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

Matthias Wichtlhuber[1], Eric Strehle[2], Daniel Kopp[1], Lars Prepens[1], Stefan Stegmueller[1]
Alina Rubina[1], Christoph Dietzel[1], Oliver Hohlfeld[2]
[1]DE-CIX [2]Brandenburg 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 at Internet Exchange Points (IXPs) which see large volumes and varieties of DDoS. IXP Scrubber continuously learns DDoS traffic properties from neighboring Autonomous Systems (ASes). It utilizes BGP signals to drop traffic for certain routes (blackholing) to sample DDoS and can 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

ACM Reference Format:
Matthias Wichtlhuber, Eric Strehle, Daniel Kopp, Lars Prepens, Stefan Stegmueller, Alina Rubina, Christoph Dietzel, Oliver Hohlfeld. 2022. IXP Scrubber: Learning from Blackholing Traffic for ML-Driven DDoS Detection at Scale. In *ACM SIGCOMM 2022 Conference (SIGCOMM '22), August 22–26, 2022, Amsterdam, Netherlands.* ACM, New York, NY, USA, 16 pages. https://doi.org/10.1145/3544216.3544268

1 INTRODUCTION

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

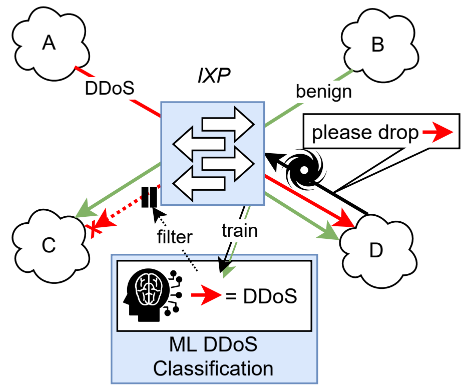


Figure 1: IXP Scrubber applies an ML DDoS classifier at IXPs at the Internet's core and filters DDoS traffic for connected networks. It learns continuously from ASes (A-D) marking unwanted traffic (blackholing).

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 victims. 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*

SIGCOMM '22, August 22–26, 2022, Amsterdam, Netherlands

ACM ISBN 978-1-4503-9420-8/22/08…$15.00
https://doi.org/10.1145/3544216.3544268

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 blackholing 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.

Table 1: Related work dataset overview.

Dataset	Year	Size	Time span	Used in:
DARPA [3]	1998	–	9 weeks	[31]
KDD99 [1]	1999	1.2 GB	9 weeks	[61]
CAIDA [4]	2007	21 GB	1 hour	[31]
ISCX2012 [8]	2012	84,42 GB	7 days	[58]
CIC-DoS [2]	2017	4.6 GB	24 hours	[26]
CICIDS2017 [50]	2017	51.1 GB	5 days	[26, 31]
IDS2018 [7]	2018	502.1 GB	9 days	[26]
Our aggregated dataset	2019-2021	3 month up to 2 years of online analysis from 5 IXPs (see § 4.1)		

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 black-box 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 (ISPs) [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.

3 BLACKHOLING: CROWDSOURCED DDOS LABELING

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

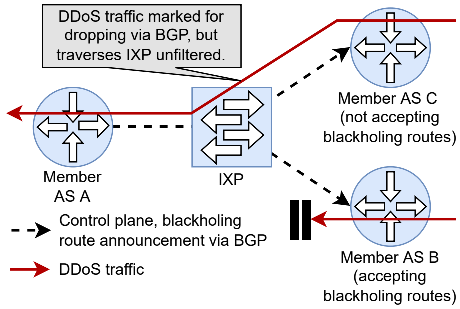


Figure 2: Blackholing is implemented between IXP members, i.e., member networks are responsible for dropping traffic. Members not accepting blackholing routes send unfiltered traffic visible at the IXP.

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 than suitable to identify hosts under attack, while our flow tagging approach (§ 5.1) extracts DDoS traffic in blackholing traffic. Our approach thus enables us to generate high-quality training data by means of low-quality labels from blackholing announcements. We thereby make blackholing announcements usable as a source for collecting DDoS samples.

Capturing blackholing traffic at IXPs. Many IXPs (including the largest ones, such as AMS-IX, DE-CIX, and LINX) offer blackholing as a feature to members to remotely drop traffic. Commonly, the mechanism is implemented between member network's routers, as shown in Figure 2, i.e., the IXP collects and distributes BGP blackholing announcements among member networks (e.g. from AS A). The blackholing announcement—indicated by carrying the blackholing community attribute [36]—indicates that traffic to the announced prefix should be dropped by other members (e.g. AS B).

Thus, for capturing blackholing traffic, we need to listen to BGP for blackholing announcements and capture all traffic routed to the blackholed prefixes. Whether any traffic can be captured depends on how blackholing announcements are handled by the receiving networks: IXP members may not adhere to the announcement and forward traffic despite receiving a blackholing route announcement. This is traffic that is *i)* unwanted by the receiver but *ii)* unfiltered and, as we show in § 4.2, carries predominantly DDoS. Notably, this only works at traffic hubs like IXPs where a larger number of networks interact. Despite its limitations [19, 29], blackholing is widely used; data from the DE-CIX looking glass shows an hourly sampled average of 2580 blackholes in 2021 (min 1290, max 3620).

Generating Balanced Data. The vast majority of traffic at IXPs is not blackholed. Figure 3a shows a CDF of the blackholing traffic share over one week for all five studied IXPs (§ 4.1). The share of blackholing traffic does not exceed 0.8% of any IXP's total traffic. In 90% of the minute bins, blackholing traffic is below 0.1% of the total traffic. Using the data as is for ML-training leads to algorithms favoring the dominant class (not blackholed) over the blackholing class. For generating a useful dataset, both traffic classes need a comparable share of data, i.e., they need to be balanced. This could be achieved with ML techniques like stratified K-folding on the whole dataset, which is offered by ML frameworks [12]. However, the sheer amount of data (more than 50 TB of flow data) breaks common toolchains. Thus, we need to vastly reduce data before training and do so in a balanced way. In our case, this means: we need to sample *i)* an equal number of destination IPs and *ii)* an equal number of flows per destination IP for blackholing and benign traffic to ensure a comparable number of data points in both classes.

Balancing procedure. For creating a balanced dataset with equal distributions of blackholing flows (underrepresented) and benign (overrepresented) flows, we use the balancing procedure shown in Figure 3b. It selects traffic for the underrepresented traffic class first (blackholing flows). We identify all traffic flows matching a BGP blackholing announcement. We then sample an equal amount of benign traffic that matches the number of target IPs and flows per time bin (in our case, at one minute resolution). Since the benign traffic is overrepresented compared to the blackholed traffic, 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 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

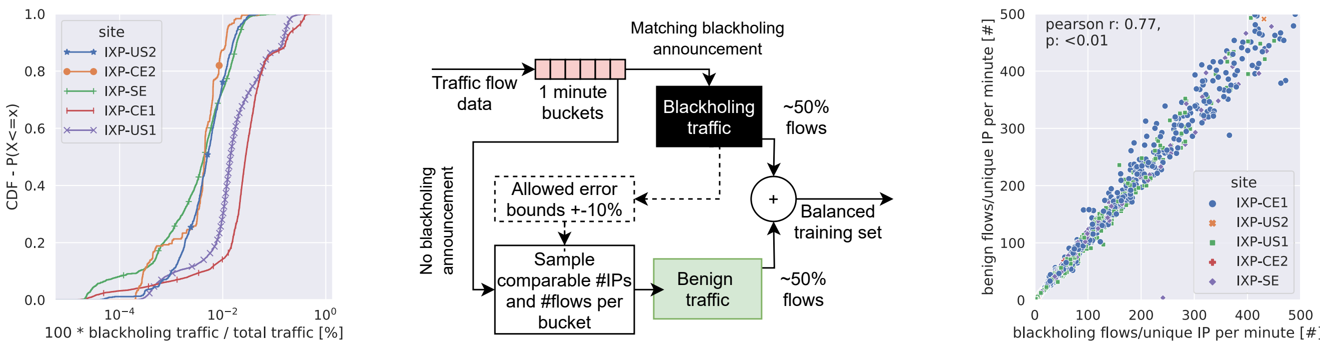
Next, we present the datasets processed for this work. In particular, we used two types of datasets: *i)* the *ML training set* predominantly used for training of ML models, and *ii)* the *self-attack set* predominantly used for validation.

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



(a) Share of blackholing traffic compared to overall traffic.

(b) Procedure for balancing benign and blackholing data to obtain a balanced training set.

(c) Flows per unique IP blackholing vs. flows per unique IP benign.

Figure 3: Basic properties of raw data, balancing procedure, balancedness of training set.

Table 2: Dataset overview (*=sum recorded online, unbalanced part of data was discarded early).

	IXP properties		Before balancing		After balancing		
	#connected ASes	Traffic peak [Tbps]	Raw flow data [TB]	#Flow records	#Flow records	Blackhole flow share [%]	Flows balanced/ flows unbalanced [%]
IXP-CE1	>800	>10.00	50.04*	685 B*	202 M	52.61	0.0294
IXP-US1	>250	>1.00	4.48*	61 B*	16 M	51.68	0.0264
IXP-SE	209	0.69	1.56*	21 B*	7 M	55.38	0.0304
IXP-US2	103	0.53	1.31*	18 B*	90 k	48.86	0.0005
IXP-CE2	211	0.12	0.22*	3 B*	9 k	48.05	0.0003
SAS (self-att. set)	-	-	-	338 k	702 k	48.16	-

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.

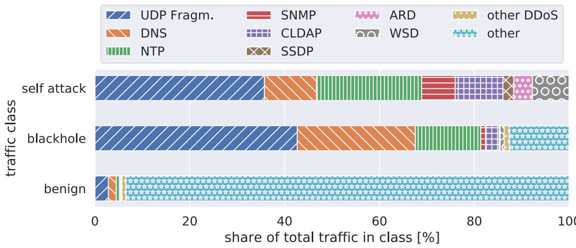
4.2 Dataset Validation

Quality of balancing in ML training set. We begin by validating our balancing approach that ensures an equal share of benign and blackholed traffic. The number of flows in the benign and blackholing class 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).

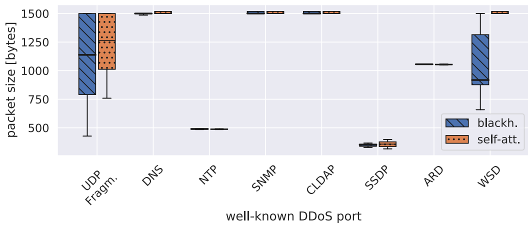
Service distribution. Figure 4a shows the share of well-known DDoS ports across three classes: *benign* and *blackholing* data of the ML training set (across all IXPs) and the SAS. Recall the SAS acts as a baseline containing DDoS traffic only. While the benign class contains ~7.5% of traffic from well-known DDoS ports such as NTP, SNMP, or LDAP, the blackhole class contains more than ~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, blackholing data contains up to 12.5% of possibly benign data, which has to be considered for ML.*



(a) Share of well-known DDoS ports (other DDoS: Ubiq. SD, rpcbind (UDP/TCP), MSSQL, DNS (TCP), chargen, DHCPDisc., GRE, memcached, WCCP, NetBios, RIP, OpenVPN, TFTP, Micr. TS).



(b) Comparison of packet size characteristics of well-known DDoS ports blackholing vs self-attack data.

Figure 4: Dataset validation.

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).

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.

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.

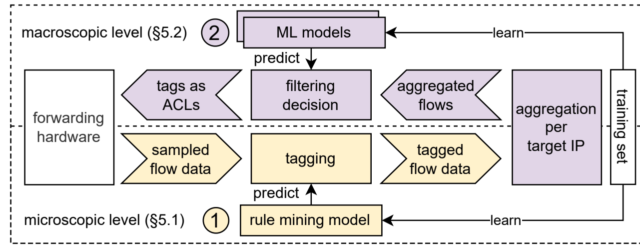


Figure 5: Overview of the machine learning model.

Step 1 introduces rule tagging to tag individual flows as benign or malicious (*microscopic 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.

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 $A \to C$, where A is a set of items called the antecedent and C is a

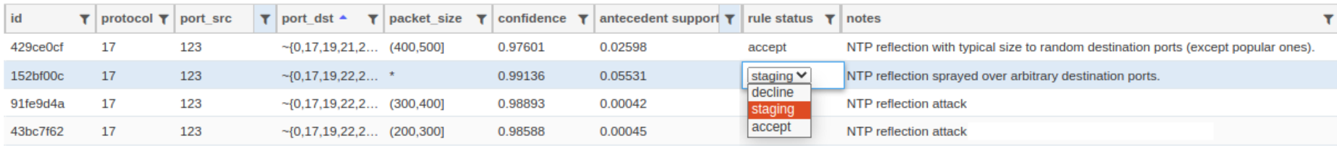


Figure 6: User interface used by network operators to validate tagging rules mined in step 1.

Algorithm 1 minimize association rules

```
function MinimizeAssociationRules(L_c, L_s,
    R = [(A_0, c_0, s_0), (A_1, c_1, s_1), ..., (A_n, c_n, s_n)] )
  while true do
    D ← ∅                                        ▷ rule indices to delete
    for i = 0 to n do                  ▷ pair-wise iteration of rules in R
      for j = 0 to n do
        if i ≠ j then
          if A_i ⊂ A_j then                  ▷ rule i's antecedent is in j's
            if (c_i − c_j < L_c) ∧ (s_i − s_j < L_s) then
              D ← {i}               ▷ remove i; limited loss in conf./supp.
    if |D| = 0 then
      break                              ▷ no more dispensable rules in R
    for k = 0 to n do                               ▷ remove rules from R
      if k ∈ D then
        R ← R − {R[k]}
  return R
```

---

set of items called the consequent. We apply it to our dataset to mine for packet headers as antecedents, which often occur with the {blackhole} consequent. For example, IXP members receiving NTP traffic and blackholing this traffic generate an association rule {protocol = UDP, port src = 123} ⟶ {blackhole}.

Performance metrics. ARM provides comprehensible metrics to assess the relevance of an association rule: antecedent support s and confidence c; s represents the share of the antecedent in the whole dataset, i.e., the overall relevance of the respective header combination; c represents the share of cases where the antecedent co-occurred with the consequent, i.e., if a header combination co-occurs with blackhole in 90% of the cases the confidence is 0.9.

Rule set minimization. A direct application of rule mining can easily result in a filtering rule set that is too large to comprehend. To illustrate this, we apply the FP-Growth [33] association rule learning algorithm with a minimum required confidence of 0.8 to our data and obtain 7,859 rules—too large to be curated manually.

For rule minimization, we design a two-step method: *i)* we first remove all association rules where the consequent is not {blackhole}. This removes 6,565 association rules, leaving us with 1,469 remaining rules. Based on the reduced rule set, we *ii)* further minimize the association rules using Algorithm 1, reducing the rule set to a manageable size of 367 rules.

Algorithm 1 utilizes the situation that antecedents of two association rules can be a proper subset of each other if they have the same consequent (recall in our case all consequents are {blackhole}). If this is the case, the antecedent with more elements has a likely lower or equal confidence c and support s, as it is more specific. Algorithm 1 runs on a list of all antecedents ($A_{0..n}$), their confidences ($c_{0..n}$), and support values ($s_{0..n}$). It tests all antecedents for being subsets of each other. If two antecedents are found to be subsets of each other, the confidences/support values are compared to not exceed a loss threshold L_c/L_s. The process is repeated until no more rules can be removed. Algorithm 1 has a complexity of $O(|R|^2)$ with R representing the number of rules to be minimized. However, execution time never exceeded 60 seconds with the given data on a standard consumer laptop. We set both L_c/L_s to 0.01, based on a parameter sensitivity analysis presented in Appendix A, Figure 15.

5.1.2 Interpretability. After minimizing the rule set, it is presented to network operators in a user interface (UI) as shown in Figure 6. The UI shows the header data, confidence and antecedent support. Users can tag and order rules by arbitrary columns and classify each rule as *decline* (will never show up again), *staging* (consider for later acceptance), and *accept* to accept the rule in the association rule set for further processing. For documentation, comments can be added. Moreover, association rules can be exported and imported, where they can be merged with freshly mined rules. This allows a growing tagging rule set over time. Please note that for legal reasons we cannot release raw datasets. However, we release the tagged rules mined on our datasets under an open source license; see Appendix F for details.

5.1.3 Interpretability Evaluation with Operators. Mined tagging rules are interpretable by network operators and can be manually verified quickly. We test these hypotheses in a small-scale subjective study that involves *i)* two network operators at one IXP and *ii)* three of the authors that did not design the rule mining approach. We base this study on a rule set that we mined from the SAS. The rule set presented to the subjects consists of 38 rules that need to be accepted (drop traffic) or declined (pass traffic) by each subject. The goal of the study is to evaluate *i)* the quality of the compiled rules and *ii)* the time needed for classification. We evaluate the selection quality by matching the rule set compiled by each subject to the test data and compute *i)* the percentage of correctly dropped traffic from the DDoS self-attacks and *ii)* the percentage of incorrectly dropped traffic from the benign dataset.

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.

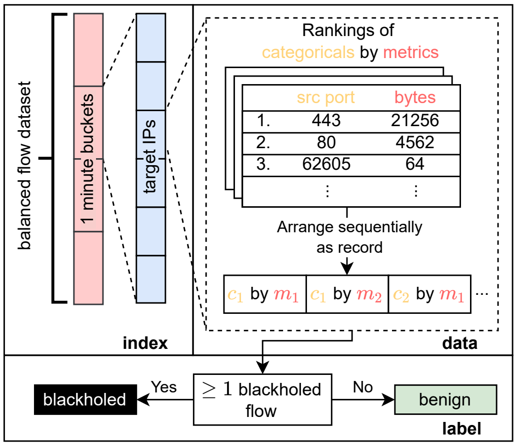


Figure 7: All flow data per 1-minute bucket and target IP is aggregated into rankings. The example shows source ports ranked by bytes received from each port.

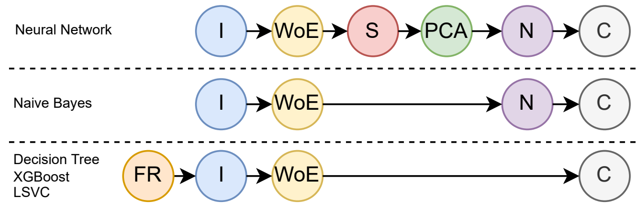


Figure 8: Overview of the machine learning model preprocessing pipelines; FR=feature reduction, drops unnecessary features identified upfront; I=imputer, replaces null values with -1; WoE=weight of evidence encoding of all categorical columns; S=standardize by mean and unit variance; PCA=principal component analysis; N=normalizer, normalize values to interval [0;1];C=classifier.

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 = {*source IPs, source port, destination port, source MAC address, transport protocol*} are ranked based on the non-categorical flow metrics M = {*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 $|M| * |C|$ 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 IPs 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 $\mathrm{WoE}(x_i) = \ln(\frac{P(X=x_i|y=1)}{P(X=x_i|y=0)})$, where y is the blackhole label[1]. 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

[1]We handle the division-by-zero case by adding 1.0 to the numerator and the denominator. Unknown x_is are encoded as $\mathrm{WoE}(x_i) = 0.0$ during prediction, i.e., neutral.

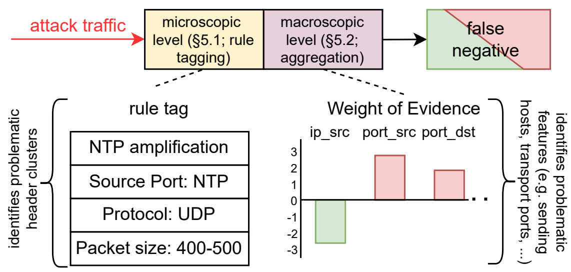


Figure 9: Weight of Evidence and rule tags can be used to understand and debug misclassifications.

high number of defaults will lead to high WoE and vice versa. In our case, the WoE allows mapping IPs, MACs and port numbers 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 selection 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.*

5.2.3 *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 $F_1 = \mathrm{tp}/(\mathrm{tp}+\frac{1}{2}(\mathrm{fp}+\mathrm{fn}))$. Since false positive classifications (i.e., wrongly attributed attack traffic that would be blocked) have a more severe negative impact on the model deployment than undetected attacks (false negative), we additionally show a weighted ratio F_β, which we use in the remainder of this work, expressed as $F_\beta = \frac{(1+\beta^2)\cdot\mathrm{tp}}{(1+\beta^2)\cdot\mathrm{tp}+\beta^2\cdot\mathrm{fn}+\mathrm{fp}}$ 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.

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).

	$F_{\beta=0.5}$	F_1	mcc	tnr	fnr	tpr	fpr	UDP Fragm.	DNS	NTP	SNMP	LDAP	SSDP	Apple RD	$F_{\beta=0.5}$ (all on SAS)
XGB	**0.989**	**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	0.010	**0.994**	0.993	0.990	**0.996**	0.991	0.959	0.991	0.631
LSVM	0.978	0.973	**0.001**	0.981	0.035	0.965	0.019	0.993	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	0.009	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

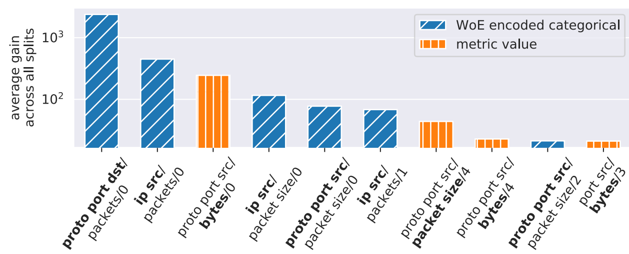


Figure 10: XGB features with highest avg. gain for all splits (notation: categorical/metric/rank, see Figure 7).

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 benefit 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).*

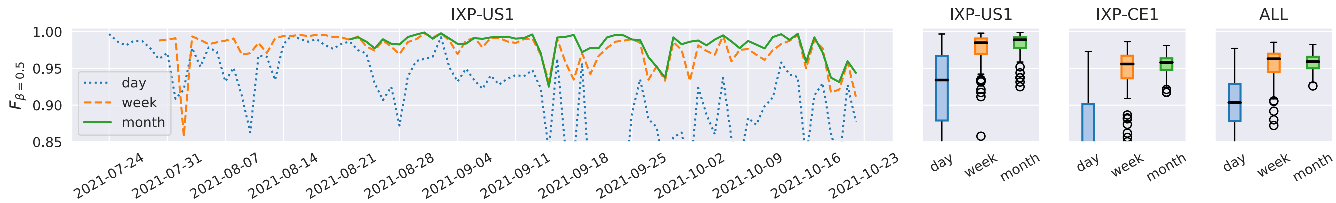
6.2 Features

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 reflector). Since reflector IPs can change over time, we evaluate the temporal model drift in § 6.3 and specifically focus on reflector 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.

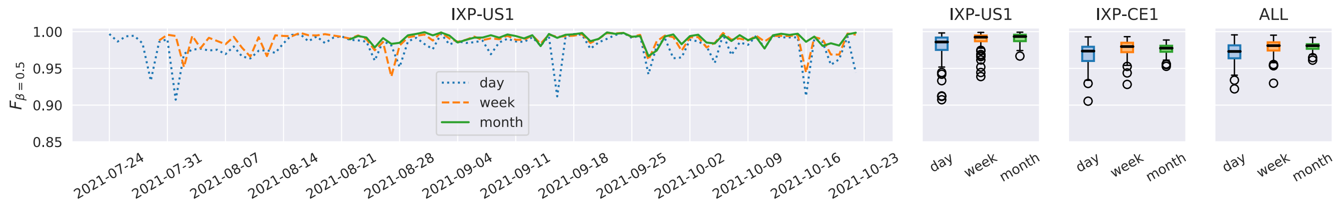
6.3 Temporal Model Drift

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. With the evaluation of model performance over large traffic traces, we provide the first evaluation of temporal stability and complement a large body of prior work that focuses on showing model performance over the entire dataset only.

For how long do we need to train? To answer this question, we show the XGB model performance as $F_{\beta=0.5}$ score over time for



(a) One-shot training.



(b) Sliding Window training.

Figure 11: XGB model performance as $F_{\beta=0.5}$ score over time (YYYY-MM-DD) for IXP-US1, IXP-CE1 and a model learned on all five IXPs. The training intervals are one *i)* day (blue), *ii)* week (orange) or *iii)* month (green).

IXP-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 IPs; these require forgetting information when, e.g., IPs 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 IPs appearing in the WoE encoding. In order to restrict this analysis to the knowledge of reflectors, we only consider source IPs with a WoE> 1.0, i.e., IPs that are $e = 2.71$ 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 IPs anymore and need to utilize other location independent features. Note that knowledge on reflection

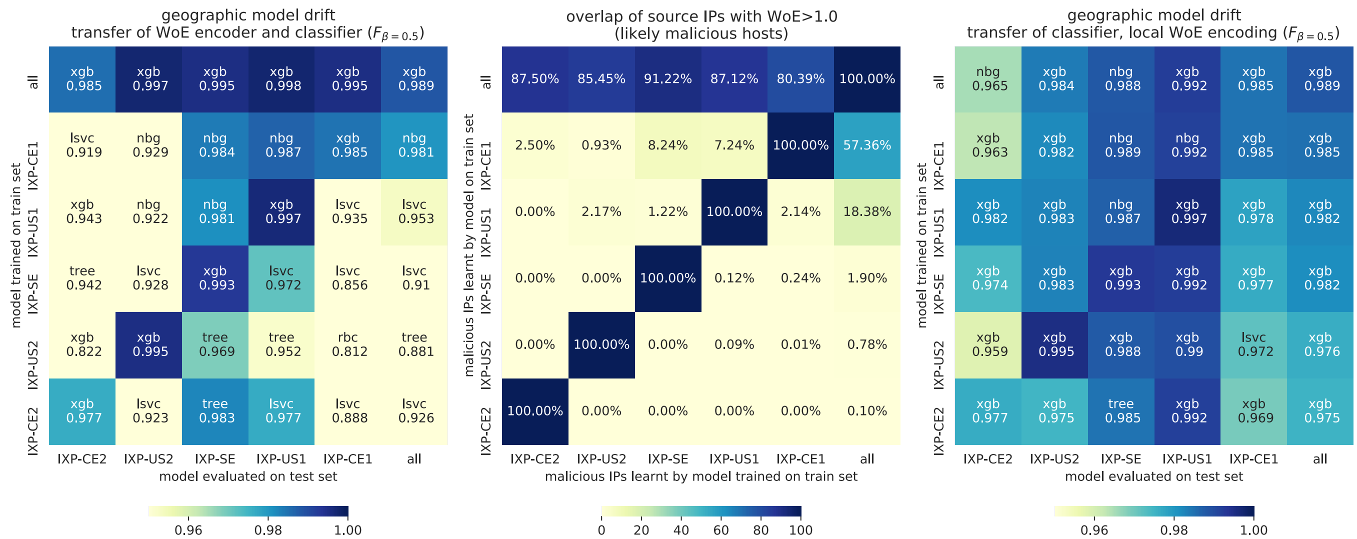


Figure 12: Geographic model drift across different IXPs.

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 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 IXP 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 IXP 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.

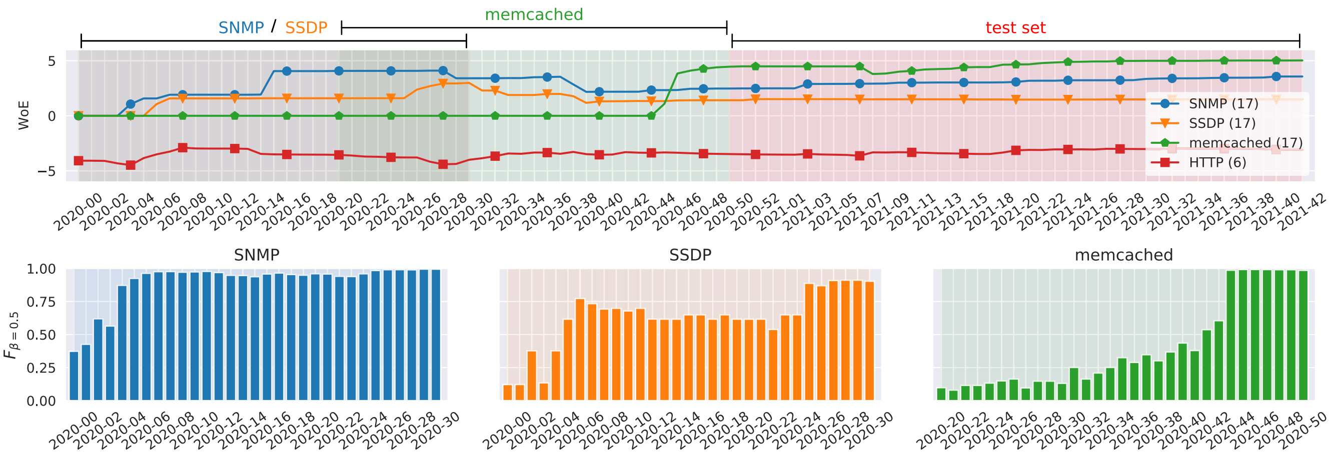
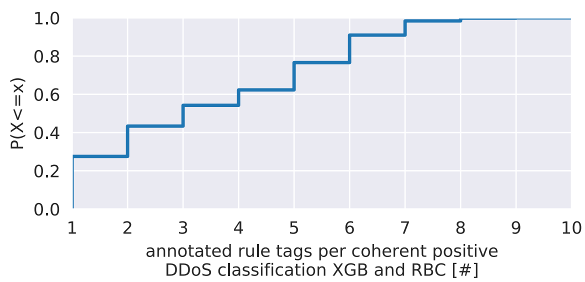
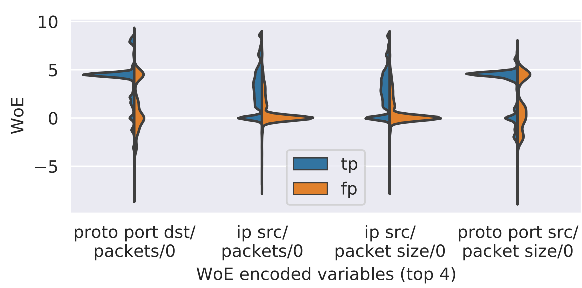


Figure 13: IXP Scrubber learns new DDoS vectors without IXP operators' intervention as they are blackholed.



(a) Rule tags per coherent positive decision XGB/RBC.



(b) WoE distr. of top four XGB features for tp/fp.

Figure 14: Tagging rule annotations and WoE encoding enable local explainability.

WoE encoding. Figure 14b compares WoE distributions for true positive/false positive classifications for the four top WoE encoded features for XGB. The results show considerably different distributions for true and false classifications, where false positives are more likely to exhibit lower WoE scores compared to true positives, especially with respect to unknown source IPs (WoE= 0). A false positive can be mitigated by moving one or more feature's WoE outside the WoE distribution for true positives, e.g., whitelisting source IPs or transport ports with a static, negative WoE. The analysis for true negatives/false negatives looks similar (not shown).

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 IXP 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 $F_{\beta=0.5}$-score on all targets and more than 0.99 $F_{\beta=0.5}$-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 $F_{\beta=0.5}$-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).

REFERENCES

[1] 1999. KDD Cup 1999 Data. http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html. Accessed: 2022-01-25.

[2] 2022. CIC DoS Dataset (2017). https://www.unb.ca/cic/datasets/dos-dataset.html. Accessed: 2022-01-25.

[3] 2022. DARPA Intrusion Detection Evaluation. https://archive.ll.mit.edu/ideval/index.html. Accessed: 2022-01-31.

[4] 2022. DDoS 2007 Attack. https://catalog.caida.org/details/dataset/ddos_attack_2007. Accessed: 2022-1-31.

[5] 2022. DecisionTreeClassifier. https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html. Accessed: 2022-02-02.

[6] 2022. FastNetMon. https://fastnetmon.com/. Accessed: 2022-01-26.

[7] 2022. Intrusion Detection Evaluation Dataset (CSE-CIC-IDS2018). https://www.unb.ca/cic/datasets/ids-2018.html. Accessed: 2022-01-31.

[8] 2022. Intrusion Detection Evaluation Dataset (ISCXIDS2012). https://www.unb.ca/cic/datasets/ids.html. Accessed: 2022-01-25.

[9] 2022. LinearSVC. https://scikit-learn.org/stable/modules/generated/sklearn.svm.LinearSVC.html. Accessed: 2022-02-02.

[10] 2022. Naive Bayes. https://scikit-learn.org/stable/modules/classes.html?highlight=naive%20bayes#module-sklearn.naive_bayes Accessed: 2022-02-02.

[11] 2022. NeuralNet. https://skorch.readthedocs.io/en/stable/user/neuralnet.html. Accessed: 2022-02-02.

[12] 2022. Scikit learn: Stratified k-fold. https://scikit-learn.org/stable/modules/cross_validation.html#stratified-k-fold Accessed: 2022-07-01.

[13] 2022. XGBoost Parameters. https://xgboost.readthedocs.io/en/stable/parameter.html. Accessed: 2022-02-02.

[14] R. Agrawal, T. Imieliński, and A. Swami. 1993. Mining Association Rules Between Sets of Items in Large Databases. In *ACM SIGMOD*.

[15] Akamai. 2018. Memcached DDoS Explained. https://www.akamai.com/our-thinking/threat-advisories/memcached-ddos-explained. Accessed: 2022-07-01.

[16] A. Aljuhani. 2021. Machine Learning Approaches for Combating Distributed Denial of Service Attacks in Modern Networking Environments. *IEEE Access* 9 (2021), 42236–42264.

[17] M. Antonakakis, T. April, M. Bailey, M. Bernhard, E. Bursztein, J. Cochran, Z. Durumeric, et al. 2017. Understanding the Mirai Botnet. In *USENIX Security*.

[18] B. Arzani, K. Hsieh, and H. Chen. 2021. Interpretable Feedback for AutoML and a Proposal for Domain-Customized AutoML for Networking. In *SIGCOMM HotNets*.

[19] M. Nawrockiand J. Blendin, C. Dietzel, T. C. Schmidt, and M. Wählisch. 2019. Down the Black Hole: Dismantling Operational Practices of BGP Blackholing at IXPs. In *ACM IMC*.

[20] K. Bock, A. Alaraj, Y. Fax, K. Hurley, E. Wustrow, and D. Levin. 2021. Weaponizing Middleboxes for TCP Reflected Amplification. In *USENIX Security*.

[21] A. Büscher and T. Holz. 2012. Tracking DDoS Attacks: Insights into the Business of Disrupting the Web. In *USENIX Workshop on LEET*.

[22] O. Çetin, C. Gañán, L. Altena, T. Kasama, D. Inoue, K. Tamiya, Y. Tie, K. Yoshioka, and M. van Eeten. 2019. Cleaning Up the Internet of Evil Things: Real-World Evidence on ISP and Consumer Efforts to Remove Mirai. In *NDSS*.

[23] T. Chan and C. Guestrin. 2016. Xgboost: A Scalable Tree Boosting System. In *SIGKDD*.

[24] A. D'Amour, K. Heller, D. Moldovan, B. Adlam, B. Alipanahi, A. Beutel, C. Chen, J. Deaton, J. Eisenstein, M. D. Hoffman, et al. 2020. Underspecification Presents Challenges for Credibility in Modern Machine Learning. *arXiv preprint arXiv:2011.03395* (2020).

[25] S. Das, A. M. Mahfouz, D. Venugopal, and S. Shiva. 2019. DDoS intrusion detection through machine learning ensemble. In *2019 IEEE 19th international conference on software Quality, Reliability and Security Companion (QRS-C)*. IEEE, 471–477.

[26] F. S. de Lima Filho, F. A. F. Silveira, A. de Medeiros Brito Júnior, G. Vargas-Solar, and L. F. Silveira. 2019. Smart Detection: An Online Approach for DoS/DDoS Attack Detection Using Machine Learning. *Security and Communication Networks* 2019 (2019), 1574749:1–1574749:15.

[27] B. S. Kiruthika Devi, G. Preetha, G. Selvaram, and S. Mercy Shalinie. 2014. An Impact Analysis: Real Time DDoS Attack Detection and Mitigation Using Machine Learning. In *2014 ICRTITA*. IEEE, 1–7.

[28] C. Dietzel, A. Feldmann, and T. King. 2016. Blackholing at IXPs: On the Effectiveness of DDoS Mitigation in the Wild. In *PAM*.

[29] C. Dietzel, M. Wichtlhuber, G. Smaragdakis, and A. Feldmann. 2018. Stellar: Network Attack Mitigation Using Advanced Blackholing. In *ACM CoNEXT*.

[30] T. Greene. 2016. How the Dyn DDoS Attack Unfolded. https://www.networkworld.com/article/3134057/how-the-dyn-ddos-attack-unfolded.html. Accessed: 2022-07-01.

[31] Y. Gu, K. Li, Z. Guo, and Y. Wang. 2019. Semi-Supervised K-means DDoS Detection Method Using Hybrid Feature Selection Algorithm. *IEEE Access* 7 (2019), 64351–64365.

[32] D. Gunning, M. Stefik, J. Choi, T. Miller, S. Stumpf, and G. Z. Yang. 2019. XAI-Explainable Artificial Intelligence. *Science Robotics* 4, 37 (2019), eaay7120.

[33] J. Han, J. Pei, and Y. Yin. 2000. Mining Frequent Patterns without Candidate Generation. In *ACM SIGMOD*.

[34] B. Jia, X. Huang, R. Liu, and Y. Ma. 2017. A DDoS Attack Detection Method Based on Hybrid Heterogeneous Multiclassifier Ensemble Learning. *J. Electr. Comput. Eng.* 2017 (2017), 4975343:1–4975343:9.

[35] M. Jonker, A. King, J. Krupp, C. Rossow, A. Sperotto, and A. Dainotti. 2017. Millions of targets under attack: a macroscopic characterization of the DoS ecosystem. In *ACM IMC*.

[36] T. King, C. Dietzel, J. Snijders, G. Doering, and G. Hankins. 2016. BLACKHOLE Community. IETF RFC 7999.

[37] D. Kopp, C. Dietzel, and O. Hohlfeld. 2021. DDoS Never Dies? An IXP Perspective on DDoS Amplification Attacks. In *PAM*.

[38] D. Kopp, M. Wichtlhuber, I. Poese, J. Santanna, O. Hohlfeld, and C. Dietzel. 2019. DDoS Hide and Seek: On the Effectiveness of a Booter Services Takedown. In *ACM IMC*.

[39] B. Krebs. 2016. KrebsOnSecurity Hit With Record DDoS. https://krebsonsecurity.com/2016/09/krebsonsecurity-hit-with-record-ddos. Accessed: 2022-07-01.

[40] X. Luo and R. KC. Chang. 2005. On a New Class of Pulsing Denial-of-Service Attacks and the Defense.. In *NDSS*.

[41] N. Mehrabi, F. Morstatter, N. Saxena, K. Lerman, and A. Galstyan. 2021. A Survey on Bias and Fairness in Machine Learning. *ACM Computing Surveys (CSUR)* 54, 6 (2021), 1–35.

[42] J. Mohamed. 2016. Daily Mirror: Hackers Attack the Stock Exchange: Cyber Criminals Take Down Website for more than Two Hours as Part of Protest Against World's Banks. http://www.dailymail.co.uk/news/article-3625656/Hackers-attack-Stock-Exchange-Cyber-criminals-website-two-hours-protest-against-world-s-banks.html. Accessed: 2022-07-01.

[43] C. Morales. 2018. NETSCOUT Arbor Confirms 1.7 Tbps DDoS Attack; The Terabit Attack Era Is Upon Us. https://www.netscout.com/blog/asert/netscout-arbor-confirms-17-tbps-ddos-attack-terabit-attack-era. Accessed: 2022-07-01.

[44] G. C. M. Moura, C. Hesselman, G. Schaapman, N. Boerman, and O. de Weerdt. 2020. Into the DDoS Maelstrom: A Longitudinal Study of a Scrubbing Service. In *IEEE EuroS&P Workshops*. 550–558.

[45] M. Nawrocki, M. Jonker, T. C. Schmidt, and M. Wählisch. 2021. The Far Side of DNS Amplification: Tracing the DDoS Attack Ecosystem from the Internet Core. In *ACM IMC*.

[46] M. Prince. 2013. The DDoS That Knocked Spamhaus Offline (And How We Mitigated It). https://blog.cloudflare.com/the-ddos-that-knocked-spamhaus-offline-and-ho/. Accessed: 2022-07-01.

[47] M. Prince. 2014. Technical Details Behind a 400Gbps NTP Amplification DDoS Attack. https://blog.cloudflare.com/technical-details-behind-a-400gbps-ntp-amplification-ddos-attack/. Accessed: 2022-07-01.

[48] C. Rudin. 2019. Stop Explaining Black Box Machine Learning Models for High Stakes Decisions and Use Interpretable Models Instead. *Nature Machine Intelligence* 1, 5 (2019), 206–215.

[49] A. Rukavitsyn, K. Borisenko, and A. Shorov. 2017. Self-Learning Method for DDoS Detection Model in Cloud Computing. In *2017 IEEE EIConRusNW*.

[50] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani. 2018. Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization. *ICISSp* 1 (2018), 108–116.

[51] I. A. Sofi, A. Mahajan, and V. Mansotra. 2017. Machine Learning Techniques used for the Detection and Analysis of Modern Types of DDoS Attacks. *Int. Res. J. Eng. Technol* (2017).

[52] Akamai Technologies. 2018. 2018 State of the Internet / Security: A Year in Review. https://web.archive.org/web/20210308082738/https://www.akamai.com/us/en/multimedia/documents/state-of-the-internet/2018-state-of-the-internet-security-a-year-in-review.pdf. Accessed: 2022-07-01.

[53] A. Toh. 2022. Azure DDoS Protection—2021 Q3 and Q4 DDoS Attack Trends. https://azure.microsoft.com/en-us/blog/azure-ddos-protection-2021-q3-and-q4-ddos-attack-trends/. Accessed: 2022-07-01.

[54] N. N. Tuan, P. H. Hung, N. D. Nghia, N. V. Tho, T. V. Phan, and N. H. Thanh. 2020. A DDoS Attack Mitigation Scheme in ISP Networks Using Machine Learning Based on SDN. *Electronics* 9, 3 (2020).

[55] D. Wagner, D. Kopp, M. Wichtlhuber, C. Dietzel, O. Hohlfeld, G. Smaragdakis, and A. Feldmann. 2021. United We Stand: Collaborative Detection and Mitigation of Amplification DDoS Attacks at Scale. In *ACM CCS*.

[56] D. Weed. 2005. Weight of Evidence: A Review of Concept and Methods. *Risk Analysis* 25, 6 (2005), 1545–1557.

[57] A. Welzel, C. Rossow, and H. Bos. 2014. On Measuring the Impact of DDoS Botnets. In *EuroSec*. 1–6.

[58] X. Yuan, C. Li, and X. Li. 2017. DeepDefense: Identifying DDoS Attack via Deep Learning. In *IEEE SMARTCOMP*. 1–8.

[59] ZDNet. 2018. GitHub Hit with the Largest DDoS Attack Ever Seen. https://www.zdnet.com/article/github-was-hit-with-the-largest-ddos-attack-ever-seen/. Accessed: 2022-07-01.

[60] B. Zhang, T. Zhang, and Z. Yu. 2017. DDoS Detection and Prevention Based on Artificial Intelligence Techniques. In *2017 IEEE ICCC*. 1276–1280.

[61] N. Zhang, F. Jaafar, and Y. Malik. 2019. Low-Rate DoS Attack Detection Using PSD Based Entropy and Machine Learning. In *IEEE CSCloud and IEEE Edgecom*. 59–62.

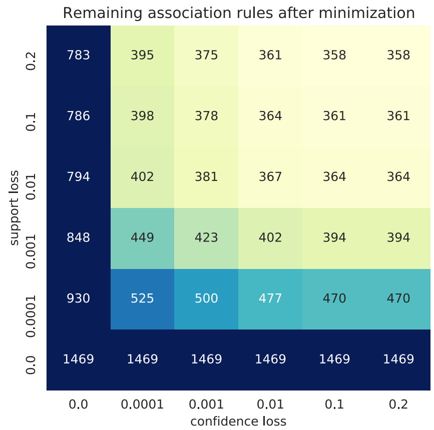
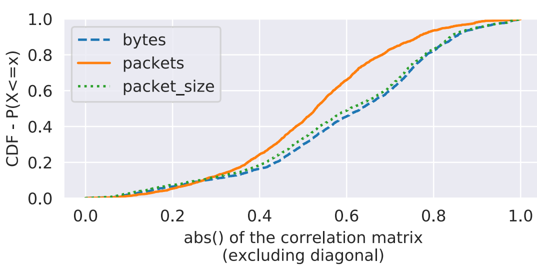
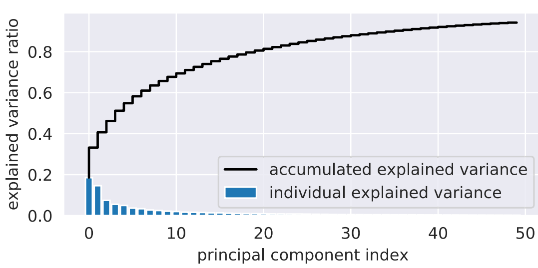


Figure 15: Association rules after minimization for different combinations of support loss L_s and confidence loss L_c.



(a) CDF of correlation matrix (absolute values).



(b) Principal component analysis.

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

Classifier	**Parameter**	**Parameter Space**
Naive Bayes	*var. smoothing*[1]	{10^{-9}, 10^{-8}, 10^{-7}, 10^{-6}, 10^{-5}, 10^{-4}, 10^{-3}, 0.01, **0.1**, 1}
	additive smoothing[2]	{10^{-8}, 10^{-4}, 10^{-3}, 0.01, 0.1, 0.5, 1.0, **2.0**, 10.0}
Decision Tree	*ccp alpha*	{10^{-9}, 10^{-7}, 10^{-5}, **0**}
	min. impurity decrease	{$\mathbf{10^{-5}}$, 10^{-3}}
	min. samples leaf	{**1**, 100, 300}
	min. samples split	{**2**, 100}
XGBoost	*# estimators*	{2, 4, 8, 16, **24**}
	max. depth	{4, 8, 16, **24**}
	learning rate	{0.01, 0.1, 0.2, **0.3**}
LSVC	*regularization (C)*	{$\mathbf{10^{-5}}$, 10^{-4}, 10^{-3}, 0.01 , 0.1, 1, 10, 100, 1000, 10000}
	class weight	{**none**, balanced}
	optimization	{primal, **dual**}
	loss	{hinge, **squared hinge**}
Neural Network	*# PCA components*	{25, 50, **75**}
	# neurons hidden layer	{4, 8, **16**, 32}
	dropout	{**0** , 0.3, 0.6, 0.9}
	learning rate	{10^{-5}, $6.3 \cdot 10^{-5}$, $3.9 \cdot 10^{-4}$, $\mathbf{2.5 \cdot 10^{-3}}$}

[1] Applies only to the Gaussian Naive Bayes.

[2] Applies to Naive Bayes types other than Gaussian.

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

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.

	$F_{\beta=0.5}$	F_1	mcc	tnr	fnr	tpr	fpr	UDP Fragm.	DNS	NTP	SNMP	LDAP	SSDP	Apple RD	$F_{\beta=0.5}$ (all on SAS)
XGB	**0.989**	**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	0.010	**0.994**	0.993	0.990	**0.996**	0.991	0.959	0.991	0.631
LSVM	0.978	0.973	**0.001**	0.981	0.035	0.965	0.019	0.993	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	0.009	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
NB-C	0.898	0.908	0.018	0.881	0.075	0.925	0.119	0.991	0.991	0.989	**0.996**	0.991	0.958	0.989	0.568
NB-M	0.894	0.906	0.018	0.873	0.072	0.928	0.127	0.991	0.991	0.989	**0.996**	0.991	0.958	0.989	0.568
RBC	-	-	-	-	-	-	-	-	-	-	-	-	-	-	**0.917**
NB-B	0.769	0.768	0.022	0.757	0.233	0.767	0.243	0.944	0.931	0.943	0.730	0.940	0.635	0.694	0.000
DUM	0.511	0.506	-	0.501	0.498	0.500	0.499	-	-	-	-	-	-	-	0.530

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 *i)* 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); this will most likely be done by operators for well-known DDoS features anyways, e.g., setting a positive WoE for DDoS transport ports and a negative one for important protocols like HTTP(S).

The most likely scenario is that attackers create a positive WoE for hosts by spoofing source IPs, which may qualify these hosts as DDoS reflectors. However, spoofed IPs are present in DDoS traffic as of today and the algorithms presented in this work can cope 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
}
```