, 16, 32]</td>
</tr>
<tr>
<td>RBC</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Neural Network</td>
<td>learning rate</td>
<td>[0.01, 0.1, 0.2, 0.3]</td>
</tr>
<tr>
<td>Neural Network</td>
<td>dropout</td>
<td></td>
</tr>
</tbody>
</table>

Table 5: Classification results (except the last column) are based on a random 2/3 train set 1/3 test set split on all dataset combined. The last column applies models learned on 2/3 of all datasets to self-attack data.
with spoofed IPs. After all, the classification is not only based on the WoE of source IPs, but also on other criteria like transport ports and even more importantly the traffic volumes that are measured per source IP, transport ports, etc. (see Figure 10).

Summing up, poisoning IXP Scrubber’s training data requires the attacker to rent sufficient port capacity at the IXP and to inject substantially high volumes of traffic. This way, the security properties of IXP Scrubber follow a classical assumption that assumes that the majority of the traffic/participants are not malicious (e.g., similar to Tor where a majority of nodes needs to be malicious to compromise the system).

F SUPPLEMENTAL MATERIAL

The mined filtering rules (see § 5.1) are made available via Github (https://github.com/DE-CIX/ripe84-learning-acls) under the GPLv3 open source license. The list comprises roughly 300 filtering rules in a JSON format with a confidence of > 0.9, i.e., a packet matching the rule has a probability > 90% to be routed to a blackhole according to our dataset. A sample rule is listed below.

Please note there are some caveats with the encoding, see repository’s README. The released list may be used in multiple ways, e.g., for generating Access Control Lists (ACLs) for blocking/monitoring or for classifying packet traces. More detailed, researchers can use this list for tagging DDoS flows in traffic traces (on the flow or even packet level) with tunable confidence (see confidence field in the rule definition). For an assessment of the performance of all rules applied together, see § 6.1 (RBC classifier). Moreover, this list can be used in more practical settings to generate sets of ACLs to monitor and/or block DDoS attacks with low effort.

```
"0a42ee90": {     # Unique identifier
  "protocol":17,  # Protocol (IANA code)
  "port_src":123,  # Transport src port
  "port_dst":28960, # Transport dst port
  "packet_size":"(400,500)", # Packet size interval
  "confidence":0.99, # Confidence, see §6.1
  "antecedent_support":1021 # Ant. supp., see §6.1
}
```