TechRxiv
Powered by IEEE

UNR-IDD: Intrusion Detection Dataset using Network Port Statistics

This paper was downloaded from TechRxiv (https://www.techrxiv.org).

LICENSE

CC BY 4.0

SUBMISSION DATE / POSTED DATE

25-05-2022 / 17-06-2022

CITATION

Das, Tapadhir; Abu Hamdan, Osama; Shukla, Raj; Sengupta, Shamik; Arslan, Engin (2022): UNR-IDD: Intrusion Detection Dataset using Network Port Statistics. TechRxiv. Preprint. https://doi.org/10.36227/techrxiv.19877311.v2

DOI

10.36227/techrxiv.19877311.v2

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*—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.**

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

I. Introduction

Modern computer networks and their connected applications have reached unprecedented growth with implementations like the internet of things, smart homes, and software-defined networks. However, this has also increased the potential for network intrusions, which are a continuous threat to network infrastructures as they attempt to compromise the major principles of computing systems: availability, authority, confidentiality, and integrity [1]. These threats are difficult to detect unaided, as they display indistinguishable network traffic patterns as normal functionality [2]. Approaches like firewalls cannot detect these intrusions as they do not possess the ability to inspect and conduct deep packet inspection. This has led to the rise in Network Intrusion Detection Systems (NIDS) which aim to provide higher protection [3]. NIDS constantly monitor the network traffic and its patterns for any kind of malicious trend or behavior. Modern networking applications have become more diverse, dynamic, interconnected, and geographically distributed, with a larger number of users, applications, and nodes. This has resulted in the generation of substantial amounts of shared data. Protecting these large-scale networks has become a challenge as existing NIDS show inefficient intrusion detection performance, including high false alarm rates and zero-day attacks [4].

To provide enhanced protection against intrusions, the usage of machine learning (ML) for NIDS has gained traction in the last decade as various open-sourced datasets have been proposed and established by multiple research groups globally. Most used NIDS datasets include NSL-KDD [5], UNSW-NB15 [3], and CIC-IDS-2018 [6]. However, a common problem that has been identified with many of these datasets is inadequate modeling of tail classes [7]. Tail classes refer to classes or labels in the datasets that have limited samples compared to other classes, which leads to poor performance when fitting the ML model. Researchers have been looking at methods to address the issue of tail classes in datasets. Common methods that have been investigated include under-sampling and oversampling. However, oversampling increases the size of the dataset, thereby increasing training time, memory requirements, and time complexity. Correspondingly, undersampling can reduce data samples from the majority classes, which can affect the overall performance of prediction models [8]. It can also be argued that because these techniques manipulate already existing data samples, they do not add any additional value or new insights.

Other methods that have been investigated to address the insufficient representation of tail classes include Information Augmentation like transfer learning and data augmentation, and Module Improvements like representation learning and ensemble methods [7]. These techniques, however, have limitations in their own regard. For example, class-agnostic information augmentation techniques have the potential of further increasing class variability as head classes have more samples and would be augmented more. Module Improvement methods like representation learning and ensemble learning have their shortcomings as well. One critique of representation learning is that the accumulated training stages make decoupled training less practical to be integrated with existing well-formulated methods. Ensemble methods can lead to higher computational costs due to the presence of multiple classifiers, which can be a hindrance to performance. One valid method to ensure that tail classes have adequate representation is to prioritize them during data generation so that all classes can achieve adequate modeling and high performance. This necessitates the need for a new NIDS dataset in which all labels are adequately represented. Another limitation of the current datasets is that they mostly depend on flow level statistics. This can limit the transferability of the NIDS solutions to other network configurations since the flow statistics depend on the network topology and traffic characteristics. In addition, many of these datasets contain redundant features that play no impact in being able to detect potential network intrusion types. Depending on the machine learning approach being utilized, these features can also increase the cardinality of the dataset, leading to increased training and inference times. Finally, some existing datasets suffer from incomplete or missing records. These records or samples must be ignored or dropped from the overall dataset, which leads to sub-optimal performance. Addressing the above-mentioned limitations is vital to ensuring that proper NIDS are being developed to adequately protect networking infrastructures from intrusions.

In this paper, we propose the University of Nevada - Reno Intrusion Detection Dataset (UNR-IDD), a NIDS dataset. The main difference between UNR-IDD and existing datasets is that UNR-IDD consists primarily of network port statistics. These refer to the observed port metrics recorded in switch/router ports within a networking environment. The dataset also includes delta port statistics which indicates the change in magnitude of observed port statistics within a time interval. Compared to datasets that primarily use flow level statistics, these port statistics can provide a fine-grained analysis of network flows from the port level as decisions are made at the port level versus the flow level. This can lead to rapid identification of potential intrusions. We also address the limitation of the presence of tail classes. Our dataset ensures that there are enough samples for ML classifiers to achieve high F-Measure scores, uniquely. Our proposed dataset also ensures that there are no missing network metrics and that all data samples are filled. The proposed dataset metrics can be obtained through most networking architectures. However, for our testbed, we have used a software-defined network (SDN) environment, due to the wide relevance of and usage of SDN architectures across industries, organizations, and critical infrastructures. In summary, the main contributions of our dataset include:

- The primary usage of port and delta port statistics to model the various intrusions in the dataset.
- Confirms enough samples to ensure high-performance metrics across all tail classes.
- Ensuring all data samples are filled and there is no missing data in the dataset.
- Provides performance comparison of intrusion detection models when using UNR-IDD and other NIDS datasets.

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

II. Related Work

Multiple NIDS datasets have emerged for research and development purposes. Some of the very first datasets that emerged for creating NIDS include the DARPA [9] and KDDCup99 [10]. The main issue with these datasets is that they are outdated. KDDCup99 suffers from redundant and duplicate data samples. DARPA dataset, on the other hand, does not represent modern networks as it was simulated in a military network environment. Thus, it contains artificially high feature magnitudes compared to real traffic data. Other proposed datasets for NIDS include CAIDA [11], CDX [12], Kyoto [13], Twente [14], and ISCX2012 [15]. CAIDA dataset suffers from having very limited features and was only constrained to being able to detect Distributed Denial of Service (DDoS) attacks. Similarly, the CDX dataset only contains five features and could only detect buffer overflow attacks. The Kyoto dataset is restricted to two classes and contains limited features, which restraints its functionality. The Twente and ISCX2012 datasets solely focused on IP flows within the network level, 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. The size of the Twente dataset is also small to perform adequate modeling of the intrusions [16].

As mentioned above, the three prominent NIDS datasets are NSL-KDD [5], CIC-IDS-2018 [6], and UNSW-NB15 [3]. These datasets also have several limitations. For instance, the UNSW-NB15 suffers from inconsistent performance for machine learning classifiers. It requires more rigorous and expanded machine learning mechanisms which increases training and inference times. The NSL-KDD and CIC-IDS-2018 datasets suffer from missing data samples within their datasets. Many of these datasets also suffer from the issue of containing inadequately modeled tail classes which leads to inconsistent performance.

For more effective intrusion detection, we need to ensure that a dataset contains a wide variety of intrusion categories. We also need to make sure that it is complete as missing data can negatively impact the performance of prediction models. The primary usage of port level statistics, in conjunction with some flow statistics, for NIDS is another attractive research direction that can be employed to check their efficacy at detecting network intrusions. Lastly, it is critical to ensure that tail classes have adequate representation so that prediction models can accurately capture their unique behavior and attain high performance.

III. UNR-IDD Dataset

We setup up our testbed using an SDN simulation environment 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 the testbed configuration, 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

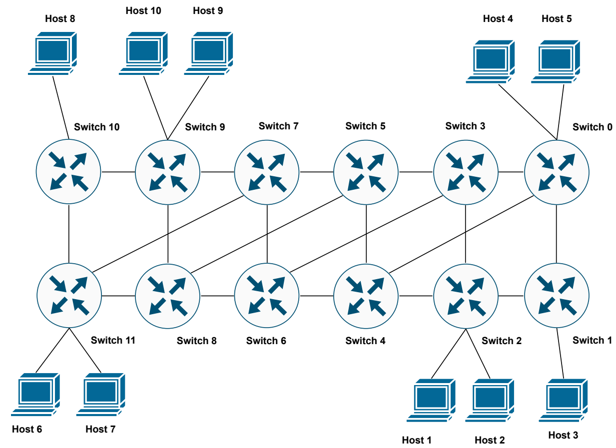


Fig. 1: Simulated SDN topology

at runtime for the creation and activation of components and for auto-wiring components together, which makes 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 (component) 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 is illustrated in Figure 1. The topology 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 a duration of 5 seconds. We then simulate flows under normal and intrusion conditions to gather data in every possible scenario. To ensure that each normal and anomaly 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 in every 5 seconds) from OpenFlow (OF) switches. The statistics are collected through by 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

TABLE I: Port statistics collected for every port on every switch

Port Statistic	Description
Received Packets	Number of packets received by the port
Received Bytes	Number of bytes received by the port
Sent Packets	Number of packets sent by the port
Sent Bytes	Number of bytes sent
Port alive Duration	The time port has been alive in seconds
Packets Rx Dropped	Number of packets dropped by the receiver
Packets Tx Dropped	Number of packets dropped by the sender
Packets Rx Errors	Number of transmit errors
Packets Tx Errors	Number of receive errors

TABLE II: Delta Port statistics collected for every port on every switch every 5 seconds

Delta Port Statistic	Description
Delta Received Packets	Number of packets received by the port
Delta Received Bytes	Number of bytes received by the port
Delta Sent Packets	Number of packets sent by the port
Delta Sent Bytes	Number of bytes sent
Delta Port alive Duration	The time port has been alive in seconds
Delta Packets Rx Dropped	Number of packets dropped by the receiver
Delta Packets Tx Dropped	Number of packets dropped by the sender
Delta Packets Rx Errors	Number of transmit errors
Delta Packets Tx Errors	Number of receive errors

collected statistics to a Javascript Object Notation (.json) file with a name $N_i.json$. Table I shows the collected port statistics and their descriptions per port on every switch in the simulated SDN. These statistics relay the collected metrics and magnitudes from every single port within the SDN, when a flow is simulated between two hosts. Table II illustrates the collected delta port statistics and their descriptions per port on every switch. 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. The time interval for these observed metrics is configured as 5 seconds, which can provide greater detail in detecting intrusions. Additionally, we also collect some flow entry and flow table statistics to work in conjunction with the collected port statistics as seen in Table III. These metrics provide information about the conditions of switches in the network and can be collected in any network setting.

D. Intrusions

The following intrusions are simulated in the data collection phase:

- **TCP-SYN Flood**: A Distributed Denial of Service (DDoS) attack where attackers target hosts by initiating many Transmission Control Protocol (TCP) handshake processes without waiting for the response from the target node. By doing so, the target device's resources are consumed as it has to keep allocating some memory space for every new TCP request.
- **Port scan**: An attack in which attackers scan available ports on a host device in order to learn information about the services, versions, and even security mechanisms that are running on that host.
- **Flow Table Overflow**: An attack that targets network switches/routers where attacks compromise the functionality of a switch/router by consuming the flow tables that

TABLE III: Flow statistics collected

Statistic	Description
Connection Point	Network connection point expressed as a pair of the network element identifier and port number
Total Load/Rate	Obtain the current observed total load/rate (in bytes/s) on a link
Total Load/Latest	Obtain the latest total load bytes counter viewed on that link
Unknown Load/Rate	Obtain the current observed unknown-sized load/rate (in bytes/s) on a link
Unknown Load/Latest	Obtain the latest unknown-sized load bytes counter viewed on that link
Time seen	When the above-mentioned values were last seen
is_valid	Indicates whether this load was built on valid values
TableID	Returns the Table ID values
ActiveFlowEntries	Returns the number of active flow entries in this table.
PacketsLookedUp	Returns the number of packets looked up in the table.
PacketsMatched	Returns the number of packets that successfully matched in the table
MaxSize	Returns the maximum size of this table.

TABLE IV: Multi-class Classification Labels

Label	Description
Normal	Normal Network Functionality
Attack	Network Intrusion

forward packets with illegitimate flow entries and rules so that legitimate flow entries and rules cannot be installed.

- **Blackhole**: An attack that targets network switches/routers to discard the packets that pass through, instead of relaying them on to the next hop.
- **Traffic Diversion**: A attack that targets network switches/routers to reroute the direction of packets away from their destination, with the goal of increasing travel time and/or to spy on network traffic through a man-in-the-middle scenario.

These intrusion types were selected for this dataset as they are common cyber attacks that can occur in any networking environment. Also, these intrusion types cover attacks that can be launched on both network devices and end hosts.

E. Labels

This dataset can be broken down into two different machine learning classification problems: binary and multi-class classification. The goal of binary classification is to differentiate intrusions from normal working conditions. Binary classification can estimate if a network is under attack but does not provide any information about the type of attack. The labels for binary classification in UNR-IDD are illustrated in Table IV.

The goal for multi-class classification, however, 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. The labels for multi-class classification in UNR-IDD are illustrated in Table V.

TABLE V: Multi-class Classification Labels

Label	Description
Normal	Normal Network Functionality
TCP-SYN	TCP-SYN Flood
PortScan	Port Scanning
Overflow	Flow Table Overflow
Blackhole	Blackhole Attack
Diversion	Traffic Diversion Attack

TABLE VI: Binary Classification Performance

Label	*P*	*R*	*F*
Attack	1.0	1.0	1.0
Normal	1.0	1.0	1.0

IV. Experimentation, Results, and Analysis

To showcase the functionality of our proposed dataset, 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 in the literature. For performance evaluation, we are using accuracy (*A*), precision (*P*), Recall (*R*), and F-Measure (F) scores as the metrics. In addition, we are also using the mean scores for precision (P_μ), recall (R_μ), and f-measure(F_μ) across all label types in the datasets during multi-class classification. This will provide us with the mean performance achieved on the dataset for all label types.

First, we observe the performance that is being achieved from the proposed dataset on both binary classification and multi-class classification scenarios. For this, we are utilizing a Random Forest (RF) as our ML algorithm. We chose to use an RF due to its relevance and wide usage when studying NIDS in literature. The binary classification performance and multi-class classification performance achieved from the dataset can be observed in Table VI and Table VII, respectively. We can see that in Table VI, both label types are providing a performance of 1.0 for *P*, *R*, and *F* scores. This means that the dataset is linearly separable, and the RF has no trouble detecting if a given network flow is normally functioning or under any potential intrusion. Similarly, in Table VII, we see the RF achieving excellent performance as well. All the label types achieve high scores for *P*, *R*, and *F* scores. These can be attributed to the fact that each label type has adequate representation and enough data samples thereby, makes them linearly separable from each other. This makes it easier for machine learning classifiers to recognize them individually, which does not deteriorate performance. These results demonstrate one of the main contributions of this proposed dataset, which is ensuring that all labels have enough data samples to achieve high performances, individually and as a collective.

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). We chose to use these algorithms due to their relevance and wide usage when studying NIDS in literature, along with their ease of accessibility

TABLE VII: Multi-class Classification Performance

Label	P	R	F
Blackhole	0.98	0.98	0.98
Diversion	0.99	0.97	0.98
Normal	1.0	1.0	1.0
Overflow	0.98	0.76	0.86
PortScan	0.91	0.94	0.92
TCP-SYN	0.91	0.92	0.92

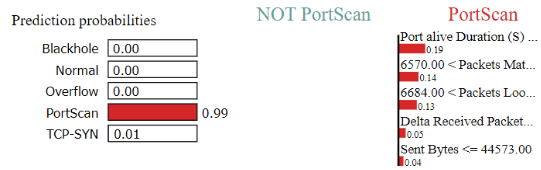
TABLE VIII: Multi-class Classification Performance using Machine Learning Algorithms

Algorithm	P_μ	R_μ	F_μ
SVM	0.89	0.79	0.81
MLP	0.59	0.54	0.54
RF	0.96	0.92	0.94
BC	0.95	0.93	0.94
KNC	0.79	0.75	0.77
ABC	0.69	0.59	0.55

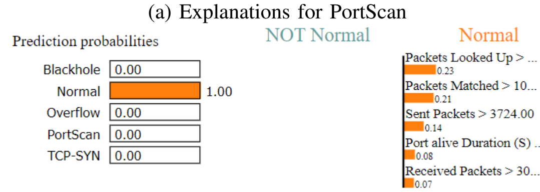
through the sklearn libraries. The performance achieved by the algorithms on UNR-IDD is provided in Table VIII. We can see that the best performance is achieved by the RF and BC classifiers as they achieve the near-optimal P_μ, R_μ, and F_μ scores. This is followed by the SVM, KNC, and ABC classifiers which achieves above-average scores, succeeded by the MLP which achieves substandard performance. These results can be associated with the fact that an RF classifier is an ensemble classifier consisting of multiple decision trees and can overcome the problem of overfitting. Similar functionality occurs in BC as it also is an ensemble classifier whose default classifier is a decision tree. Accuracy and variable importance are also automatically generated in RF and BC [17], compared to the other classifiers observed.

We also analyze the explainability of the RF model across various labels for a more comprehensive analysis. This is conducted by analyzing the predictions of the model, on testing samples with varying predicted labels, using the Local Interpretable Model-agnostic Explanations (LIME) framework [18]. These results are provided in Figure 2. We observe the top 5 influential features for the labels *PortScan*, *Normal*, and *Overflow*. Through this analysis, we notice that the most influential feature for *PortScan* intrusion is *Port alive Duration*, for *Normal* traffic it is *Packets Looked Up*, and for *Overflow* intrusion is Delta Sent Packets. We also observe that most of the influential features that are captured using the LIME framework are primarily port or delta port statistics. This demonstrates another proposed contribution to the UNR-IDD dataset, which incorporates the primary usage of port and delta port statistics to model the various intrusions in the dataset.

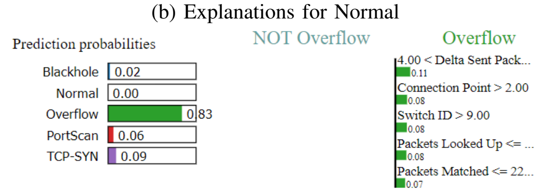
Lastly, we compare the performance that is being achieved from the proposed UNR-IDD dataset to two open-sourced NIDS datasets: NSL-KDD and CIC-IDS-2018. These two are established NIDS datasets that are frequently used for researching network intrusion and anomaly detection. We use the same RF classifier for all three datasets and their performance is evaluated using A, P_μ, R_μ, F_μ. We also introduce a new metric, min F, which represents the minimum F-Measure score that is achieved for any label in that dataset.



(a) Explanations for PortScan



(b) Explanations for Normal



(c) Explanations for Overflow

Fig. 2: LIME Explanations for UNR-IDD

This metric can highlight the variability between F_μ and the min F value in each dataset. The results of this analysis are illustrated in Figure 3. The training times for each of the datasets are also provided in Figure 4.

From 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 than NSL-KDD and significantly smaller than CIC-IDS-2018. However, this size variability is advantageous for the UNR-IDD dataset as it takes significantly less time to train in comparison to both other datasets while achieving comparable performance. In Figure 4, we note that UNR-IDD takes 4.28 seconds to train, while NSL-KDD and CIC-IDS-2018 take 9 and 9056 seconds to train, respectively. We also note that the P_μ score for the UNR-IDD dataset is equivalent to that of CIC-IDS-2018 at 96%. Compared to this, NSL-KDD achieves a P_μ score of 79%. Similarly, the R_μ 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_μ 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 prioritization towards the F-Measure score as it achieves the least variability between the F_μ and the min F value observed among all the datasets.

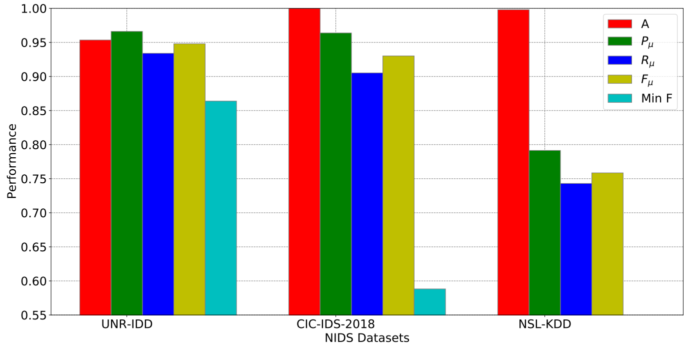


Fig. 3: Performance Analysis of UNR-IDD, NSL-KDD, and CIC-IDS-2018 datasets

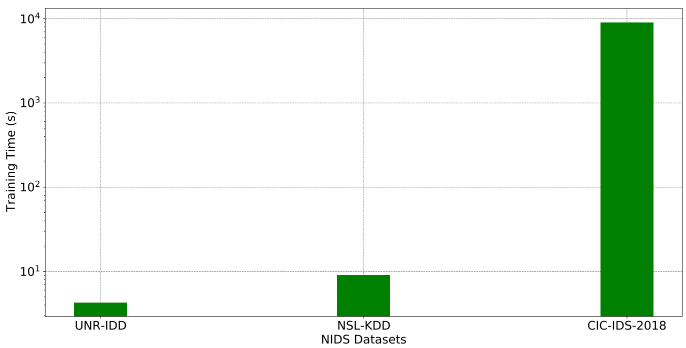


Fig. 4: Training Time of UNR-IDD, NSL-KDD, and CIC-IDS-2018 datasets

The proposed UNR-IDD dataset can perform network intrusion detection with efficient performance. This dataset takes the least amount of time to train and achieves better performance at conducting network intrusion detection than some of the other customary and established NIDS datasets currently being used in literature. The dataset prioritizes representation for all tail classes and ensures that each label achieves near-optimal performance and F scores.

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 addresses several limitations of existing datasets by primarily using network port statistics and delta port statistics, to achieve a more fine-grained analysis of the network and rapid identification of potential intrusions within a network. Emphasis is also placed on improving representation for all tail classes, so that each class has adequate representation and achieves high performance individually, in conjunction with the whole dataset. Also, to ensure optimal performance, there are no incomplete or missing data samples within the dataset. The dataset is generated using an SDN simulation environment and desired network statistics are collected through OpenFlow metrics. Simulation and results show that the dataset is able to achieve efficient performance for both binary and multi-class classification scenarios and achieves the best performance when using an RF classifier. Results also show that the dataset achieves better performance than other traditional and established datasets, with a priority for improved and high scoring F_μ and min F scores. This highlights the dataset's prioritization of the adequate label representation, compared to other datasets.

References

[1] R. Heady, G. Luger, A. Maccabe, and M. Servilla, "The architecture of a network level intrusion detection system," Los Alamos National Lab.(LANL), Los Alamos, NM (United States); New Mexico ..., Tech. Rep., 1990.

[2] T. Das, R. M. Shukla, and S. Sengupta, "The devil is in the details: Confident & explainable anomaly detector for software-defined networks," in *2021 IEEE 20th International Symposium on Network Computing and Applications (NCA)*. IEEE, 2021, pp. 1–5.

[3] N. Moustafa and J. Slay, "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)*. IEEE, 2015, pp. 1–6.

[4] Z. Ahmad, A. Shahid Khan, C. Wai Shiang, J. Abdullah, and F. Ahmad, "Network intrusion detection system: A systematic study of machine learning and deep learning approaches," *Transactions on Emerging Telecommunications Technologies*, vol. 32, no. 1, p. e4150, 2021.

[5] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the kdd cup 99 data set," in *2009 IEEE symposium on computational intelligence for security and defense applications*. IEEE, 2009, pp. 1–6.

[6] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization." *ICISSp*, vol. 1, pp. 108–116, 2018.

[7] Y. Zhang, B. Kang, B. Hooi, S. Yan, and J. Feng, "Deep long-tailed learning: A survey," *arXiv preprint arXiv:2110.04596*, 2021.

[8] A. Y.-c. Liu, "The effect of oversampling and undersampling on classifying imbalanced text datasets," Ph.D. dissertation, Citeseer, 2004.

[9] R. Lippmann, J. W. Haines, D. J. Fried, J. Korba, and K. Das, "The 1999 darpa off-line intrusion detection evaluation," *Computer networks*, vol. 34, no. 4, pp. 579–595, 2000.

[10] "KDD Cup 1999 Data." [Online]. Available: http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html

[11] "Downloads of CAIDA Online Datasets," Jun. 2013. [Online]. Available: https://www.caida.org/catalog/datasets/about/downloads/tables/

[12] B. Sangster, T. O'Connor, T. Cook, R. Fanelli, E. Dean, C. Morrell, and G. J. Conti, "Toward instrumenting network warfare competitions to generate labeled datasets." in *CSET*, 2009.

[13] J. Song, H. Takakura, and Y. Okabe, "Description of kyoto university benchmark data," *Available at link: http://www. takakura. com/Kyoto_data/BenchmarkData-Description-v5. pdf [Accessed on 15 March 2016]*, 2006.

[14] A. Sperotto, R. Sadre, F. v. Vliet, and A. Pras, "A labeled data set for flow-based intrusion detection," in *International Workshop on IP Operations and Management*. Springer, 2009, pp. 39–50.

[15] A. Shiravi, H. Shiravi, M. Tavallaee, and A. A. Ghorbani, "Toward developing a systematic approach to generate benchmark datasets for intrusion detection," *computers & security*, vol. 31, no. 3, pp. 357–374, 2012.

[16] A. Thakkar and R. Lohiya, "A review of the advancement in intrusion detection datasets," *Procedia Computer Science*, vol. 167, pp. 636–645, 2020.

[17] N. Farnaaz and M. Jabbar, "Random forest modeling for network intrusion detection system," *Procedia Computer Science*, vol. 89, pp. 213–217, 2016.

[18] M. T. Ribeiro, S. Singh, and C. Guestrin, "" why should i trust you?" explaining the predictions of any classifier," in *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, 2016, pp. 1135–1144.