Establishing the Contaminating Effect of Metadata Feature Inclusion in Machine-Learned Network Intrusion Detection Models

Laurens D'hooge(✉), Miel Verkerken, Bruno Volckaert, Tim Wauters, and Filip De Turck

IDLab, Department of Information Technology, Ghent University - imec, Technologiepark-Zwijnaarde 126, Gent, Belgium
laurens.dhooge@ugent.be

Abstract. Modern datasets in intrusion detection are designed to be evaluated by machine learning techniques and often contain metadata features which ought to be removed prior to training. Unfortunately many published articles include (at least) one such metadata feature in their models, namely destination port. In this article, it is shown experimentally that this feature acts as a prime target for shortcut learning. When used as the only predictor, destination port can separate ten state of the art intrusion detection datasets (CIC collection, UNSW-NB15, CIDDS collection, CTU-13, NSL-KDD and ISCX-IDS2012) with 70 to 100% accuracy on class-balanced test sets. Any model that includes this feature will learn this strong relationship during training which is only meaningful within the dataset. Dataset authors can take countermeasures against this influence, but when applied properly, the feature becomes non-informative and could just as easily not have been part of the dataset in the first place. Consequently, this is the central recommendation in this article. Dataset users should not include destination port (or any other metadata feature) in their models and dataset authors should avoid giving their users the opportunity to use them.

Keywords: Intrusion detection · Machine learning · Shortcut learning · Dataset issues

1 Introduction

Intrusion detection systems (IDS) are part of multi-layered network defenses. They are tasked with identifying unusual patterns that are a consequence of malicious use of the network. In the academic literature, modern intrusion detection methods rely on machine learning (ML) to accomplish this goal. They are evaluated on a set of specialty datasets. Method recognition as an improvement over the current state-of-the-art is conditioned on its ability to outperform previous

L. Cavallaro et al. (Eds.): DIMVA 2022, LNCS 13358, pp. 23–41, 2022.
https://doi.org/10.1007/978-3-031-09484-2_2

methods on standard classification metrics. Intentional manipulation notwithstanding, it is still possible that choices in dataset (pre)processing lead to overly optimistic results.

The goal of this article is to demonstrate that one common choice in the preprocessing stage unintentionally overstates the results in a major way. That choice is to keep destination port in the feature set. Destination port is part of the network flow metadata quadfecta: source and destination IP addresses and ports. While most researchers do realize that IP addresses would bias the model and that source port should never be informative because it is randomly assigned by the operating system, destination port gets to stay in the training data much more frequently. It is kept in the training data often without a stated motivation or analysis. If a motivation is given, then it tends to align with the statement that the feature contains relevant information to identify attacks that are bound to specific services. From a theoretical perspective there is some merit to this statement, although it too introduces unnecessary bias in the model. Most importantly, the datasets are not generated in a way that counterbalances the potential bias of destination port. Depending on the attack type in the dataset, the distribution of destination port can exhibit significant scatter for attacks and concentrated values for the selection of (emulated) benign services. This pattern is quickly picked up by learning methods and then acts as a shortcut, leading to inflation of the results.

We demonstrate that the shortcut learning effect of destination port in ML-IDS is real and that proposed methods which include it are not screened out during peer review. In order to demonstrate the effectiveness of destination port, it is used as the single predictor to classify ten state-of-the-art IDS datasets (and their different subsets, i.e. attack classes). Destination port can separate the datasets with accuracies between 70 and 100% (overview results in Tables 2, 3 and 4). When using the other metadata features as single predictors, their models reach similar heights of undeserved classification performance (summarized results in Subsect. 4.7).

Accuracy can be used as a metric because the dataset is balanced by class label prior to sampling train and test portions. Without this balancing, using accuracy as a metric would be very misleading due to the inherent class imbalance in many of the IDS datasets. Even though this balancing reduces the amount of samples to train and test on, the accuracy on a second test set (the remainder of the samples, always one class) stays in-line with the results on the balanced test set. Furthermore, the results on a 20–80 train-test split, rather than the typical 80–20 split, are just marginally worse, indicating that even fewer samples are enough to learn the relation between destination port and the class label.

The recommendation that follows from this analysis is simple. The destination port should be viewed as part of the metadata features to be omitted during preprocessing. Proposed methods that keep it (or any other metadata feature) should be sent back during review to re-evaluate the method without its (or their) inclusion. This issue can be addressed at the source by dataset authors if they would stop providing the metadata features altogether when they publish new datasets.

The remainder of this article is laid out in six other sections. The related work introduces the datasets and demonstrates that the problem of metadata inclusion is widespread in the literature. The methodology provides the details regarding algorithm choice, dataset preprocessing and the ML pipeline. In the interest of full transparency, all datasets and preprocessing code (original, clean, and dirty with metadata as used in this analysis) are openly available at https://gitlab.ilabt.imec.be/lpdhooge/ids-dataset-collection. The code for this analysis is also openly stored on the same platform at https://gitlab.ilabt.imec.be/lpdhooge/ids-metadata-contamination. Section 4 (results) is primarily conveyed through Tables 2, 3 and 4 with additional explanation in textual format in the subsections. The results also cover the other metadata features (source and destination IP addresses, source port and metadata time features) in a summarized format. The common patterns are discussed in Sect. 5 after which the article concludes and proposes future work into uncovering non-metadata contaminant features in IDS datasets.

2 Related Work

This section briefly introduces the concept of shortcut learning, followed by introductions for each of the ten datasets that are part of the analysis (Subsect. 2.2). Subsequently a non-exhaustive list (Table 1) of published methods is presented in which the authors included one or more metadata features from the aforementioned datasets. It is not difficult to find published articles, even with high citation counts (recorded through Google Scholar in March 2022) which included features prone to shortcut learning in their models.

2.1 Shortcut Learning

The issue of shortcut learning refers to the ability of learning algorithms to find and exploit spurious correlations in the data. This commonly happens when the dataset has an insufficient sampling of the problem and/or the data generation process left artifacts in the data. The algorithms learn to recognize the artifacts and seemingly perform well on the original dataset's test set(s). However, when exposed to a broader test set, the models fail to maintain their performance. Good ML practitioners use methods from the domains of interpretable ML and explainable artificial intelligence (XAI) to check whether their models have learnt shortcuts. Especially in the context of deep neural networks, the issue has gotten much attention [9]. More pertinent to this application domain, the issue of spurious correlations in cybersecurity-related ML research has been raised by Arp et al. [4].

2.2 State of the Art Datasets

For clarity's sake, when metadata features are mentioned, they include source and destination IPs, source and destination ports, timestamps and NetFlow IDs. As the objective of this article is to definitively place destination port among the metadata features which ought not to be used in models, we will always refer to it as a metadata feature.

NSL-KDD, ISCX-IDS2012 and the CIC Collection. The university of New Brunswick has the longest-standing history within the field of generating (network) intrusion datasets. It is the leading university in the Canadian Institute for Cybersecurity (CIC), a partnership between government, academia and industry. In 2009 they published an updated version of KDD99, dubbing it NSL-KDD [34], to address KDD99's shortcomings. NSL-KDD subsequently dominated as the dataset of choice, indirectly expanding the lifespan of KDD99 well beyond a decade of use. Interestingly, NSL-KDD (and KDD99) did not include destination port explicitly, instead opting to include a service field which contains the same information, abstracted away from any specific port. The ability of this destination port proxy to contaminate will be evaluated (Sect. 4.1).

In 2012, the first dataset from their own testbed was published. The paper which introduced the methodology behind ISCX-IDS2012 [32] laid out a framework of attack execution and user simulation at various levels of automation. The research group chose to develop their own feature extractor (ISCXFlowmeter) which operates on reconstructed network flows. The flow feature set of ISCX-IDS2012 was limited and also included reconstructed packet payloads (20 features: 9 flow, 4 payload, 1 label, 6 metadata). All targets (labels) were pre-encoded by the dataset authors in a binary *attack/normal*. A higher level of resolution for the labeling is not available.

The foundational parts had been put in place to build upon and refine the dataset generation. In 2017, this resulted in CIC-IDS2017 [30], which uses a new flow-level feature extractor CICFlowmeter, which outputs 80 features. The variety of attacks was increased and the labeling much more granular (individual attacks within 6 broad attack classes). The same attack class divisions were kept in next year's CSE-CIC-IDS2018 [30]. The CIC collection also includes the specialized CIC-DoS2017 [12] and CIC-DDoS2019 [31] datasets with even more attacks in their respective classes.

Thanks in part to the quality, ease of access and ease of use of the CIC collection, adoption has grown rapidly. For the foreseeable future, these datasets will see further adoption until they displace NSL-KDD.

CTU-13. The technical university of Prague published a specialized intrusion detection dataset in 2014 [8], centered around 13 captures of botnets, intermixed with normal and background traffic. The traffic is intricately labeled at flow-aggregation level. The flow extraction was done by the open source tool Argus. 9 flow-based features were captured and 1 label (plus 4 metadata features).

UNSW-NB15. The university of New South Wales has arguably published the second most widely adopted intrusion detection dataset [21]. The authors specifically set out to modernize the IDS data landscape, in response to the obsolescence of the attacks in the KDD collection. The dataset is geared towards machine learning, with 49 features (4 metadata, 2 labels and 43 content). Like NSL-KDD, the dataset authors published designated train and test sets to encourage researchers to publish results at least starting from the same data selection. These pre-selections no longer include destination port as a feature (though its presence is still reflected in the service feature).

CIDDS Collection. The CIDDS dataset collection (CIDDS-001 [25] & CIDDS-002 [26]), published by the university of Coburg in 2017 is less well-known, but is quite similar to CIC-IDS2017 in terms of experimental aim and methodology. A potential reason for its lack of adoption may lie in the amount of features it includes. Out of 16 total features, 5 are explicitly metadata and 4 are related to labeling, leaving only 7 features to learn from.

2.3 Proposed ML-IDS Systems

All datasets in Subsect. 2.2 contain destination port or equivalent information. The metadata inclusion does allow researchers more flexibility, but also places the burden of selecting the a priori features to remove with them. Invariably, this leads to proposals which keep certain metadata features, while others do not. In Table 1, we outline recent, (often well-cited) IDS proposals validated on any of the aforementioned datasets which included (at least) destination port (or service, its equivalent).

Some of the proposed methods tried to eliminate the metadata features, but failed to accomplish this because they added newly engineered features based on the original metadata. For newly engineered features, computed from the flow features after grouping by IP addresses in the dataset [5,23], it should not be surprising that the results are excellent. The experimental setups for data generation typically have no overlap in the roles for their nodes. A node's role is either attacker or victim (or in some datasets: other infrastructure) and maintains this role throughout the data generation process.

3 Evaluation Methodology

The methodology is straightforward, all datasets (as downloaded straight from their sources) underwent preprocessing including: ensuring data integrity with regard to expected feature types (as described by the dataset authors) and removing invalid and duplicate samples. During this preprocessing stage, maximal attention was given to clean up dataset quirks while retaining as many samples as possible. During this stage, the entire feature set (i.e. including metadata features) was taken into account when evaluating sample uniqueness. To

Table 1. Published articles on the included datasets which included metadata features in their models

Ref	Year	Citations	Metadata inclusion
NSL-KDD			
[13]	2017	8	Service (numerical encoding)
[33]	2020	75	Service (OHE encoding)
[7]	2020	7	Feature-selected service
[41]	2020	169	Feature selection method found service as a top feature
[38]	2021	6	Service (OHE encoding), kept after feature selection
[20]	2021	7	Service
ISCX-IDS2012			
[39]	2013	102	All metadata features
[3]	2015	28	Only metadata features
[6]	2017	32	Leaky new features (time-aggregation) metadata features included
[5]	2018	7	Leaky new features, derived from IP metadata
[19]	2021	6	All metadata features
CTU-13			
[6]	2017	32	All metadata features
[27]	2018	15	Categorical metadata features (IPs, service/port)
[29]	2019	14	All metadata features
[23]	2019	21	Leaky new features, aggregated from IP metadata
UNSW-NB15			
[11]	2017	106	Service
[14]	2019	29	Service
[40]	2019	21	All metadata features
[16]	2020	37	Service
[20]	2021	7	Service
[24]	2021	0	Service
CIDDS-001/CIDDS-002			
[2]	2018	59	All metadata features
[36]	2018	87	All metadata features
[1]	2019	94	All metadata features
[10]	2019	3	All metadata (except 'date first seen')
[35]	2020	23	Source and destination port
CIC-IDS2017			
[18]	2020	57	Destination port
[41]	2020	169	Feature selection method found destination port as a top feature
[20]	2021	7	Several metadata features
CSE-CIC-IDS2018			
[15]	2020	63	At least destination port
[17]	2020	73	Destination port
CIC-DDoS2019			
[28]	2020	3	Destination port
[24]	2021	0	Timestamp and source port
[37]	2021	1	Destination port

be as transparent as possible, the preprocessing code and datasets are publicly available at https://gitlab.ilabt.imec.be/lpdhooge/ids-metadata-contamination.

The actual experiments use the preprocessed datasets and add three specific preprocessing steps. After the data is loaded, only the destination port and the

target (label) are kept. The label is binarized which collapses all attacks into one label. Subsequently, the dataset is balanced with regard to the label before splitting into train and test sets.

The remainder (remaining samples of the majority class) is kept as a secondary verification for the trained model. The balancing is done so that the standard classification metrics remain informative. For the remainder, accuracy $\frac{TP+TN}{TP+TN+FP+FN}$ is reported. However, if the majority of the samples in the subset was malicious (positive class), then the remainder only contains malicious samples and the evaluation metric simplifies to the true positive rate (i.e. recall $\frac{TP}{TP+FN}$). Conversely, if the majority of the samples in the subset was benign, then the metric for the remainder simplifies to the true negative rate ($\frac{TN}{TN+FP}$).

Representation of attack classes was considered sufficient if at least 200 samples exist in the dataset. This does lead to some missing results in the tables. No feature scaling is applied, because the selected ML algorithm does not require uniform ranges for the features. If scaling were to be applied, then the choice to balance the datasets before scaling would bias the scaling parameters towards the minority class. Two train-test split options have been tested (80–20 and 20–80). The 80-train, 20-test split typically outperforms the 20–80 split, but the difference is often marginal.

The selected algorithm was a random forest with fixed hyperparameters (most importantly, maximally 16 levels deep, 10 individual estimators, each given access to a different 50% of the training set). One could optimize these parameters further (especially the depth can be reduced even below 8 without significant performance loss). The splitting mechanism of decision trees is ideal to clearly separate a categorical feature such as destination port. Regression models (linear or otherwise) with 1 feature (i.e. coefficient) are underparametrized to capture the distribution. That shortcoming could be alleviated by one-hot-encoding the destination port to create a separate binary feature for every unique value. Modern, overparametrized models such as the various flavors of neural networks would be able to model the distribution, even if not one-hot-encoded, but at a significantly higher computational cost.

For every dataset, 10 independent iterations have been completed and the reported classification metrics are averages with standard deviations.

3.1 Two Notes on Methodological Design Choices

Time-Aware Train and Test Sets. Most datasets include a timestamp or another time-related feature that could be used to create time-aware data splits. The importance of such time-aware splits to combat concept drift and to reach greater real-world generalization has been well-established in malware classification [22]. However, in the authors' view, current intrusion detection datasets have issues in this regard when compared to malware detection datasets. Malware datasets have multi-year histories which do give rise to actual concept drift in the form of malware (family) mutations as well as fully novel samples and families. Intrusion detection datasets are typically gathered from one-off experiments where the variety of attacks is much lower (a handful of representatives per

attack class), where there are no variations of the included attacks and where the time span is at most a couple of weeks. Due to a lack of interoperability between the academic IDS datasets, expanding variety and time frames through combination is impossible. The authors do concede that using time-aware dataset splits would partially eliminate the contaminating influence of destination port, especially in the combined datasets with multiple attack classes all squeezed into one label *malicious*.

The Drawbacks of Balancing. Two issues raised in the "Dos and Don'ts of Machine Learning in Computer Security" by Arp et al. [4] are linked to our choice to balance the datasets to keep standard classification metrics informative. Creating class-balanced train and test sets does avoid accuracy from becoming an inappropriate measure of performance, but the balancing leads the evaluation further away from the real-world and potentially under-estimates the false positive rate by ignoring the base rate of benign traffic. The same issue of bias in the evaluation due to imbalanced testing (away from a real-world estimate of the true class balance) was also identified in [22].

4 Results

The results are presented, grouped per dataset or per collection of datasets when appropriate. The textual summaries and explanations with regard to the expectedness of the results accompany the tabular formats for the same data. The results conclude with a Subsect. 4.7 dedicated to the contaminating influence of other metadata features in the IDS datasets.

4.1 NSL-KDD

When adhering to the author-provided train-test split for NSL-KDD (Table 2), the performance of the service (i.e. destination port) model is 75% accurate on the test set. It is held back by its recall (65.6%), not by its precision (89.8%) indicating that the authors added attacks in the test set which targeted different services. Still, the single feature metadata model managed to score 50% better than random. Dataset users who do not adhere to the designated train-test split, can expect the destination port model to rise to 90% accuracy with recall and precision now equally at 89–90% (regardless of 80–20 or 20–80 train-test split). Researchers who still use NSL-KDD should therefore stick to the author-provided train-test split, since that split is not merely the result of sampling the entire dataset.

4.2 ISCX-IDS2012

For ISCX-IDS2012, the individual subsets have been tested, as well as the concatenated total dataset. The subsets represent different classes of attacks

(Table 2 HTTP-DoS, DDoS by botnet, SSH brute forcing and infiltration from within the network).

There are huge differences in the discriminative power of destination port between the subsets of ISCX-IDS2012. For the brute force and infiltration subsets, accuracy varies between 95 and 100%. One of the subsets for which the authors state that it contains no attacks, does contain more than 2000 attack samples and it too can be classified with an accuracy well above 99%.

This contrasts sharply with the results on the HTTP-DoS and DDoS (by botnet) subsets for which classification accuracy is no more than 66%. This is easily explainable when grouping the data by class and listing their distributions. The HTTP-DoS attacks predictably target port 80 and port 443. The DDoS attacks also overwhelmingly targeted port 80 (32034/32183 positive samples).

The generated benign traffic always includes *browsing* by simulated users, which means that ports 80 and 443 have significant representation. This highlights the futility of using destination port. If the benign and attack traffic are kept proportional for common ports, then the feature is useless. However, if this is not guaranteed, then the feature reveals itself to be an ideal shortcut by which the classification performance gets overstated.

4.3 CTU-13

The CTU-13 (Table 3) dataset captured 7 distinct botnets in various configurations for a total of 13 scenarios. Some results are missing, but this is due to the authors' decision that subsets with less than 200 samples of the minority class are not evaluated further. Destination port by itself yields 4 models with 70–80% accuracy, 3 with 80–90% accuracy and 2 with 90–100% accuracy. The dataset authors provide a table with details about which botnet behaviors and targets were activated in each scenario. Linking these with the results for destination port as a single predictor, it is revealed that the best classification scores, obtained on scenarios 3, 5, 8, 9 (arguably) and 13, all used the botnet for port scanning. The inherent scatter of port scanning compared to the focused user simulation, yields an easily separable distribution.

4.4 UNSW-NB15

UNSW-NB15, like NSL-KDD, has designated train and test sets. The authors realized the potential for destination port to serve as a shortcut and omitted it as a feature from the designated sets. The dataset is also published in full format which plenty of researchers have used instead of the preselected train and test sets. That rawer version does contain destination port as a feature and when kept, can single-handedly perform 50% better than random (Table 3).

Although the preselected sets no longer have destination port, they (again like NSL-KDD) do still include a feature named service. The labeling for this feature is less granular when compared to its labeling in NSL-KDD, many flows have an empty service label. Solely relying on service to classify the designated test set yields a meager 56.7% accuracy. The drop in performance is not attributable

to the sparse labeling of the service field. When the samples with empty service values are removed a new model is trained on the preselected training set and tested on the preselected test set, it is still just 55% accurate.

The difference is attributable to the different distributions for the same feature (for both targets), between the designated train and test selections. This will lead to poor generalization, which is painfully obvious when using a metadata feature as the only predictor. However, if these distributional shifts are present enough for the other features, then performance will always be weakened. A random forest suffers hard from these distributional shifts, but the loss of performance would be observable in any model. If dataset authors provide preselected train and test splits, they should be wary of this distributional issue.

4.5 CIDDS Collection

CIDDS-001. The creation methodology of CIDDS-001 included a completely controlled internal network with multiple segments and a publicly available external server. The internal network is attacked by machines connected on the same internal infrastructure. The external server is also attacked by author-controlled attackers outside of the internal network. The external server exposes services on the internet (on ports 80 and 443) and as such may have been probed and or attacked during the dataset collection period. The authors made this choice consciously, but also acknowledge that this has implications for the labeling of the dataset. Only the attacks executed by the dataset authors were explicitly labeled as such.

The vast majority of the samples is collected on the internal network (99+%, ~5.5 million samples). A destination port model is more than 70% accurate on the internal network and at least 85% accurate on the external network (Table 3). The added *confusion* on the internal network is again due to the mixture of benign and malicious traffic on port 80. Even though the authors (probably without direct intention) balanced somewhat on port 80, they did not for port 443. Port 443 is the most common port for benign traffic in the internal dataset (600000+ samples), but the attack frequency against the same service is vanishingly small (< 2500) samples.

CIDDS-002. The CIDDS-002 documentation and publication explicitly mention that the dataset is a port scanning dataset. Compared to CIDDS-001, there is no external network anymore. All attacks are executed within the internal network architecture set up in earlier work. All port scanning was done with the nmap tool and the dataset authors list the scanning type (protocol) and timing controls as the primary sources of variance between the attacks. Predictably, destination port is a serious contaminant in CIDDS-002, yielding models with accuracies in excess of 96% (Table 3).

4.6 CIC Collection

The CIC data collection is displacing the older datasets as the new data of choice to model and compare ML-based network intrusion detection systems. Samples from a range of attack classes and normal user behavior are included in the main IDS datasets (2017 & 2018) and the pure network attacks DoS (2017) and DDoS (2019) have special, augmented representations in the form of specialty datasets. Yet, despite the advances made by the CIC research team to improve the state-of-the-art in dataset generation, the datasets are extremely vulnerable to shortcut learning through the destination port metadata feature (Table 4).

CIC-IDS2017. Starting with CIC-IDS2017 and its seven attack classes (of which six are included because infiltration only has 36 positive samples in CIC-IDS2017), classification of a destination port model on the whole dataset is 91.8% accurate. At the individual attack class resolution level, the accuracy of destination port models is exaggerated further beyond 95–99%.

CIC-DoS2017. The CIC-DoS2017 dataset is impacted to a lesser degree, because its attacks directly target web servers on port 80 and a tiny fraction targeted at port 443, both of which also had been included in the user simulation. Still the destination port model can 100% reliably predict benign on all other ports, ultimately leading to an accuracy of 71.6%.

CSE-CIC-IDS2018. The follow-up CIC-IDS dataset, was released one year after CIC-IDS2017 and includes the same attack categories with better representation and execution on Amazon Web Services (AWS) infrastructure. Unfortunately, all attack classes except infiltration are very vulnerable to shortcut learning if the destination port is part of the model. By itself, it can predict the entire dataset with 90% accuracy and within the subsets, the accuracy of the feature regularly jumps to 99.9+%. Only for infiltration is the accuracy far below 90%, reaching just 55–65%. Upon inspection of the data broken down by label, the five most common ports for both subsets of infiltration attacks (53, 443, 3389, 445 and 80) are shared at about equal frequency by the samples for both targets. Those samples represent 75% of the samples of infiltration subset 1 and 66% of the samples of infiltration subset 2.

CIC-DDoS2019. The variance in CIC-DDoS2019 is captured with 91.5% accuracy by a destination port model. The individual attack sets of CIC-DDoS2019 are captured even better, often reaching 95–99% accuracy. Any proposed models that included the feature should be dismissed out-of-hand because this feature by itself can generate a near-perfect model.

Table 2. Results of the pure destination port (service) models on NSL-KDD and ISCX-IDS2012.

Subset	Accuracy	Precision	Recall	Remainder acc
NSL-KDD				
Designated-train-test-split	75.4 ± 0.9%	89.8 ± 0.1%	65.6 ± 1.6%	–
Straight-sampling	90.0 ± 0.2%	89.9 ± 1.2%	90.1 ± 1.5%	–
ISCX-IDS2012				
HTTP-DoS	67.9 ± 1.3%	94.5 ± 0.9%	37.5 ± 2.2%	97.8 ± 0.3%
Unspecified	99.7 ± 0.2%	99.5 ± 0.3%	100 ± 0.0%	99.3 ± 0.0%
Infiltration	95.3 ± 0.3%	98.4 ± 0.2%	92.2 ± 0.5%	98.4 ± 0.1%
Brute force	100 ± 0.0%	100 ± 0.0%	100 ± 0.0%	100 ± 0.0%
DDoS	57.9 ± 0.3%	54.4 ± 0.3%	100 ± 0.0%	15.9 ± 0.0%
Combined	66.3 ± 0.2%	96.4 ± 0.4%	34.0 ± 0.4%	98.7 ± 0.1%

4.7 Other Metadata Features

Although they are much more frequently stripped from evaluation, as they should be, other metadata features are present in the IDS datasets. This subsection summarizes their effectiveness as shortcut features across the included datasets. Full numerical results for these other metadata features are stored in the online repository for this experiment.

Source IP. Source IP addresses are easily recognized as a source of contamination. IDS datasets typically have defined, disjoint sets of nodes that will function either as attacker or as target. Unsurprisingly, ML methods quickly pick up on this pattern, often leading to perfect models for the included data sets. Deriving new features after grouping by source IP (blocks) should also be discouraged, because the composition of those groups will be heavily skewed and leak into the derived features.

Source Port. Source port should not be considered as a genuine feature, because the choice is determined by the operating system (OS). Furthermore, the range of available ports for use as source port OS-specific. On the author's personal Linux system, this range encompasses 32768 to 60999 by default. Conversely, on the author's Windows system, this range starts at 1025 and ends at 65535. At worst, if the data generation design used only Linux systems as attackers and only Windows systems as targets, the non-overlapping portions of these ranges would immediately identify a node's role. In the included datasets, using just the source port as a predictor leads to models with 60 to 90+% accuracy.

Table 3. Results of the pure destination port (service) models on CTU-13, UNSW-NB15 and the CIDDS collection. Po = Portscan, Br = Bruteforce, Pi = Pingscan

Scenario	Accuracy	Precision	Recall	Remainder acc
CTU-13				
1. Neris	79.3 ± 0.4%	72.1 ± 0.7%	95.4 ± 0.3%	63.2 ± 0.0%
2. Neris	88.1 ± 0.6%	86.8 ± 1.0%	89.7 ± 0.7%	86.5 ± 0.1%
3. Rbot	99.2 ± 0.2%	98.9 ± 0.4%	99.5 ± 0.3%	98.4 ± 0.0%
5. Virut	84.5 ± 1.3%	89.0 ± 0.4%	80.8 ± 11.3%	89.2 ± 8.9%
8. Murlo	96.6 ± 0.8%	99.4 ± 0.5%	93.7 ± 1.4%	99.4 ± 0.1%
9. Neris	71.1 ± 0.2%	65.1 ± 0.3%	91.3 ± 0.1%	51.1 ± 0.0%
12 .NSIS.ay	74.9 ± 1.5%	72.7 ± 1.8%	79.4 ± 2.5%	71.0 ± 0.3%
13. Virut	83.7 ± 0.2%	96.0 ± 0.3%	70.2 ± 0.5%	97.0 ± 0.0%
Combined	70.6 ± 0.2%	64.9 ± 0.2%	90.0 ± 0.2%	51.1 ± 0.0%
UNSW-NB15				
Raw UNSW-NB15	75.7 ± 0.2%	73.3 ± 4.0%	82.3 ± 9.9%	69.2 ± 10.0%
CIDDS-001				
External-2 (Po,Br)	89.7 ± 0.7%	92.6 ± 0.%	86.0 ± 1.3%	93.3 ± 0.3%
External-3 (Po,Br)	85.9 ± 0.4%	85.1 ± 0.7%	87.2 ± 0.7%	84.7 ± 0.2%
External-4 (Po,Br)	86.1 ± 1.8%	90.7 ± 1.9%	80.4 ± 0.%	91.6 ± 1.1%
Internal-1 (Po,Pi,DoS,Br)	71.0 ± 0.0%	68.0 ± 0.1%	79.3 ± 0.3%	62.6 ± 0.3%
Internal-2 (Po,Pi,DoS,Br)	70.7 ± 0.1%	66.4 ± 0.1%	84.0 ± 0.2%	57.3 ± 0.2%
Internal-combined	71.2 ± 0.1%	67.6 ± 0.1%	81.6 ± 0.1%	60.9 ± 0.1%
External-combined	86.4 ± 0.4%	85.9 ± 0.5%	87.1 ± 0.4%	85.7 ± 0.1%
Combined	71.2 ± 0.0%	67.6 ± 0.1%	81.4 ± 0.1%	61.1 ± 0.0%
CIDDS-002				
Internal-1 (Po)	97.3 ± 0.0%	98.8 ± 0.0%	95.9 ± 0.1%	98.8 ± 0.0%
Internal-2 (Po)	96.6 ± 0.1%	98.3 ± 0.1%	94.8 ± 0.2%	98.3 ± 0.0%
Combined	96.3 ± 0.5%	98.9 ± 0.1%	93.7 ± 0.9%	98.9 ± 0.1%

Table 4. Results of the pure destination port (service) models on the CIC collection.

Subset	Accuracy	Precision	Recall	Remainder acc
CIC-IDS2017				
Botnet	97.4 ± 0.6%	95.9 ± 1.4%	99.1 ± 0.4%	96.1 ± 0.3%
Bruteforce	99.6 ± 0.0%	99.3 ± 0.1%	100 ± 0.0%	99.3 ± 0.0%
DDoS	95.5 ± 0.1%	91.7 ± 0.2%	100 ± 0.0%	100 ± 0.0%
DoS	94.5 ± 0.0%	90.0 ± 0.1%	100 ± 0.0%	88.9 ± 0.0%
Portscan	98.4 ± 0.1%	98.2 ± 0.2%	98.6 ± 0.1%	98.4 ± 0.1%
Webattacks	95.0 ± 0.9%	90.9 ± 1.7%	100 ± 0.0%	88.9 ± 0.0%
Combined	91.8 ± 0.0%	86.6 ± 0.1%	99.1 ± 0.0%	84.6 ± 0.0%

(*continued*)

Table 4. (*continued*)

Subset	Accuracy	Precision	Recall	Remainder acc
CIC-DoS2017				
DoS	71.6 ± 0.3%	63.8 ± 0.3%	100 ± 0.0%	42.9 ± 0.0%
CSE-CIC-IDS2018				
Botnet	98.3 ± 0.0%	98.6 ± 0.2%	98.1 ± 0.1%	98.6 ± 0.1%
Bruteforce	100 ± 0.0%	99.9 ± 0.0%	100 ± 0.0%	99.9 ± 0.0%
DDoS-1	91.0 ± 0.1%	84.7 ± 0.1%	100 ± 0.0%	100 ± 0.0%
DDos-2	99.8 ± 0.0%	99.6 ± 0.0%	100 ± 0.0%	99.6 ± 0.0%
DoS-1	91.4 ± 0.2%	85.4 ± 0.4%	100 ± 0.0%	82.7 ± 0.0%
DoS-2	100 ± 0.0%	100 ± 0.0%	100 ± 0.0%	100 ± 0.0%
Infiltration-1	55.2 ± 0.3%	53.7 ± 0.5%	79.1 ± 1.1%	30.9 ± 1.2%
Infiltration-2	63.7 ± 0.3%	80.8 ± 0.6%	36.2 ± 0.6%	91.5 ± 0.4%
Webattacks-1	94.3 ± 2.7%	90.0 ± 4.2%	100 ± 0.0%	84.2 ± 0.0%
Webattacks-2	90.7 ± 1.6%	84.3 ± 2.3%	99.6 ± 0.5%	82.8 ± 0.2%
Combined	89.7 ± 0.0%	87.0 ± 0.1%	93.3 ± 0.0%	86.1 ± 0.0%
CIC-DDoS2019				
DNS	99.0 ± 0.2%	99.3 ± 0.2%	98.7 ± 0.5%	97.6 ± 0.0%
LDAP-1	99.3 ± 0.4%	99.7 ± 0.4%	98.8 ± 0.8%	98.9 ± 0.4%
MSSQL-1	99.1 ± 0.3%	99.5 ± 0.4%	98.6 ± 0.6%	100 ± 0.0%
NETBIOS-1	99.1 ± 0.3%	99.5 ± 0.5%	98.7 ± 0.5%	100 ± 0.0%
NTP	97.3 ± 0.2%	99.1 ± 0.2%	95.5 ± 0.4%	94.8 ± 0.6%
SNMP	98.5 ± 0.7%	98.9 ± 0.8%	98.1 ± 0.9%	97.7 ± 0.4%
SSDP	99.2 ± 0.5%	99.2 ± 0.7%	99.1 ± 0.6%	100 ± 0.0%
UDP-1	99.2 ± 0.2%	99.5 ± 0.2%	98.8 ± 0.3%	100 ± 0.0%
SYN-1	98.1 ± 1.1%	99.3 ± 0.7%	97.0 ± 1.9%	100 ± 0.0%
TFTP	96.3 ± 0.2%	98.8 ± 0.1%	93.7 ± 0.4%	99.0 ± 0.4%
UDPLAG-1	99.0 ± 0.2%	99.6 ± 0.2%	98.2 ± 0.4%	99.4 ± 0.1%
LDAP-2	94.3 ± 0.4%	92.8 ± 0.6%	96.0 ± 0.7%	93.9 ± 1.3%
MSSQL-2	96.1 ± 0.4%	95.0 ± 0.8%	97.5 ± 0.7%	92.3 ± 0.0%
NETBIOS-2	94.9 ± 0.9%	94.5 ± 1.4%	95.2 ± 1.7%	98.6 ± 2.2%
PORTMAP	97.6 ± 0.3%	97.1 ± 0.4%	98.0 ± 0.5%	97.1 ± 0.7%
SYN-2	92.6 ± 0.2%	91.7 ± 0.5%	93.6 ± 0.3%	93.6 ± 0.3%
UDPLAG-2	96.2 ± 0.3%	95.9 ± 0.5%	96.5 ± 0.7%	97.7 ± 0.4%
UDP-2	96.6 ± 0.8%	96.0 ± 1.2%	97.2 ± 0.6%	95.6 ± 5.7%
Combined	91.7 ± 0.1%	91.0 ± 0.2%	92.5 ± 0.3%	92.9 ± 0.3%

Destination IP. Including destination IPs runs into a similar risk as including source IPs. Certain portions of the network used for the data generation process are much more likely to be the target of attacks while other parts are always part of benign flows. The most prominent example would be malicious nodes which always attack on the internal network, while the benign data generation contacts servers outside the range of nodes controlled by the dataset authors (e.g. browsing the internet or exchanging emails). Again, for the included datasets, the effectiveness of just using destination IP ranges from 60 to 99+% accuracy.

Metadata Time Features. The labeling strategy of IDS datasets is typically based on a combination of strict role definition for nodes in the experimental network and precise bookkeeping of attack time frames. The CIC datasets have a Timestamp feature, CIDDS has a feature called date_first_seen, CTU-13 has starttime, ISCX-IDS2012 and UNSW-NB15 have start_date_time (stime) and stop_date_time (ltime). When using these features as single predictors, models with accuracies of 70% (UNSW-NB15) to 85% and above (all other datasets) can be expected. The models simply learn the time frames of the attacks.

5 Discussion

Performance on individual datasets and subsets (attack classes) varies, but the general trend is clear. Destination port (or its proxy service) can yield models which separate benign from malicious traffic with accuracies significantly better than random guessing. Put in terms of accuracy, the performance of the vast majority of models ranges between 70 and 100% accuracy.

The low end of this performance range is heavily correlated with greater overlap between the services that are attacked and which are also well-represented in user emulation. The top end of the performance range is characterized by models that were tasked with learning dataset (subsets, i.e. attack classes) where the attacks are scattered across many ports while the user emulation keeps to its narrow selection of emulated services.

Models that have included destination port suffer harder on datasets with pre-selected train and test splits, NSL-KDD and UNSW-NB15. The reason is that the dataset authors did not just take (stratified) samples from their total sample collection, but actively introduced new attacks (on unseen ports) in the test sets. These distributional differences between the preselected train/test sets are particularly detrimental for a categorical feature like destination port. It is yet another reason why it should not be included as a feature.

6 Conclusion

Modern network intrusion datasets, intended to be modeled by machine learning techniques, are often published with the inclusion of metadata features such as source and destination IP addresses, ports and time-based features used for

sample labeling. Of these metadata features, destination port is most likely to be kept by researchers when training their models under the rationale that it contains useful information to identify service-specific attacks. This short article aims to demonstrate that the destination port or its proxy feature service are in fact contaminants that result in shortcut learning. This claim is demonstrated by showing the discriminative power of this feature by itself in models trained to classify ten state-of-the-art IDS datasets.

There is variability between the datasets, but the standard outcome is that destination port by itself can be used to generate models that are 70 to 100% accurate. The datasets in descending order of vulnerability to this effect are CIC-DDoS2019, CSE-CIC-IDS2018, CIC-IDS2017, CIDDS-002, UNSW-NB15, CIC-DoS2017, CIDDS-001, CTU-13 and ISCX-IDS2012.

The end result is a paradox for dataset publishers. If they equalize the proportion of benign and malicious samples on all represented ports, then the discriminating power of the feature is no better than random guessing. If they do not equalize representation across the ports, then the feature quickly rises to yield excellent, but ultimately meaningless models.

Based on this information, we propose that future dataset publications for intrusion detection omit this feature altogether. Reviewers evaluating newly proposed methods validated on the current datasets should inspect the list of included features. If destination port (or service) remained in the dataset, then the results have a near certain probability of being too optimistic. This potential for (unintended) inflation of the results is particularly poignant in this research domain where new proposals are considered more meritorious if they can demonstrate an increase in classification performance beyond the state-of-the-art.

7 Future Work

Settling the theoretical debate about whether intrusion detection models should include samples' destination port as a feature was the obvious first step in trying to improve the data handling by users of the state-of-the-art IDS datasets.

The conclusion of this article primarily targets other IDS researchers to interact more critically with the available datasets. In a secondary capacity it is addressed to dataset authors in the field to consider omitting the feature when publishing data.

The follow-up to this article will focus on finding less obvious contaminants in the state-of-the-art IDS datasets. We have already identified several high-impact features in the state-of-the-art datasets which may be the result of experimental design and execution, rather than a consequence of real attack behavior. The end goal is to also eliminate these features from the datasets to try to increase robustness and generalizability of ML-IDS models.

References

1. Abdulhammed, R., Faezipour, M., Abuzneid, A., AbuMallouh, A.: Deep and machine learning approaches for anomaly-based intrusion detection of imbalanced network traffic. IEEE Sensors Lett. **3**(1), 1–4 (2019). https://doi.org/10.1109/LSENS.2018.2879990
2. Althubiti, S.A., Jones, E.M., Roy, K.: LSTM for anomaly-based network intrusion detection. In: 2018 28th International Telecommunication Networks and Applications Conference (ITNAC), pp. 1–3 (2018). https://doi.org/10.1109/ATNAC.2018.8615300
3. Ammar, A., et al.: A decision tree classifier for intrusion detection priority tagging. J. Comput. Commun. **3**(04), 52 (2015)
4. Arp, D., et al.: Dos and don'ts of machine learning in computer security. In: Proceedings of the USENIX Security Symposium (2022)
5. Atli, B.G., Miche, Y., Jung, A.: Network intrusion detection using flow statistics. In: 2018 IEEE Statistical Signal Processing Workshop (SSP), pp. 70–74 (2018). https://doi.org/10.1109/SSP.2018.8450709
6. Bansal, A., Mahapatra, S.: A comparative analysis of machine learning techniques for botnet detection. In: Proceedings of the 10th International Conference on Security of Information and Networks, SIN 2017, pp. 91–98. Association for Computing Machinery, New York (2017). https://doi.org/10.1145/3136825.3136874
7. Farhat, S., Abdelkader, M., Meddeb-Makhlouf, A., Zarai, F.: Comparative study of classification algorithms for cloud ids using nsl-kdd dataset in weka. In: 2020 International Wireless Communications and Mobile Computing (IWCMC), pp. 445–450 (2020). https://doi.org/10.1109/IWCMC48107.2020.9148311
8. García, S., Grill, M., Stiborek, J., Zunino, A.: An empirical comparison of botnet detection methods. Comput. Secur. **45**, 100–123 (2014)
9. Geirhos, R., et al.: Shortcut learning in deep neural networks. Nat. Mach. Intell. **2**(11), 665–673 (2020). https://www.proquest.com/scholarly-journals/shortcut-learning-deep-neural-networks/docview/2621045756/se-2?accountid=11077, Springer Nature Limited 2020, Accessed 19 Jan 2022
10. He, W., Li, H., Li, J.: Ensemble feature selection for improving intrusion detection classification accuracy. In: Proceedings of the 2019 International Conference on Artificial Intelligence and Computer Science, AICS 2019, pp. 28–33. Association for Computing Machinery, New York (2019). https://doi.org/10.1145/3349341.3349364
11. Janarthanan, T., Zargari, S.: Feature selection in unsw-nb15 and kddcup'99 datasets. In: 2017 IEEE 26th International Symposium on Industrial Electronics (ISIE), pp. 1881–1886 (2017). https://doi.org/10.1109/ISIE.2017.8001537
12. Jazi, H.H., Gonzalez, H., Stakhanova, N., Ghorbani, A.A.: Detecting http-based application layer dos attacks on web servers in the presence of sampling. Comput. Netw. **121**, 25–36 (2017)
13. Ji, H., Kim, D., Shin, D., Shin, D.: A study on comparison of KDD CUP 99 and NSL-KDD using artificial neural network. In: Park, J.J., Loia, V., Yi, G., Sung, Y. (eds.) CUTE/CSA -2017. LNEE, vol. 474, pp. 452–457. Springer, Singapore (2018). https://doi.org/10.1007/978-981-10-7605-3_74
14. Jing, D., Chen, H.B.: SVM based network intrusion detection for the unsw-nb15 dataset. In: 2019 IEEE 13th International Conference on ASIC (ASICON), pp. 1–4 (2019). https://doi.org/10.1109/ASICON47005.2019.8983598

15. Karatas, G., Demir, O., Sahingoz, O.K.: Increasing the performance of machine learning-based idss on an imbalanced and up-to-date dataset. IEEE Access **8**, 32150–32162 (2020). https://doi.org/10.1109/ACCESS.2020.2973219
16. Kasongo, S.M., Sun, Y.: Performance analysis of intrusion detection systems using a feature selection method on the UNSW-NB15 dataset. J. Big Data **7**(1), 1–20 (2020). https://doi.org/10.1186/s40537-020-00379-6
17. Kim, J., Kim, J., Kim, H., Shim, M., Choi, E.: CNN-based network intrusion detection against denial-of-service attacks. Electronics **9**(6) (2020). https://doi.org/10.3390/electronics9060916, https://www.mdpi.com/2079-9292/9/6/916
18. Kurniabudi, S.D., Darmawijoyo, Bin Idris, M.Y., Bamhdi, A.M., Budiarto, R.: Cicids-2017 dataset feature analysis with information gain for anomaly detection. IEEE Access **8**, 132911–132921 (2020). https://doi.org/10.1109/ACCESS.2020.3009843
19. Kushwah, G.S., Ranga, V.: Optimized extreme learning machine for detecting ddos attacks in cloud computing. Comput. Secur. **105**, 102260 (2021)
20. Lohiya, R., Thakkar, A.: Intrusion detection using deep neural network with antirectifier layer. In: Thampi, S.M., Lloret Mauri, J., Fernando, X., Boppana, R., Geetha, S., Sikora, A. (eds.) Applied Soft Computing and Communication Networks. LNNS, vol. 187, pp. 89–105. Springer, Singapore (2021). https://doi.org/10.1007/978-981-33-6173-7_7
21. Moustafa, N., Slay, J.: Unsw-nb15: a comprehensive data set for network intrusion detection systems (unsw-nb15 network data set). In: 2015 Military Communications and Information Systems Conference (MilCIS), pp. 1–6 (2015). https://doi.org/10.1109/MilCIS.2015.7348942
22. Pendlebury, F., Pierazzi, F., Jordaney, R., Kinder, J., Cavallaro, L.: Tesseract: eliminating experimental bias in malware classification across space and time. In: Proceedings of the 28th USENIX Conference on Security Symposium, SEC 2019, pp. 729–746. USENIX Association, USA (2019)
23. Piskozub, M., Spolaor, R., Martinovic, I.: Malalert: detecting malware in large-scale network traffic using statistical features. SIGMETRICS Perform. Eval. Rev. **46**(3), 151–154 (2019). https://doi.org/10.1145/3308897.3308961
24. Priya Devi, A., Johnson Singh, K.: A machine learning approach to intrusion detection system using UNSW-NB-15 and CICDDoS2019 datasets. In: Satapathy, S.C., Bhateja, V., Favorskaya, M.N., Adilakshmi, T. (eds.) Smart Computing Techniques and Applications. SIST, vol. 225, pp. 195–205. Springer, Singapore (2021). https://doi.org/10.1007/978-981-16-0878-0_20
25. Ring, M., Wunderlich, S., Grüdl, D., Landes, D., Hotho, A.: Creation of flow-based data sets for intrusion detection. J. Inf. Warfare **16**, 40–53 (2017)
26. Ring, M., Wunderlich, S., Grüdl, D., Landes, D., Hotho, A.: Flow-based benchmark data sets for intrusion detection. In: Proceedings of the 16th European Conference on Cyber Warfare and Security (ECCWS), pp. 361–369. ACPI (2017)
27. Ryu, S., Yang, B., et al.: A comparative study of machine learning algorithms and their ensembles for botnet detection. J. Comput. Commun. **6**(05), 119 (2018). https://doi.org/10.4236/jcc.2018.65010
28. Sbai, O., El boukhari, M.: Data flooding intrusion detection system for manets using deep learning approach. In: Proceedings of the 13th International Conference on Intelligent Systems: Theories and Applications, SITA 2020. Association for Computing Machinery, New York (2020). https://doi.org/10.1145/3419604.3419777

29. Shamshirband, S., Chronopoulos, A.T.: A new malware detection system using a high performance-elm method. In: Proceedings of the 23rd International Database Applications; Engineering Symposium, IDEAS 2019. Association for Computing Machinery, New York (2019). https://doi.org/10.1145/3331076.3331119
30. Sharafaldin., I., Habibi Lashkari., A., Ghorbani., A.A.: Toward generating a new intrusion detection dataset and intrusion traffic characterization. In: Proceedings of the 4th International Conference on Information Systems Security and Privacy - ICISSP, pp. 108–116. INSTICC, SciTePress (2018). https://doi.org/10.5220/0006639801080116
31. Sharafaldin, I., Lashkari, A.H., Hakak, S., Ghorbani, A.A.: Developing realistic distributed denial of service (ddos) attack dataset and taxonomy. In: 2019 International Carnahan Conference on Security Technology (ICCST), pp. 1–8 (2019). https://doi.org/10.1109/CCST.2019.8888419
32. Shiravi, A., Shiravi, H., Tavallaee, M., Ghorbani, A.A.: Toward developing a systematic approach to generate benchmark datasets for intrusion detection. Comput. Secur. **31**(3), 357–374 (2012)
33. Su, T., Sun, H., Zhu, J., Wang, S., Li, Y.: Bat: deep learning methods on network intrusion detection using nsl-kdd dataset. IEEE Access **8**, 29575–29585 (2020). https://doi.org/10.1109/ACCESS.2020.2972627
34. Tavallaee, M., Bagheri, E., Lu, W., Ghorbani, A.A.: A detailed analysis of the kdd cup 99 data set. In: 2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications, pp. 1–6 (2009). https://doi.org/10.1109/CISDA.2009.5356528
35. Thapa, N., Liu, Z., KC, D.B., Gokaraju, B., Roy, K.: Comparison of machine learning and deep learning models for network intrusion detection systems. Fut. Internet **12**(10) (2020). https://doi.org/10.3390/fi12100167, https://www.mdpi.com/1999-5903/12/10/167
36. Verma, A., Ranga, V.: Statistical analysis of cidds-001 dataset for network intrusion detection systems using distance-based machine learning. Procedia Comput. Sci. **125**, 709–716 (2018). https://doi.org/10.1016/j.procs.2017.12.091, https://www.sciencedirect.com/science/article/pii/S1877050917328594
37. Vuong, T.-H., Thi, C.-V.N., Ha, Q.-T.: N-tier machine learning-based architecture for DDoS attack detection. In: Nguyen, N.T., Chittayasothorn, S., Niyato, D., Trawiński, B. (eds.) ACIIDS 2021. LNCS (LNAI), vol. 12672, pp. 375–385. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-73280-6_30
38. Xu, W., Jang-Jaccard, J., Singh, A., Wei, Y., Sabrina, F.: Improving performance of autoencoder-based network anomaly detection on nsl-kdd dataset. IEEE Access **9**, 140136–140146 (2021). https://doi.org/10.1109/ACCESS.2021.3116612
39. Yassin, W., Udzir, N.I., Muda, Z., Sulaiman, M.N., et al.: Anomaly-based intrusion detection through k-means clustering and naives bayes classification. In: Proceedings of 4th International Conference on Computer Informatics, ICOCI, vol. 49, pp. 298–303 (2013)
40. Zhiqiang, L., Mohi-Ud-Din, G., Bing, L., Jianchao, L., Ye, Z., Zhijun, L.: Modeling network intrusion detection system using feed-forward neural network using unsw-nb15 dataset. In: 2019 IEEE 7th International Conference on Smart Energy Grid Engineering (SEGE), pp. 299–303 (2019). https://doi.org/10.1109/SEGE.2019.8859773
41. Zhou, Y., Cheng, G., Jiang, S., Dai, M.: Building an efficient intrusion detection system based on feature selection and ensemble classifier. Comput. Netw. **174**, 107247 (2020)