UNR-IDD: Intrusion Detection Dataset using Network Port Statistics

Tapadhir Das $^{1,1,1}$, Osama Abu Hamdan $^2$, Raj Shukla $^2$, Shamik Sengupta $^2$, and Engin Arslan $^2$

$^1$University of Nevada  
$^2$Affiliation not available

October 31, 2023

Abstract

With the expanded applications of modern-day networking, network infrastructures are at risk from cyber attacks and intrusions. Multiple datasets have been proposed in literature that can be used to create Machine Learning (ML) based Network Intrusion Detection Systems (NIDS). However, many of these datasets suffer from sub-optimal performance and do not adequately represent all types of intrusions in an effective manner. Another problem with these datasets is the low accuracy of tail classes. To address these issues, in this paper, we propose the University of Nevada - Reno Intrusion Detection Dataset (UNR-IDD) that provides researchers with a wider range of samples and scenarios. The proposed dataset utilizes network port statistics for more fine-grained control and analysis of intrusions. We provide a benchmark to show efficient performance for both binary and multi-class classification tasks using different ML algorithms. The paper further explains the intrusion detection activities rather than providing a generic black-box output of the ML algorithms. In comparison with the other established NIDS datasets, we obtain better performance with an $F_\mu$ score of 94% and a minimum F score of 86%. This performance can be credited to prioritizing high scoring average and minimum F-Measure scores for modeled intrusions.
UNR-IDD: Intrusion Detection Dataset using Network Port Statistics

Tapadhir Das*, Osama Abu Hamdan*, Raj Mani Shukla†, Shamik Sengupta*, Engin Arslan*  
*Department of Computer Science and Engineering, University of Nevada, Reno, USA  
†School of Computing and Information Science, Anglia Ruskin University, UK  
Email: tapadhird.oabuhamdan@nevada.unr.edu, raj.shukla@aru.ac.uk, ssengupta,earslan@unr.edu

Abstract—Multiple datasets have been proposed to create Machine Learning (ML)-based Network Intrusion Detection Systems (NIDS). However, many of these datasets suffer from suboptimal performance and inadequate tail class representation. In this paper, we propose the University of Nevada - Reno Intrusion Detection Dataset (UNR-IDD), which utilizes network port statistics for fine-grained analysis of intrusions. Evaluation results show that UNR-IDD is better than existing NIDS datasets with an F score of 94% and a minimum F-score of 86%. This is mainly because of sufficient and equal representation of various anomaly types in the UNR-IDD dataset.

Index Terms—Computer networks, network intrusion detection, machine learning, dataset

I. INTRODUCTION

The usage of machine learning (ML) for Network Intrusion Detection Systems (NIDS) has gained traction in the last decade as various open-sourced datasets have been established [1] [2] [3]. However, a commonly identified problem with many of these datasets is inadequate modeling of tail classes [4]. Tail classes refer to labels with fewer samples, leading to poor performance when fitting the ML model. Researchers have been looking at methods to address this issue of tail classes. Common methods include undersampling and over-sampling. However, over-sampling increases the size of the dataset, increasing training time, memory, and complexity. Correspondingly, undersampling can reduce data samples from the majority classes, affecting the overall performance [5]. Other investigated methods include transfer learning, data augmentation, and ensemble methods [4]. Transfer learning and data augmentation can further increase class variability as head classes would be augmented more, while ensemble methods can lead to higher computational costs. Therefore, datasets for NIDS need to prioritize tail class representation during data generation.

Another limitation of current datasets is that they mostly depend on flow-level statistics, which can limit the transferability of the NIDS solutions to other network configurations since flow statistics depend on topology and traffic characteristics. Depending on the ML approach, these features can also increase the cardinality of the dataset. Addressing the above-mentioned limitations is vital for ensuring that proper NIDS are being developed to adequately protect networks from intrusions.

In this paper, we propose the University of Nevada - Reno Intrusion Detection Dataset (UNR-IDD). UNR-IDD consists primarily of network port statistics. These refer to the observed port metrics recorded in switch/router ports within a network. The dataset also includes delta port statistics which indicates the change in magnitude of observed port statistics within a time interval. Compared to flow statistics-based datasets, UNR-IDD can provide a more fine-grained analysis of network flows as decisions are made at the port level versus the flow level, leading to the rapid identification of potential intrusions. Our dataset also ensures that there are enough samples for ML classifiers to achieve high F-Measure scores, for all classes.

The main contributions of this work include:

- Usage of port and delta port statistics under the various intrusion scenarios.
- Presentation of feature importance to gain insights into how different intrusions are affecting various port statistics.
- Performance comparison against other NIDS datasets in terms of the performance of ML-based intrusion detection models.

The rest of the paper is structured as follows: Section II provides the literature study. Our dataset collection, configuration, and generation methods are detailed in Section III. Section IV presents experimental results and discussions. Finally, conclusions are drawn in Section V.

II. RELATED WORK

Some of the very first NIDS datasets include DARPA [6] and KDDCup99 [7]. The main issue with these datasets is that they are outdated and not representative of modern network traffic. Other datasets include CAIDA [8], CDX [9], Kyoto [10], Twente [11], and ISCX2012 [12]. CAIDA dataset suffers from limited features and only detects Distributed Denial of Service (DDoS) attacks. Similarly, the CDX dataset only contains five features and detects buffer overflows. The Kyoto dataset is restricted to two classes and contains limited features. The Twente and ISCX2012 datasets solely focused on IP flows, which restricts their capability as they are flow-oriented and do not provide network-level information that can be used to detect network-wide issues [13].

Three common NIDS datasets are NSL-KDD [1], CIC-IDS-2018 [3], and UNSW-NB15 [2]. These datasets also have several limitations. For instance, the UNSW-NB15 suffers from inconsistent performance for machine learning classifiers. NSL-KDD and CIC-IDS-2018 datasets suffer from missing data samples. Many of these datasets also suffer from the issue of containing inadequately modeled tail classes.
III. UNR-IDD DATASET

We setup our testbed using a Software-defined Network (SDN) simulation environment, Mininet due to the ease of usability and implementation. It also ensures that the dataset is not dependent on any static topology and can be configured to reproduce the network activity of various topologies. Following this, we perform flow simulations within the SDN topology to replicate appropriate functionality. During these flow simulations, desired network statistics are collected, under normal and attack conditions.

A. Testbed Configuration

To set up the testbed, we use Open Network Operating System (ONOS) SDN controller (API version 2.5.0) alongside Mininet for the network topology generation. ONOS uses the Open Service Gateway Initiative (OSGi) service component at runtime for the creation and activation of components and auto-wiring components together, making it easy to create and deploy new user-defined components without altering the core constituents. Mininet creates the desired virtual network, and runs a real kernel, switch, and application code, on a single machine, thereby generating a realistic testbed environment. We also implemented our ONOS application to collect network statistics. Specifically, we gathered delta and cumulative port, flow entry, and flow table statistics for each connected Open vSwitch in the Mininet topology. We created a custom Mininet topology using Mininet API (version 2.3.0) with Open Flow (OF) 14 protocol deployed to the switches. The generated SDN topology for our experiments is illustrated in Figure 1, which consists of 10 hosts and 12 switches.

B. Flow Simulation

IPerf is used to create TCP and UDP data streams simulating network flows in virtual and real networks using dummy payloads. By using the Mininet API and IPerf, we created a Python script to simulate realistic network flows. Once every 5 seconds, we initiated Iperf traffic between a randomly chosen source-destination host pair with a bandwidth of 10 Mbps and duration of 5 seconds. These values must be carefully chosen as they are dependent on the number of nodes, hosts, switches, and geographical spread of the simulated network. We then simulate flows under normal and intrusion conditions to gather data in every scenario. To ensure that each normal and intrusion category is minimally variable and adequately represented, we execute the same number of flows while simulating each scenario.

C. Data Collection

We create a custom application to collect and log the available statistics that are captured periodically (once every 5 seconds) from OpenFlow (OF) switches. The statistics are collected through means of OFPPortStatsRequest and OFPPortStatsReply messages between controller and switches. The delta port statistics are computed on the controller side by taking the difference between the last two collected data instances. We create a key-value map of this data by gathering it from the data storage service, using the "Device Service" API provided by ONOS. After this, we logged the map of the collected statistics to a Javascript Object Notation (.json) file with a name $N_i$. The collected port statistics include Received Packet, Received Bytes, Sent Packets, Sent Bytes, Port alive Duration, Packets Rx Dropped, Packet Tx Dropped, Packet Rx Errors, Packet Tx Errors. These statistics relay the collected metrics and magnitudes from every single port within the SDN when a flow is simulated between two hosts. Similarly, the collected delta port statistics include Delta Received Packet, Delta Received Bytes, Delta Sent Packets, Delta Sent Bytes, Delta Port alive Duration, Delta Packets Rx Dropped, Delta Packet Tx Dropped, Delta Packet Rx Errors, Delta Packet Tx Errors. These delta statistics are used to capture the change in collected metrics from every single port within the SDN when a flow is simulated between two hosts, at a time interval of 5 seconds. Additionally, we also collect some flow entry and flow table statistics to work in conjunction with the collected port statistics, which include Connection Point, Total Load/Rate, Total Load/Latest, Unknown Load/Rate, Unknown Load/Latest, Time seen, is_valid, TableID, ActiveFlowEntries, PacketsLookedUp, PacketsMatched, MaxSize. These metrics provide information about the conditions of switches in the network and can be collected in any network setting.

D. Labels

This dataset can be broken down into two different ML classification problems: binary and multi-class. The goal of binary classification is to differentiate intrusions from normal working conditions. Labels for binary classification are: Normal and Attack.

The goal of multi-class classification is to differentiate the intrusions not only from normal working conditions but also from each other. Multi-class classification helps us to learn about the root causes of network intrusions. Intrusion labels for multi-class classification are TCP-SYN flood, Port scan, Flow table overflow, Blackhole, Diversion. These intrusion types were selected for this dataset as they are common cyber
TABLE I: Multi-class Classification Performance

<table>
<thead>
<tr>
<th>Label</th>
<th>P</th>
<th>R</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>Blackhole</td>
<td>0.98</td>
<td>0.98</td>
<td>0.98</td>
</tr>
<tr>
<td>Diversion</td>
<td>0.99</td>
<td>0.97</td>
<td>0.98</td>
</tr>
<tr>
<td>Normal</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
</tr>
<tr>
<td>Overflow</td>
<td>0.98</td>
<td>0.96</td>
<td>0.86</td>
</tr>
<tr>
<td>PortScan</td>
<td>0.91</td>
<td>0.94</td>
<td>0.92</td>
</tr>
<tr>
<td>TCP-SYN</td>
<td>0.91</td>
<td>0.92</td>
<td>0.92</td>
</tr>
</tbody>
</table>

TABLE II: Multi-class Classification Performance using ML Algorithms

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>$P_{\mu}$</th>
<th>$R_{\mu}$</th>
<th>$F_{\mu}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVM</td>
<td>0.89</td>
<td>0.79</td>
<td>0.81</td>
</tr>
<tr>
<td>MLP</td>
<td>0.59</td>
<td>0.54</td>
<td>0.54</td>
</tr>
<tr>
<td>RF</td>
<td>0.96</td>
<td>0.92</td>
<td>0.94</td>
</tr>
<tr>
<td>BC</td>
<td>0.95</td>
<td>0.93</td>
<td>0.94</td>
</tr>
<tr>
<td>KNC</td>
<td>0.79</td>
<td>0.75</td>
<td>0.77</td>
</tr>
<tr>
<td>ABC</td>
<td>0.69</td>
<td>0.59</td>
<td>0.55</td>
</tr>
</tbody>
</table>

attacks that can occur in any networking environment and can be launched on both network devices and/or end hosts.

IV. EXPERIMENTATION, RESULTS, AND ANALYSIS

To showcase the functionality of UNR-IDD, we run evaluations using the dataset and demonstrate the performance achieved. We illustrate results across multiple scenarios by varying the classification type, the ML algorithms, and other prominent NIDS datasets. For performance evaluation, we are using accuracy ($A$), precision ($P$), Recall ($R$), and F-Measure ($F$), mean precision ($\mu P$), mean recall ($\mu R$), and mean f-measure ($\mu F$) across all label types.

First, we observe the performance that is being achieved from UNR-IDD on multi-class classification as seen in Table I. We are utilizing a Random Forest (RF) as our ML algorithm for this. We can see that the RF achieves excellent performance as all the label types attain high $P$, $R$, and $F$ scores. These can be attributed to the fact that each label type has adequate representation and enough data samples, making them linearly separable from each other, demonstrating one of the contributions of this proposed work. This makes it easier for ML classifiers to recognize them individually, which does not deteriorate performance.

Next, we observe the performance that is being achieved from the proposed dataset using multiple ML algorithms: RF, Multi-layer Perceptron (MLP), Support Vector Machine (SVM), Bagging Classifier (BC), KNeighborsClassifier (KNC), and AdaBoost Classifier (ABC) in Table II. We can see that the best performance is achieved by the RF and BC classifiers as they achieve near-optimal $P_{\mu}$, $R_{\mu}$, and $F_{\mu}$ scores. This is followed by the SVM, KNC, and ABC classifiers which achieves above-average scores, succeeded by the MLP which achieves substandard performance. These results can be associated with the fact that RF and BC classifiers are ensemble classifiers consisting of multiple decision trees and can overcome the problem of overfitting. Accuracy and variable importance are also automatically generated in RF and BC [14], compared to the other classifiers observed.

We also analyze the explainability of the RF model across various labels. This is conducted by analyzing the predictions of the model, on random testing samples, using the Local Interpretable Model-agnostic Explanations (LIME) framework [15]. These results are provided in Figure 2, and showcase that our utilized port and delta port statistics are very influential in discerning between intrusion labels. This emphasizes another proposed contribution of our UNR-IDD dataset.

Lastly, we compare the performance that is being achieved from the proposed UNR-IDD dataset to two open-sourced NIDS datasets: NSL-KDD and CIC-IDS-2018. We use the same RF classifier for all three datasets and their performance is evaluated using $A$, $P_{\mu}$, $R_{\mu}$, and $F_{\mu}$. We also introduce a new metric, min $F$, which represents the minimum F-Measure score that is achieved for any label in that dataset. This metric can highlight the variability between $F_{\mu}$ and the min $F$ value in each dataset.

We observe the impact of the dataset sizes on the training times. We provide this comparison in Table III where we provide the dataset dimensions (samples and features) for all the datasets. The proposed UNR-IDD dataset has the lowest observed operational footprint. For evaluation, we note both the Overall Training Time (OTT) in seconds (s) and Normalized Training Time (NTT) in milliseconds (ms). We observe that UNR-IDD, due to its smaller dimensions, takes less time to train than NSL-KDD and much less time to train than CIC-IDS-2018. NTT can be defined as the time taken to train one sample and can be computed by dividing the OTT by the number of samples. We notice that the NTT is lowest for the NSL-KDD, with UNR-IDD achieving comparable performance to it. This can be attributed to NSL-KDD having only 3 categorical features per sample, whereas UNR-IDD...
UNR-IDD CIC-IDS-2018 NSL-KDD NIDS Datasets
0.55
0.60
0.65
0.70
0.75
0.80
0.85
0.90
0.95
1.00Performance
Accuracy
Mean Precision
Mean Recall
Mean F-Measure
Minimum F-Measure

Intrusion Detection Dataset (UNR-IDD) for network intrusion detection. The dataset addresses several limitations of existing datasets by primarily using network port statistics and delta port statistics, to achieve fine-grained analysis of the network and rapid identification of potential intrusions. Emphasis is also placed on improving representation for all classes so that each class achieves high performance uniquely. Results show that UNR-IDD helps ML models to attain better performance when classifying intrusions compared to existing datasets. Future work in this research can include augmenting the dataset with more intrusion categories. Presently, this dataset can be publicly accessed on Kaggle [16].

V. CONCLUSION AND FUTURE WORK

In this paper, we propose the University of Nevada - Reno Intrusion Detection Dataset (UNR-IDD) for network intrusion detection. The dataset contains 5 categorical features per sample. CIC-IDS takes the most NTT out of the three datasets. From the perceived OTT and NTT, we observe that the UNR-IDD provides the quickest OTT and very comparable NTT. This signifies that using UNR-IDD, a competent ML model for intrusion detection can be generated much quicker as it can train the overall dataset the fastest while training each sample with a comparable speed to that of the other observed NIDS datasets.

In Figure 3, we observe that both the NSL-KDD and CIC-IDS-2018 datasets achieve 99% A scores. Relatively, the UNR-IDD dataset achieves comparable performance with an A score of 95%. This can be attributed to the UNR-IDD dataset being smaller, overall, than NSL-KDD and significantly smaller, overall, than CIC-IDS-2018 in terms of both the number of samples and features. We also note that the $P_\mu$ score for the UNR-IDD dataset is equivalent to that of CIC-IDS-2018 at 96%. Compared to this, NSL-KDD achieves a $P_\mu$ score of 79%. Similarly, the $R_\mu$ score for UNR-IDD is higher than CIC-IDS-2018 and NSL-KDD at 93% versus 91% and 74%, respectively. The most important contribution of the proposed UNR-IDD dataset is its effect on F-Measure scores. Since each tail class is adequately represented in the dataset, it achieves the highest $F_\mu$ out of all three datasets with 94% compared to 93% and 76% for CIC-IDS-2018 and NSL-KDD, respectively. Similarly, the minimum $F$ score that is achieved across all three datasets is highest in the UNR-IDD dataset with 86%, while the CIC-IDS-2018 and NSL-KDD datasets achieve a minimum $F$ score of 58% and 0% respectively. This highlights the UNR-IDD’s prioritization of the F-Measure score as it achieves the least variability between the $F_\mu$ and the min $F$ value observed among all the datasets.

The proposed UNR-IDD dataset can perform network intrusion detection with competent performance. The dataset prioritizes representation for all tail classes and ensures that each label achieves high performance and F scores. Compared to customary datasets, UNR-IDD is a smaller overall dataset. However, the dataset still provides efficient performance across all the labels. Due to this, anomaly/intrusion detection could be trained more easily in resource-constrained network devices or low-end servers.

ACKNOWLEDGEMENT

This research is supported by the National Science Foundation (NSF) Award #2019164.

REFERENCES