# TOWARDS A SITUATIONAL AWARENESS-ORIENTED NETWORK INTRUSION DETECTION DATASET

by

STYLIANOS KLADOS

Bachelor's degree, Informatics Engineering, Technological Educational Institute of Crete, 2020

A THESIS

submitted in partial fulfillment of the requirements for the degree

MASTER OF SCIENCE

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

SCHOOL OF ENGINEERING

HELLENIC MEDITERRANEAN UNIVERSITY

2022

Approved by:

Evangelos Markakis

# Abstract

In recent years, cyber-attacks have rapidly increased and become more and more sophisticated. As a result, the detection of malicious activity in cyberspace has become a complex task. Machine learning (ML)-based Network Intrusion Detection Systems (NIDS) are one of the most promising areas of study that can aid in discovering and assessing malicious activities. Such systems utilize specially created datasets to train their ML algorithms and eventually predict if an attack is occurring or not. Most ML-based NIDSs are trained on network-only data, therefore system or firewall logs, which may give critical cybersecurity incident information, are seldom used. To the best of our knowledge, a dataset that fuses such heterogeneous data for ML-based NIDS applications is not yet proposed. This thesis proposes a heterogeneous dataset that consists of three different types of data, namely: network traffic data, information stemming from a vulnerability assessment tool, and system logs. The network part of the dataset comprises NetFlow network protocol data [1]. Regarding the vulnerability part, an extra feature has been added to the dataset indicating the existence or not of any system vulnerability. Moreover, the system logs that have been added to the dataset were interpreted into numerical scores, through sentiment analysis by using Natural Language Processing (NLP). The aforementioned different types of data were time-correlated and fused into a single heterogeneous dataset. The proposed dataset was used to train six (6) ML algorithms, and the prediction results from this procedure were used to evaluate it as a training dataset. KDD'99 and CCD-IDSv1 were also utilized to train the same ML algorithms, and their prediction results were compared to those of the proposed dataset. Finally, the findings showed that the ML models trained with the suggested heterogeneous dataset had higher post-training accuracy predictions.

# Table of Contents

List of Figures ..... v
List of Tables ..... vi
Chapter 1 - Introduction ..... 1
Chapter 2 - State of the Art ..... 3
Chapter 3 - Implementation ..... 7
Network Topology ..... 7
Data Collection Process ..... 9
Data Pre-processing ..... 10
Network Data ..... 11
System and Firewall Logs ..... 12
Vulnerability Information ..... 14
Chapter 4 - Evaluation ..... 17
Aim of the experiment ..... 17
Method ..... 17
Variables ..... 19
Dependent Variables ..... 19
Independent Variables ..... 20
Fixed Variables ..... 20
Prediction ..... 20
Results ..... 21
Binary Classification – Training Results ..... 21
Binary Classification – Post-Training Evaluation Results ..... 23
Multiclass Classification – Training Results ..... 26
Multiclass Classification – Post-Training Evaluation Results ..... 27
Evaluation Summary ..... 29
Chapter 5 - Conclusion ..... 30
Future Work ..... 30
Bibliography ..... 32
Appendix A - Confusion Matrices ..... 37

Binary Classification Predictions – KDD'99 ..... 37
Binary Classification Prediction – CCD-IDSv1 ..... 38
Binary Classification Predictions – Heterogeneous Dataset ..... 39
Multiclass Classification Predictions – KDD'99 ..... 40
Multiclass Classification Predictions – CCD-IDSv1 ..... 42
Multiclass Classification Predictions – Heterogeneous Dataset ..... 43
Appendix B - Netflow v5 Initial Feature Set ..... 45
Appendix C - Technology Enablers ..... 49

# List of Figures

Figure 1. Network Topology ..... 9
Figure 2. Network topology and systems' setup process ..... 9
Figure 3 : Data collection process flow diagram ..... 10
Figure 4. Dataset pre-processing flow diagram ..... 10
Figure 5. Dataset evaluation process flow diagram ..... 17
Figure 6. Binary Classification - Training Results (MLP, CNN, CNN-LSTM) ..... 22
Figure 7. Binary Classification - Training Results (DNN, GRU, LSTM) ..... 23
Figure 8. Binary Classification - Post-Training Evaluation Results (MLP, CNN, CNN-LSTM) 24
Figure 9. Binary Classification - Post-Training Evaluation Results (DNN, GRU, LSTM) ..... 25
Figure 10. Multiclass Classification - Training Results (MLP, CNN, CNN-LSTM) ..... 26
Figure 11. Multiclass Classification - Training Results (DNN, GRU, LSTM) ..... 27
Figure 12. Multiclass Classification - Post-Training Evaluation Results (MLP, CNN, CNN-LSTM) ..... 28
Figure 13. Multiclass Classification - Post-Training Evaluation Results (DNN, GRU, LSTM) .. 29
Figure B 14. Initial Netflow v5 Feature Set (cont. in Figure 15) ..... 45
Figure 15 Initial Netflow v5 Feature Set (cont. in Figure 16) ..... 46
Figure 16 Initial Netflow v5 Feature Set (cont. in Figure 17) ..... 47
Figure 17 Initial Netflow v5 Feature Set ..... 48

# List of Tables

Table 1. Cyberattacks and their corresponding benign systems ..... 8
Table 2. Dataset labels and their corresponding cyberattacks ..... 11
Table 3. Removed Netflow features and their description (cont. in Table 4) ..... 11
Table 4. Removed Netflow features and their description ..... 12
Table 6. System and firewall logs and their sentiment analysis scores ..... 14
Table 6. Features and their feature importance scores (cont. in Table 7) ..... 15
Table 7. Features and their feature importance scores ..... 16
Table 8. Average machine learning scores for binary classification ..... 21
Table 9. Average machine learning scores for multiclass classification ..... 21

# Chapter 1 - Introduction

The rapid progress of technology in recent years has resulted in a significant increase in the amount of sensitive and important information that is daily exchanged over the internet. In the meantime, cybersecurity attacks such as ransomware and data breaches that target such data have become very common. According to research conducted by the European Union's Agency for Cybersecurity [2], ransomware attacks were estimated to cause €18 billion worth of damage in 2021. Therefore, securing every network entity and its data is a critical task.

For that reason, network security has been a continuously emerging research field in recent years [3]. One of the most promising areas of research that can help in identifying and assess the malicious activity are Intrusion Detection Systems (IDS). IDSs [4] are systems that monitor an entire network, or individual network entities, for cybersecurity incidents. During such incidents, an alert is triggered by the IDS, and the person in charge is notified. IDS systems are split into two (2) categories: Network Intrusion Detection Systems (NIDS), which monitor network traffic for malicious activity; and Host Intrusion Detection Systems (HIDS), which mostly monitor systems' files and folders for unauthorized activity. An IDS, network or host-based, can infer if a malicious activity is occurring based on a set of rules. Those rules are created to monitor specific characteristics of a network or a network entity. Nonetheless, IDSs that work with a finite set of rules have inconsistent performance and can often present an increased number of false alarms [5]. This has led to further research to produce more efficient and accurate IDSs.

Machine learning (ML) is the current trend for analyzing patterns in data that contain a large amount of complex information [6]. As a result, the research community investigated the possibility of incorporating ML techniques to NIDSs to increase the prediction accuracy, regarding malicious activity. Existing or novel ML algorithms are trained with a set of data that portrays specific network behavior, such as malicious or benign, and produce an ML model that can eventually be used as a NIDS. ML systems' performance is solely dependent on the training data. Accordingly, the selection and the pre-processing of a dataset is vital for producing a well-trained ML model [7]. To this day, several network datasets have been created, specifically for NIDS-related use [8]. Ultimately, well-trained ML-based NIDSs can achieve high accuracy in malicious activity detection [9].

NIDSs frequently operate with a single data source, such as network traffic. However, combining numerous data sources would improve NIDS's performance [10]. The process of combining multiple sources of data to describe benign or malicious behavior is referred to as Cyber Situational Awareness (CSA). CSA can be represented more distinctly by the following principles [11]: the perception of the entities in a specific environment; the comprehension of their purpose; and the projection of their condition in the foreseeable future. From a technical perspective, those principles are interpreted as the compilation, processing, and fusion of heterogeneous data to infer if a cybersecurity incident is occurring or not. CSA paradigm is often used in cybersecurity applications, and more specifically in NIDSs, to enhance their performance [12].

In this thesis, a CSA-oriented NIDS heterogeneous dataset is proposed. This dataset consists of three different types of data, namely: network traffic data; information originating from a vulnerability assessment; and system and firewall logs. Ultimately, this dataset enables an ML-based NIDS to utilize 3 types of data at the same time for predicting a cybersecurity-related incident. The proposed dataset will be referred to as "the heterogeneous dataset". The heterogeneous dataset and two (2) more comparison datasets were used as training datasets for six (6) ML algorithms in the evaluation phase. The results showed that the ML algorithms that were trained with the heterogeneous dataset have high accuracy performance for both binary and multiclass classification (**98**% and **90**% respectively), which is approximately 30% higher than the comparison datasets' accuracy metrics.

The rest of this thesis is organized as follows: Chapter 2 showcases the State of the Art; in Chapter 3 the implementation of the dataset is showcased; Chapter 4 includes the evaluation process of the heterogeneous dataset; and Chapter 5 contains the conclusion and the future work statements.

# Chapter 2 - State of the Art

Detecting cyberattacks and malicious network behavior is becoming increasingly challenging due to ongoing cyberthreat advancements[1]. Machine learning is part of many contemporary software applications and enhances their ability to detect cyberattacks. The performance of a machine learning model is directly impacted by the data used to train it [13]. In the context of NIDS, the quality of data included in a dataset may influence cyber incident detection. In this section, benchmark cybersecurity datasets are presented.

The KDD'99 [14], which is one of the most popular NIDS datasets, was created in 1999 and contains both network traffic data and host-related information, such as the number of connections per host. It contains data derived from a simulated network that was purposely attacked. Since its creation, it has been used as a benchmark dataset for many NIDS research initiatives. Despite its popularity, KDD'99 has some drawbacks that led to further processing by Tavallaee et al. [15], resulting in the NSL-KDD dataset. While NSL-KDD is an improved version of KDD'99, it is not used as a benchmark as often as the latter.

KDD'99 served not only as a benchmark dataset, but also as a basis for the creation of new datasets, such as Kyoto 2006+ [16]. This dataset was created from real network traffic that was captured by honeypots, which function as virtual traps for cyber-attackers. The data was analyzed and annotated after the network traffic capture. Following this analytic process, three labels were derived, namely: normal, abnormal, and unknown. The dataset also includes three features that are not related to network traffic. Those features have a true or false value based on specific malicious situations, namely: IDS alert detection, malware detection, and exploit code detection. Kyoto 2006+ is one of the most popular research initiatives that has integrated heterogeneous data in a NIDS dataset.

In contrast to the previous methodology, where the network traffic was acquired from a real network, UNSW-NB15 [17] dataset's records originate from a network traffic generator tool[2]. This tool can emulate both benign and malicious network traffic. Argus[3], a network flow feature

[1] https://www.mcafee.com/enterprise/en-us/about/newsroom/press-releases/2021/20211109-01.html

[2] http://www.ixiacom.com/products/perfectstorm

[3] https://openargus.org/

extractor, Zeek[4], a cybersecurity log aggregator and intrusion detection software, and custom network analyzers were utilized to extract the final dataset's features from the emulated network traffic attributes.

A similar method was utilized to capture the benign traffic for the creation of the CIC-IDS-2017 dataset [18]. An agent was developed to emulate network traffic based on pre-created benign user profiles, which highlight actions that a real computer user might take. Malicious actor profiles were created respectively. Cyber-attacks and benign network traffic were captured for five consecutive days. Ultimately, the captured network traffic was converted to network flows, based on Netflow v5 with CICFlowMeter[5], resulting in 84 network-related features. To further advance their previous dataset, Sharafaldin et al. [18] created a newer version, namely, CSE-CIC-IDS-2018[6]. This enhanced version was enriched with the traffic produced by three new cyber-attacks. However, the features and the type of the dataset remain the same.

Following Sharafaldin's et al. example, Moustafa et al. [19] utilized ostinato[7], a software tool for network traffic generation, to produce benign network behavior. In addition, they created attack and benign user scenarios so that they could imitate real-life behavior. The features used for this dataset were derived from the Zeek cybersecurity log aggregator and intrusion detection tool. This dataset focuses on IoT devices and an IoT-related network topology.

A more realistic network traffic acquisition was performed by Damasevicious et al. [20], resulting in the LITNET-2020 dataset. The network capture process lasted for 10 months and involved network devices across 4 cities. After the network capture process, the authors analyzed the dataset to separate benign network traffic from malicious. After that, they further analyzed the data to classify the malicious traffic into a specific type of attack. The final dataset contains 85 network-related features, which consist of 49 features that originated from the NetFlow v9 protocol[8], 34 custom features that were added after some data processing, and 2 extra features for the labels.

---

[4] https://zeek.org/

[5] https://www.unb.ca/cic/research/applications.html

[6] https://registry.opendata.aws/cse-cic-ids2018/

[7] https://ostinato.org/

[8] https://datatracker.ietf.org/doc/html/rfc3954

Moreover, Thapa et al. [21] presented the CCD-IDSv1 dataset. The authors tried to create an as realistic a scenario as possible by creating scripts to emulate benign network activity, while malicious activities were generated by 5 different cyberattacks, carried out at random time intervals for 7 days. The network environment was emulated with virtual machines. Argus, the network flow feature extractor tool, was utilized to extract 25 network flow-based features.

Al-Daweri et al. [22] presented a novel methodology for creating a NIDS dataset. The authors' dataset comprises features stemming from the analysis of each attack's network traffic and other statistical attributes, such as the number of concurrent connections to the same host that could suggest that a scan attack is occurring [23], that specifically characterize each attack. Thus, the number of concurrent connections was chosen as a feature. However, those types of custom features mainly include 1 (true) or 0 (false) values. This dataset consists of four different network attacks.

Regarding the evaluation process that researchers employ to assess their proposed datasets, three distinct operations are identified:

1. Creating matrices that highlight and compare common characteristics between datasets, such as the total number of cyberattacks, or the total number of features.
2. Showcasing the results of the training of machine learning models (e.g., training accuracy, etc.).
3. Calculating the importance of the dataset's features (otherwise called feature importance).

"Feature importance" is defined as the technique of calculating a score that expresses the amount of the effect a feature has on the prediction of a certain label [24]. This procedure is used to exclude dataset features that have a negative impact on the performance of a machine learning model. The value of a feature importance score varies from negative to positive. Any feature with a score of zero or less can be removed from a dataset.

In addition, there are a few drawbacks that should be mentioned. Firstly, some research initiatives suggest that their proposed dataset includes data from multiple sources, highlighting the heterogeneity feature [18], [19]. However, these sources only produce network traffic-related data, resulting in a dataset that consists of exclusively network-related data. Nevertheless, Kyoto 2006+ added 3 more features to their dataset that are a result of a non-network-related source of data. Despite that, the authors do not mention the actual source of the extra data.

Another issue that needs to be mentioned is the process of evaluating a dataset, specifically the comparison of common characteristics between datasets. This procedure only assesses the properties (characteristics) of a dataset and not its overall quality or performance when used as a training dataset of a machine learning algorithm. This is partially mitigated by some proposals, where machine learning models are created by using the dataset and showcasing the training metrics. Even though that is a better approach than the previous one, most of the papers do not compare their results with other datasets' results. In addition, the trained machine learning models are never used in a real network environment to establish how well they can predict real attacks after they have been trained.

In this section, benchmark and well-known cybersecurity datasets were presented, the evaluation processes were highlighted, as well as the drawbacks that were identified. To address these drawbacks, we propose a situational awareness-oriented Network Intrusion Detection dataset that:

1. contains network-related data as well as data from heterogeneous sources, namely: information from a vulnerability assessment tool and system logs.
2. consists of network traffic that was captured from a realistic network topology with real cyberattacks and benign network activity.
3. was evaluated by training 6 different machine learning models and deploying them to a real network to observe their post-training efficiency.
4. includes a set of features that were chosen based on the feature importance calculation.

Ultimately, the machine learning models' training and real performance metrics are showcased and compared with 2 benchmark datasets. To the best of our knowledge, nothing similar has been performed yet.

# Chapter 3 - Implementation

In this section, the creation process of the heterogeneous dataset is presented. The applied network topology, the utilized software, and the dataset post-creation adjustments are also showcased, so that every step of this process can be reproduced by other researchers. Furthermore, the technologies utilized are presented in Appendix C - Technology Enablers.

## Network Topology

Firstly, a virtual switch was created and used as the link between systems to capture the network traffic in an isolated environment. Then, systems that have the role of "victims" were deployed. Those systems will be referred to as benign systems. A benign system produces network traffic without the presence of malicious activities, and this traffic will be referred to as "benign network traffic". Windows Server 2008 R2 was chosen as the first benign system, while Ubuntu Server 20.04 was the second benign system that was deployed. Both benign systems had their default firewall systems enabled. Following the deployment of the benign systems, an "attacker" system was put into place. This system will be referred to as the "malicious system". The Kali Linux 2021.3 operating system was chosen as the attacker. Furthermore, the network activity generated by this system will be referred to as "malicious network traffic".

After the deployment of every system, the benign systems were modified to be susceptible to cyberattacks. For that matter, older versions of software were installed on the Windows server, so that it could be vulnerable to cyber threats. However, some attacks can be launched even if a system does not have vulnerabilities. For example, denial of service (DoS) is an attack that can damage an up-to-date system that does not contain vulnerabilities [25]. Therefore, Ubuntu server had up-to-date software versions.

Even though most of the tools to perform cyberattacks were included in the malicious system, the software to execute some attacks was not available. Therefore, those attacks were performed by creating custom Python scripts. Attacks, enumeration of vulnerabilities, their descriptions, and their corresponding benign systems are presented in Table 1.

**Table 1. Cyberattacks and their corresponding benign systems**

| *Attack* | *Vulnerability Enumeration* | *Software* | *Benign System* | *Attack Description* |
|---|---|---|---|---|
| Remote File Inclusion [26] | CVE-2014-3120 | Elasticsearch | Windows Server 2008 R2 | Unauthorized file upload to gain administrative privileges in a system |
| | CVE-2015-8249 | ManageEngine Remote Desktop 9 | | |
| | CVE-2016-1209 | WordPress Ninja Forms 2.9.42 | | |
| SSH Brute Force [27] | CVE-2011-0553 | Open 22 Port with OpenSSH Server Service | | A large number of SSH login attempts is sent to guess credential combination |
| ICMP Ping Flood [28] | - | - | | A large number of ICMP packets is sent to decrease a system's performance |
| DoS Slowloris [25] | CVE-2007-6750 | Apache2 Server | Ubuntu Server 20.04 | A large number of HTTP requests that slows a system's service and a system's performance |
| TCP Fingerprinting [29] | - | - | | Network activity that scans a system to determine its operating system and/or the services that are running. |

As mentioned before (Chapter 1 - Introduction), system and firewall logs are part of the heterogeneous dataset. With benign and malicious systems and their associated software set up, unique firewall rules were created for the Windows firewall, so that it could generate system logs, when specific cyberattacks are taking place. In addition, the Wazuh (Appendix C - Wazuh) server was deployed as a host monitoring system to detect remote file inclusion attacks [26] and, thus, produce system logs. Custom rules were created to monitor core system directories and

directories that had the files of vulnerable software. A Wazuh agent, which sent alerts to the Wazuh server in the event of a cyberattack, was installed on the benign systems. A diagram of the network topology is showcased in Figure 1, and a flow diagram of the described process is presented in Figure 2.

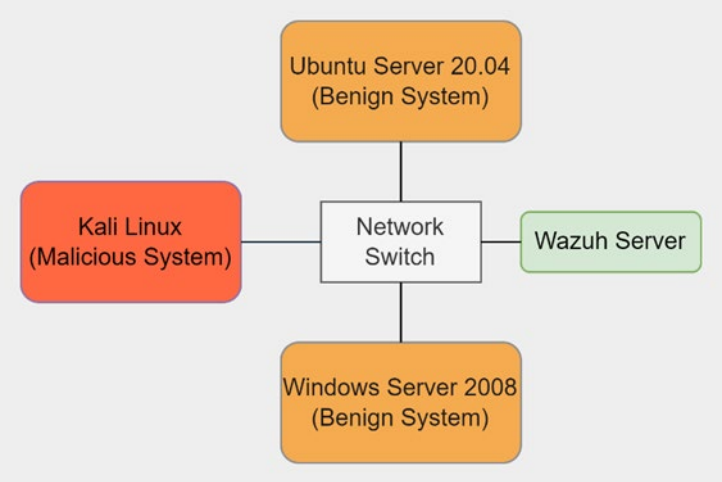



**Figure 1. Network Topology**

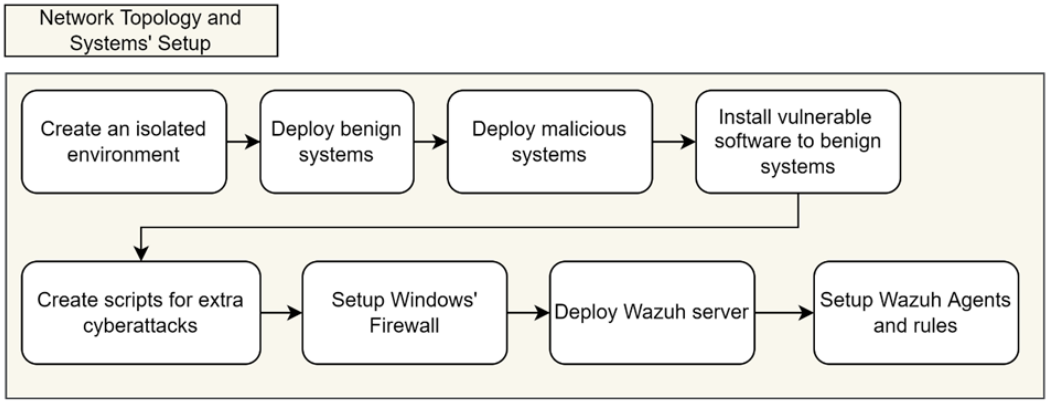



**Figure 2. Network topology and systems' setup process**

## Data Collection Process

The first step in the data collection process is the capture of network traffic. Firstly, the benign network traffic was captured. For the traffic to be as realistic as possible, the benign systems were operated by real users and the network traffic that was generated was captured (transactions among benign systems' services and internet browsing traffic). As expected,

firewall systems were not triggered during this process. The next step involved the capture of malicious network traffic. At this stage, the network-related part of the attacks, as well as the system logs produced by the attacks, were captured. To capture the cyberattacks' network footprint, Wireshark and tcpdump (Chapter 3 - Wireshark) were set up on the benign systems. Produced system logs were also collected while the cyberattacks were initiated. The flow of the data collection process is shown in Figure 3.

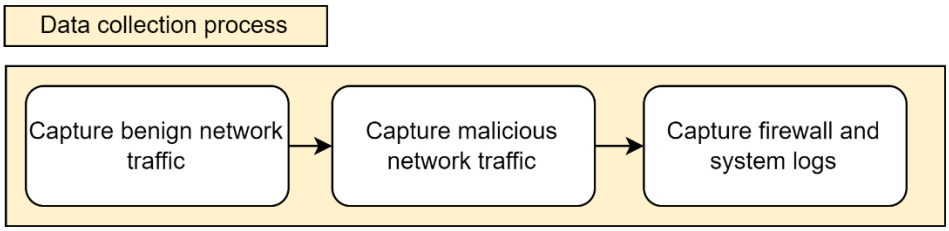



**Figure 3 : Data collection process flow diagram**

## Data Pre-processing

Ultimately, the heterogeneous dataset consists of data from three heterogeneous sources, network, vulnerability assessment, and system and firewall logs. After the data acquisition process, a pre-process phase followed, to aggregate and correlate the collected data. The steps taken to create the final version of the dataset are described below and presented as a flow diagram in Figure 4.

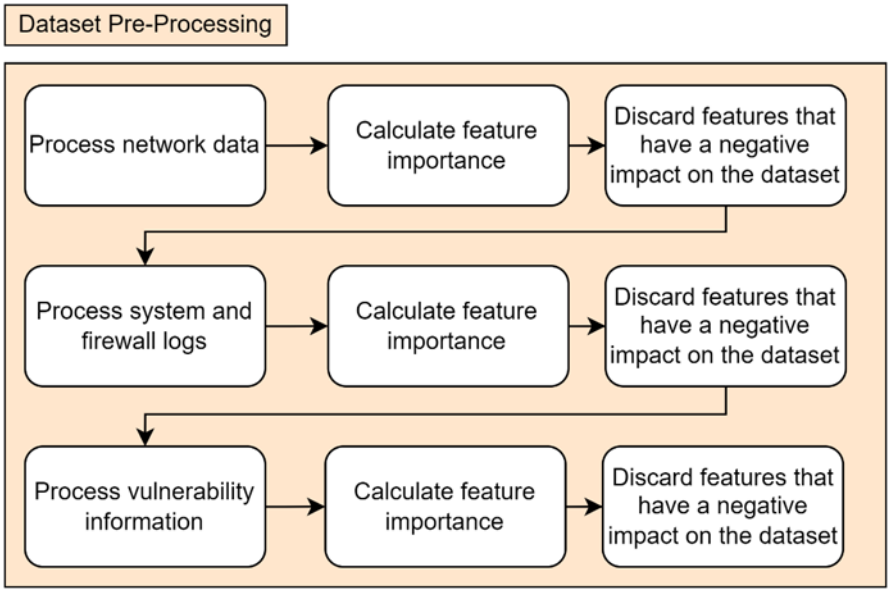



**Figure 4. Dataset pre-processing flow diagram**

## Network Data

After the benign and malicious network traffic capturing processes were finished, the gathered network data were converted to NetFlow v5 network flows. The collected network flows of both benign and malicious network traffic were aggregated into a csv[9] file, forming the initial version of the dataset. Since this is NetFlow v5 protocol data that was derived from CICFlowMeter, the dataset consisted of 83 network-related features (Figures B1, B2, B3, and B4). Thereafter, every record in this dataset was labelled. One feature, named "Label", was added to characterize each network flow transaction. Benign network traffic records were labelled as "normal". A different label was assigned for each cyberattack that was deployed. These labels and their corresponding attacks are showcased in Table 2.

**Table 2. Dataset labels and their corresponding cyberattacks**

| ***Cyberattacks*** | ***Dataset Labels*** |
|---|---|
| Remote File Inclusion | RFI |
| SSH Brute Force | SSH |
| ICMP Ping Flood | PingFlood |
| DoS Slowloris | SlowLoris |
| TCP Fingerprinting | TcpFingerprinting |

After the creation of the initial form of the dataset, a feature importance calculation process followed. The Info Gain method, which is one of the most popular and efficient attribute evaluation methods [30], was utilized. Results of this evaluation reported that 24 out of 83 features negatively impacted the dataset and, thus, were removed. The removed features are showcased in Table 4.

**Table 3. Removed Netflow features and their description (cont. in Table 4)**

| ***Removed Netflow Features*** | ***Feature Description*** |
|---|---|
| Flow ID | A unique identifier integer for each flow |
| Src IP | Source IP |
| Src Port | Source Port |
| Dst IP | Destination IP |
| Dst Port | Destination Port |

[9] https://en.wikipedia.org/wiki/Comma-separated_values

**Table 4. Removed Netflow features and their description**

| ***Removed Netflow Features*** | ***Feature Description*** |
|---|---|
| Protocol | Network protocol of the flow |
| Timestamp | Flow creation timestamp |
| Bwd Pkts/b Avg | Average number of packets bulk rate in the forward direction |
| Fwd PSH Flags | Number of times the PSH flag was set in packets travelling in the forward direction (0 for UDP) |
| Bwd Blk Rate Avg | Average number of bulk rate in the backward direction |
| Fwd Blk Rate Avg | Average number of bulk rate in the forward direction |
| Init Fwd Win Byts | The total number of bytes sent in initial window in the forward direction |
| Fwd Seg Size Min | Minimum segment size observed in the forward direction |
| Bwd Byts/b Avg | Average number of bytes bulk rate in the backward direction |
| CWE Flag Count | Number of packets with CWE |
| Fwd Pkts/b Avg | Average number of packets bulk rate in the forward direction |

## System and Firewall Logs

System logs display information about events that occur on individual systems in a network, like file integrity or permission changes, while firewall logs provide information about network-related events, such as unauthorized port access or excessive use of packets on a specific service. In the case of cyber security incidents, logs are a way to detect, and possibly an enabler to mitigate, a cyberattack. However, the monitoring of services requires human interaction, which is not always feasible. Therefore, as an initial enhancement to the network dataset, system and firewall logs were integrated into the dataset. In this section, the process of enhancing the dataset is described.

Datasets usually contain two types of data: numerical and categorical. System logs are, for the most part, categorical data. However, machine learning algorithms and neural networks require numerical data as an input during their training process. Those values are zero or positive numbers. As a result, several techniques are applied to convert categorical to numerical data. The most common and simple one is the Label Encoding (Ordinal Encoding) technique [31]. In this technique, different integer values are assigned to the distinct categorical ones. For example, if a feature contains three categorical values, numbers one to three would be assigned to the

categorical values accordingly. However, Potdar et al. [32] rated this technique as one of the least efficient encoding techniques, mainly because the numerical difference between encoded values cannot be utilized by an unsupervised algorithm since the numbers assigned to those values are arbitrary. Another popular technique is One-Hot Encoding [33]. By utilizing this method, additional features are added to the dataset that represent each distinct categorical value that exists. Each new feature has the name of the respective categorical value and a value of either zero or one. The value of the feature depends on whether the categorical value is present or not in a dataset record. As a result, if a dataset record contains a certain categorical value, its respective added feature has the value of one. In that case, the rest of the new features, which pertain to the rest of the categorical values except for the one that is present for that dataset record, have a value of zero. As stated in [34], One-Hot Encoding has two main disadvantages. The first one is that it uses a lot of storage space for storing one or zero for each element in a vector. For example, a feature in a dataset that is a categorical variable may have n possible outcomes. Consequently, to store n One-Hot Encoded values of a feature, n times the feature size in the dataset may be needed. The second disadvantage is that the extra vectors that are created have a lower feature importance value.

Therefore, in this thesis, a more advanced technique was utilized for the conversion of categorical data into numerical, namely, sentiment analysis [35]. Sentiment analysis, which is a sub-category of the broader NLP semantic analysis technique [36], can quantify a sentence by providing a score of its sentiment. There are three kinds of sentiments, namely: negative, for scores ranging from -1 to -0.25; neutral, for scores ranging from -0.24 to 0.25; and positive, for score ranges from 0.26 to 1. The logs were processed utilizing this technique to acquire a distinct numerical value based on their sentiment. As a result, some of the drawbacks, which the aforementioned categorical to numerical conversion methods have, were mitigated. For example, the numerical difference between the encoded values is not arbitrary but derived from a specific analysis function. In addition, just one new feature, named "LogSentimentAnalysisScore", was added to the previously created network dataset. Malicious network traffic is always associated with a particular log, but benign network traffic is not. Hence, the value 1, which signifies the most positive sentiment, is added to records that represent benign network traffic. Thereafter, the sentiment analysis scores were normalized to values in the range of 0 to 1. The function used to normalize the values is showcased in Equation 1. Original system and firewall logs, their

sentiment analysis scores, and their respective normalized values are presented in Table 6. After this enhancement, the feature importance calculation procedure was repeated. The results showcased that the newly added feature was one of the top 5, most important, features in the dataset.

$$z_i = \frac{x_i - min(x)}{max(x) - min(x)}$$

**Equation 1. Normalization Function**

**Table 5. System and firewall logs and their sentiment analysis scores**

| *Label* | *Log* | *Sentiment Analysis Score* | *Normalized Sentiment Analysis Score* | *Log Provider* |
|---|---|---|---|---|
| Normal | - | 1 | 1 | - |
| RFI | drop tcp send / file added to system | -0.74 / +0.59 | 0.13 / 0.7949 | Firewall / Wazuh |
| SSH | multiple authentication failures | -0.66 | 0.169 | Wazuh |
| PingFlood | drop icmp receive | -0.11 | 0.445 | Firewall |
| SlowLoris | drop tcp receive | -0.11 | 0.445 | Firewall |
| TcpFingerprinting | - | 0 | 0.5 | - |

## Vulnerability Information

After the previous data gathering processes and dataset pre-processing phases were completed, an extra feature was added in the dataset for each vulnerability in the system (VULN_1, VULN_2, etc.). Each record associated with a vulnerability has a value of one (1) in the respective extra feature, leaving the remaining extra features with a value of zero (0). This method is not to be confused with the already mentioned One-Hot Encoding technique, because in a real-case scenario, where a system may have all the vulnerabilities, every feature would have the value of one (1). Feature importance scores were already calculated in our previous work [37]. Those calculations signified that none of the five new features needed deletion and, thus, were kept. The features of the heterogeneous dataset, and their importance calculations, are presented in Tables 7, 8, and 9.

**Table 6. Features and their feature importance scores (cont. in Table 7)**

| *Feature* | *Feature Importance Score* |
|---|---|
| LogSentimentAnalysisScore | 2.132 |
| Bwd Header Len | 1.5464 |
| Fwd Pkts/s | 1.496 |
| Bwd IAT Tot | 1.4833 |
| Bwd IAT Max | 1.4791 |
| Init Bwd Win Byts | 1.4689 |
| Flow IAT Max | 1.4499 |
| Bwd IAT Mean | 1.4367 |
| Flow Duration | 1.433 |
| Bwd Pkts/s | 1.4102 |
| Flow Pkts/s | 1.3683 |
| Pkt Len Max | 1.3366 |
| Pkt Len Var | 1.3352 |
| Pkt Len Std | 1.3352 |
| Pkt Len Mean | 1.3302 |
| Pkt Size Avg | 1.3185 |
| Flow IAT Mean | 1.306 |
| Fwd Pkt Len Max | 1.2674 |
| Subflow Fwd Byts | 1.267 |
| TotLen Fwd Pkts | 1.267 |
| Fwd Pkt Len Mean | 1.2629 |
| Fwd Seg Size Avg | 1.2629 |
| Fwd Header Len | 1.262 |
| Flow IAT Std | 1.2408 |
| Flow Byts/s | 1.1398 |
| Fwd Pkt Len Std | 1.1121 |
| Fwd IAT Tot | 1.0614 |
| Fwd IAT Max | 1.0611 |
| Fwd IAT Mean | 0.9415 |
| Subflow Bwd Byts | 0.9139 |
| TotLen Bwd Pkts | 0.9139 |
| Bwd Pkt Len Max | 0.9139 |

**Table 7. Features and their feature importance scores**

| *Feature* | *Feature Importance Score* |
|---|---|
| Bwd Pkt Len Std | 0.9123 |
| Bwd Pkt Len Mean | 0.9069 |
| Bwd Seg Size Avg | 0.9069 |
| Subflow Fwd Pkts | 0.8948 |
| Tot Fwd Pkts | 0.8948 |
| Subflow Bwd Pkts | 0.8837 |
| Tot Bwd Pkts | 0.8837 |
| Bwd IAT Std | 0.8664 |
| Bwd IAT Min | 0.7567 |
| Fwd IAT Std | 0.7181 |
| Fwd Act Data Pkts | 0.6874 |
| Idle Min | 0.6345 |
| Idle Max | 0.6332 |
| Idle Mean | 0.6304 |
| VULN_4 | 0.6233 |
| Fwd IAT Min | 0.5841 |
| Flow IAT Min | 0.5532 |
| VULN_5 | 0.5374 |
| Active Mean | 0.4962 |
| Active Max | 0.4892 |
| Active Min | 0.4586 |
| Down/Up Ratio | 0.3955 |
| Idle Std | 0.2128 |
| Active Std | 0.2124 |
| SYN Flag Cnt | 0.2083 |
| VULN_3 | 0.1892 |
| ACK Flag Cnt | 0.1479 |
| VULN_1 | 0.1412 |
| VULN_2 | 0.1412 |
| Fwd Pkt Len Min | 0.137 |
| Bwd PSH Flags | 0.0763 |
| PSH Flag Cnt | 0.0763 |
| FIN Flag Cnt | 0.0213 |

# Chapter 4 - Evaluation

## Aim of the experiment

As our literature review indicated, three evaluation procedures are identified, with feature importance calculation and machine learning training results being the most common ones. Feature importance scores were calculated for the heterogeneous dataset, resulting in 24 negative impacting features, which were removed, while signifying the importance of the extra features. Initial machine learning training results are showcased and discussed below (Results). The deployment of a machine learning model in a real-like network and the assessment of its post-training performance has not been identified as an evaluation method. For the scope of this thesis, this method is proposed and used as the main evaluation method for the heterogeneous dataset as it is the most realistic way of assessing a dataset's performance because it showcases a real-world use-case scenario for the dataset. The goal of this evaluation is to demonstrate that the heterogeneous dataset has better performance than other NIDS datasets in a real-life scenario when used as a training dataset for machine learning algorithms. The flow of the evaluation process is showcased in Figure 5.

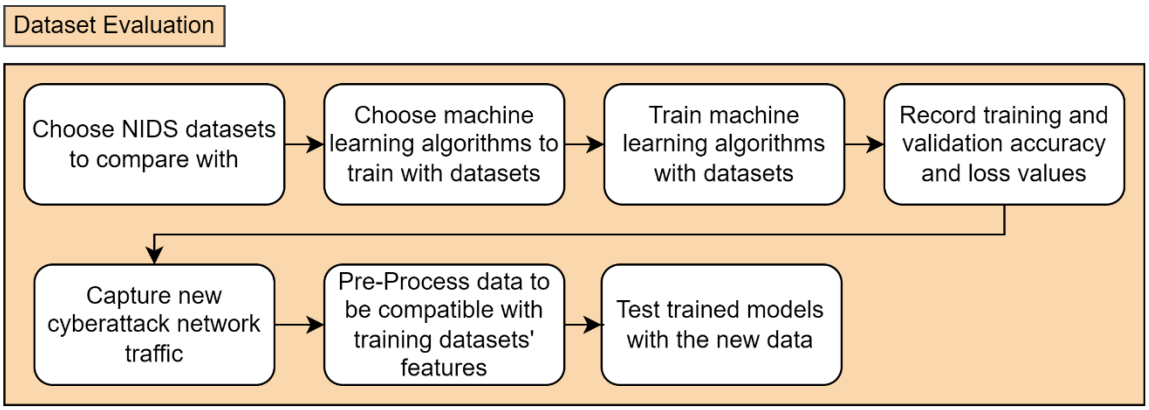



**Figure 5. Dataset evaluation process flow diagram**

## Method

KDD'99 and CCD-IDSv1 datasets were eventually chosen for the comparison experiments. Those two datasets will be referred to as the "comparison datasets". The first one was chosen due to its popularity, and because it is the most used benchmark dataset, and the second one because it is a more modern and newly created dataset.

The first step of this evaluation was to train several machine learning algorithms using every dataset. The machine learning algorithms utilized for this evaluation were acquired from Vinayakumar et al. [38] publication. The chosen algorithms [39] are Multi-Layer Perceptron (MLP), Convolutional Neural Network (CNN), Long Short-Term Memory (LSTM), Deep Neural Network (DNN), and Gated Recurrent Unit (GRU). However, the authors' shared code repository did not work properly upon download, and it was further modified so that every dataset could be used without any errors. Vinayakumar et al. used KDD'99 as a training dataset for those machine learning algorithms and presented the results of the models' training. Consequently, KDD'99 was used as the first training dataset to replicate the same training results. After the initial training, the results were the same as Vinayakumar et al. research. Due to the reproducibility of the results, those algorithms were used throughout the remainder of the evaluation process. CCD-IDSv1 and the heterogeneous dataset, were then used to train the machine learning algorithms.

The next step of the evaluation phase was to test the trained machine learning models with data that were not present on their training phase. To accomplish that, extra benign and malicious network traffic were collected and then used as an input for the trained models. When a machine learning model is trained with a specific set of features, it can only consume data that consist of those exact features. Consequently, after the initial extra data capture process, each of the two comparison datasets' features were replicated so that the new data matches each training dataset's features. This data will be used as an input in the trained machine learning models, ultimately measuring their performance.

Cyberattacks (DoS and SSH brute force) that the datasets have in common were executed on the benign systems and the network traffic produced was captured. In addition, new benign network traffic was captured. The network topology was not changed. Furthermore, no other data capture was needed since the comparison datasets contain only network traffic data. Afterwards, the captured network traffic was converted to match each dataset' features. For the conversion of network data to a KDD'99 dataset, an open-source tool, which converts network traffic data to a KDD'99 dataset, was utilized[10], and for the CCD-IDSv1 dataset, Argus (Chapter 3 - Argus Network Audit System) was used. Finally, every trained model was given their respective test

[10] https://github.com/AI-IDS/kdd99_feature_extractor

dataset to measure its performance after the initial training phase. Three labels were chosen for the final comparison, namely: Benign, SSH BruteForce, and Slowloris DoS.

## Variables

### *Dependent Variables*

Dependent variables are defined as the variables that contain the outcomes of an evaluation or experiment. In this evaluation, machine learning-related metrics were used to evaluate the heterogeneous dataset's performance. For the machine learning training phase, training accuracy and loss, and validation accuracy and loss were used as metrics. Validation and training accuracy values are an interpretable way of measuring a machine learning models' performance. They express the accuracy of the models' predictions at the time of their training. On the other hand, validation and training loss values are the sum of errors that occur during the training phase. Lesser loss values and greater accuracy values signify better performance.

To compare and assess the performance of the trained machine learning models after the initial training, the following four machine learning metrics were chosen: Accuracy (Equation 5), which is the percentage of correct predictions of a machine learning model; Precision (Equation 2), which in this case showcases the percentage of benign traffic that has been correctly classified as benign; Recall (Equation 3), which showcases the percentage of malicious traffic that has been correctly classified as malicious; and F1-Score (Equation 4), which is the harmonic mean of Recall and Precision. Those metrics are acquired using data from the prediction outcomes of a trained model (Appendix A - Confusion Matrices).

$$Precision = \frac{True\ Positive}{\text{True Positive} + \text{False Positive}}$$

**Equation 2 : Precision Equation**

$$Recall = \frac{True\ Positive}{\text{True Positive} + \text{False Negative}}$$

**Equation 3 : Recall Equation**

$$F1_Score = \frac{Precision * Recall}{\text{Precision} + \text{Recall}}$$

**Equation 4 : F1-Score Equation**

$$Accuracy = \frac{True\ Positive + True\ Negative}{\text{True Positive} + \text{True Negative} + \text{False Positive} + \text{False Negative}}$$

**Equation 5 : Accuracy Equation**

### *Independent Variables*

Independent variables are the variables that are constantly being modified to observe changes in the dependent variables. In this experiment, the datasets and the machine learning models were the independent variables since a change to at least one of them caused the training accuracy and loss, and validation accuracy and loss variables to produce different values. Furthermore, the dataset that is provided to the machine learning models after their training to measure their performance is also an independent variable, since its features were modified so that it can be utilized on every model and, thus, produce different results for the accuracy, precision, recall, and f1 score dependent variables.

### *Fixed Variables*

Fixed variables are variables that do not change during an experiment. During the evaluation phase, the network topology, benign and malicious systems, vulnerabilities, cyberattacks, and machine learning algorithms remain unaltered.

## Prediction

Machine learning training results using CCD-IDSv1 are showcased in [21] , and those for KDD'99 are presented in [38]. Both studies presented over 90% accuracy scores for the training and validation of multiple machine learning algorithms using those datasets. In this evaluation, the same machine learning model training results are expected. As for the heterogeneous dataset, since the features that negatively impacted it were discovered and removed through the feature importance calculation process, it is also expected to present machine learning training metrics that are greater than 90%. In addition, the same is expected for the post-training model

evaluation for the heterogeneous dataset and CCD-IDSv1. However, KDD'99 is expected to produce lower post-training results than the other two for both correct malicious and benign traffic predictions due to the many duplicate records that this dataset contains [40]. That may cause a machine learning model to overfit. Overfitting is a state where a machine learning model is trained upon many duplicate, or irrelevant, records and eventually produces very high-accuracy results [41]. However, the model cannot achieve the same high scores in post-training predictions.

## Results

The bar charts showcase how well the machine learning models performed when trained with every dataset. Furthermore, the charts also present the performance of every trained machine learning model after the initial training. The average values of all the results are showcased in Tables 8 and 9.

**Table 8. Average machine learning scores for binary classification**

| ***Binary Classification, Average Scores*** | *Average Training Accuracy* | *Average Training Loss* | *Average Prediction Precision* | *Average Prediction Recall* | *Average Prediction Accuracy* | *Average Prediction F1-Score* |
|---|---|---|---|---|---|---|
| *Heterogeneous Dataset* | 0.945 | 0.118 | **0.98** | **0.92** | **0.94** | **0.95** |
| *KDD'99* | **0.995** | **0.015** | 0.81 | 0.59 | 0.67 | 0.66 |
| *CCD-IDSv1* | 0.983 | 0.035 | 0.95 | 0.1 | 0.65 | 0.17 |

**Table 9. Average machine learning scores for multiclass classification**

| ***Multiclass Classification, Average Scores*** | *Average Training Accuracy* | *Average Training Loss* | *Average Prediction Precision* | *Average Prediction Recall* | *Average Prediction Accuracy* | *Average Prediction F1-Score* |
|---|---|---|---|---|---|---|
| *Heterogeneous Dataset* | 0.917 | 0.206 | **0.9** | **0.9** | **0.9** | **0.9** |
| *KDD'99* | **0.991** | **0.029** | 0.61 | 0.56 | 0.56 | 0.58 |
| *CCD-IDSv1* | 0.96 | 0.115 | 0.52 | 0.58 | 0.58 | 0.55 |

### *Binary Classification – Training Results*

The first two (2) bar charts (Figure 6 and Figure 7) present the training results for binary classification. Those charts showcase that more than half of the algorithms have better results

when trained with the KDD'99 dataset. As stated in [40], KDD'99 includes many duplicate records and, thus, produces bias that causes the machine learning models to overfit. As for the CCD-IDSv1, an average of 98% training accuracy was achieved, which was expected based on the literature [21].

Furthermore, the heterogeneous dataset has higher loss values than the comparison datasets. This occurs because the heterogeneous dataset contains records that have greater numerical differences because they are not synthetically created but realistically captured. For the same reason, there is an average of 0.01 increase in validation loss in comparison to training loss. Still, the average loss score for the heterogeneous dataset is 0.2, which is low enough to indicate that the machine learning algorithms are well-trained [42].

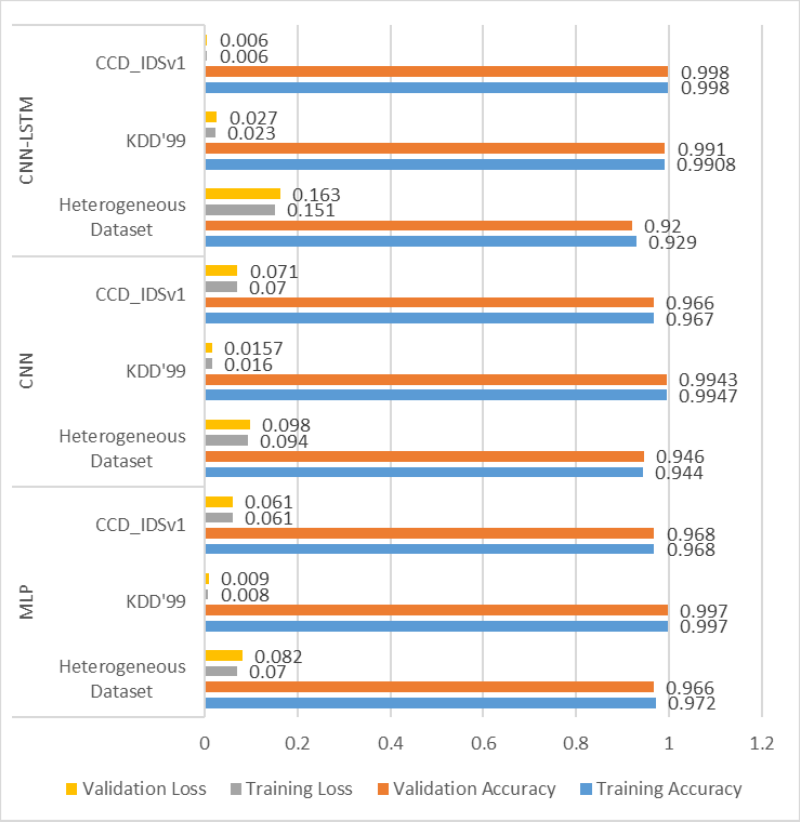



**Figure 6. Binary Classification - Training Results (MLP, CNN, CNN-LSTM)**

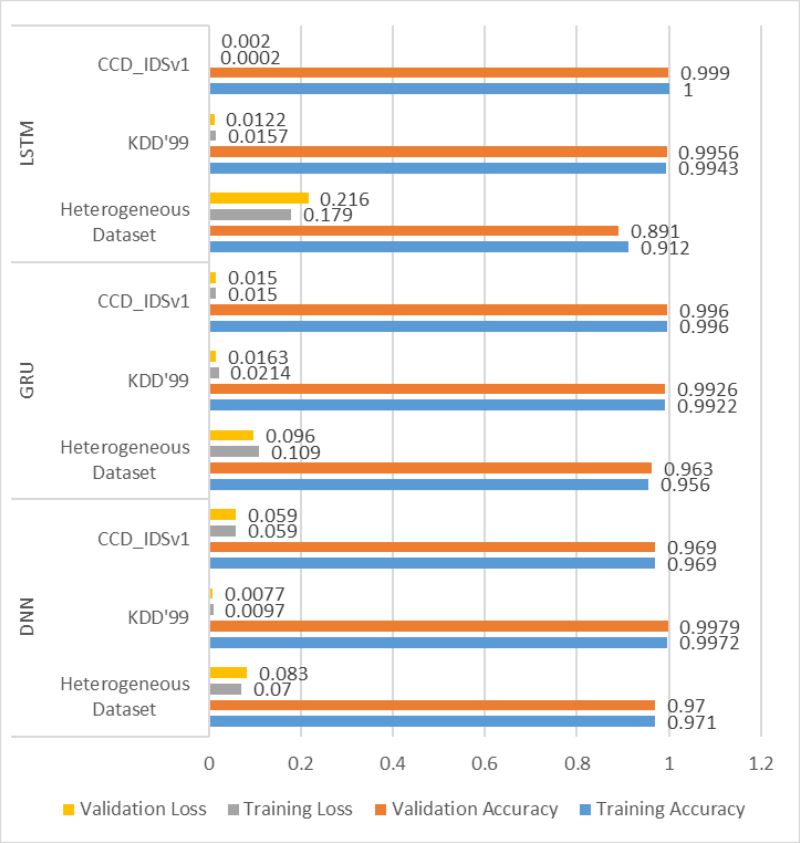



**Figure 7. Binary Classification - Training Results (DNN, GRU, LSTM)**

### *Binary Classification – Post-Training Evaluation Results*

The bar charts in Figure 8 and Figure 9 showcase the post-training results for binary classification. As illustrated, the machine learning models that were trained with KDD'99 are able to identify benign traffic better than others (higher precision scores). Almost all models, except for MLP and GRU, presented benign prediction results with over 90% correct predictions when trained with KDD'99 and CCD-IDSv1. The heterogeneous dataset produced consistent results, with its models scoring over **96**% of correct predictions in benign traffic.

For the prediction of malicious traffic (recall score), the models that were trained with the heterogeneous dataset demonstrate significantly better results than the models that were trained with the comparison datasets. The percentage of the correct malicious predictions on average for every model is **92**% for the heterogeneous dataset, 59% for KDD'99, and 9% for CCD-IDSv1.

Since the models that were trained with the heterogeneous dataset presented consistent training results with high accuracy scores and low loss scores, it was expected to produce an average greater than 90% in post-training malicious traffic predictions. Furthermore, the models that were trained with KDD'99 were expected to produce an average between 50 to 75 percent of correct malicious predictions post-training. However, the post-training performance of the models that were trained with CCD-IDSv1 was not expected to demonstrate that level of decrease. This may be the case because CCD-IDSv1 contains network flows of three (3) similar DoS attacks (UDP DoS [43], ARP DoS [44] and Slowloris DoS). Despite the fact that these attacks are performed on different protocol stacks, CCD-IDSv1 contains network flows that do not indicate to which protocol each attack belongs. Consequently, CCD-IDSv1-trained machine learning models showcase decreased performance in post-training predictions.

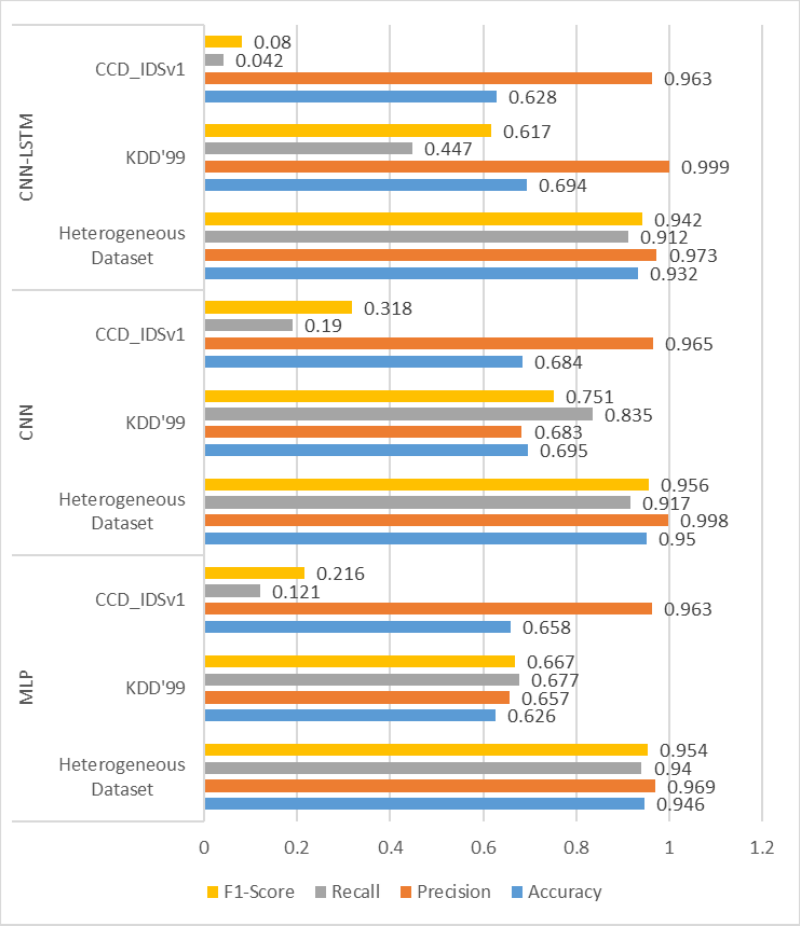



**Figure 8. Binary Classification - Post-Training Evaluation Results (MLP, CNN, CNN-LSTM)**

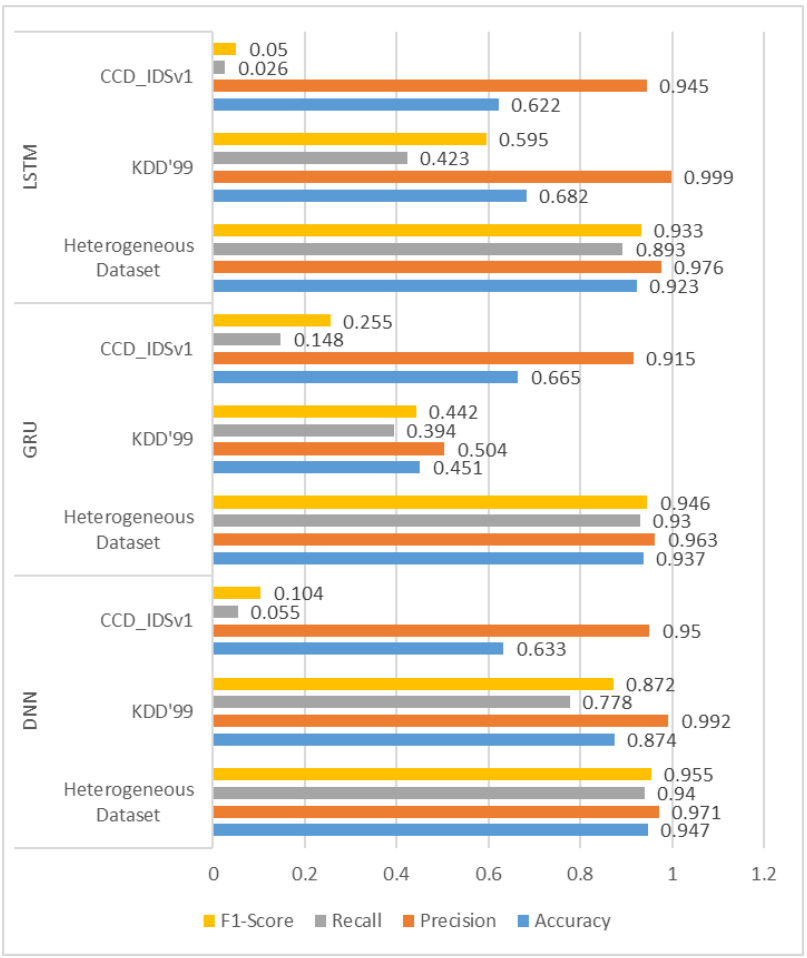



**Figure 9. Binary Classification - Post-Training Evaluation Results (DNN, GRU, LSTM)**

### *Multiclass Classification – Training Results*

Figures 10 and 11 showcase the training results of the multiclass classification. An increase in the accuracy (+0.2) of the CNN-LSTM model that was trained with CCD-IDSv1 is illustrated. However, loss values are also increased for all the models trained with CCD-IDSv1. Almost all the models that were trained with KDD'99 produce better accuracy and loss values, except for GRU and LSTM. The figures also indicate a slight decrease in the performance of all the models. Ultimately, models trained with KDD'99 present the best training results. Overall, all the models showcase almost similar training metrics as binary training.

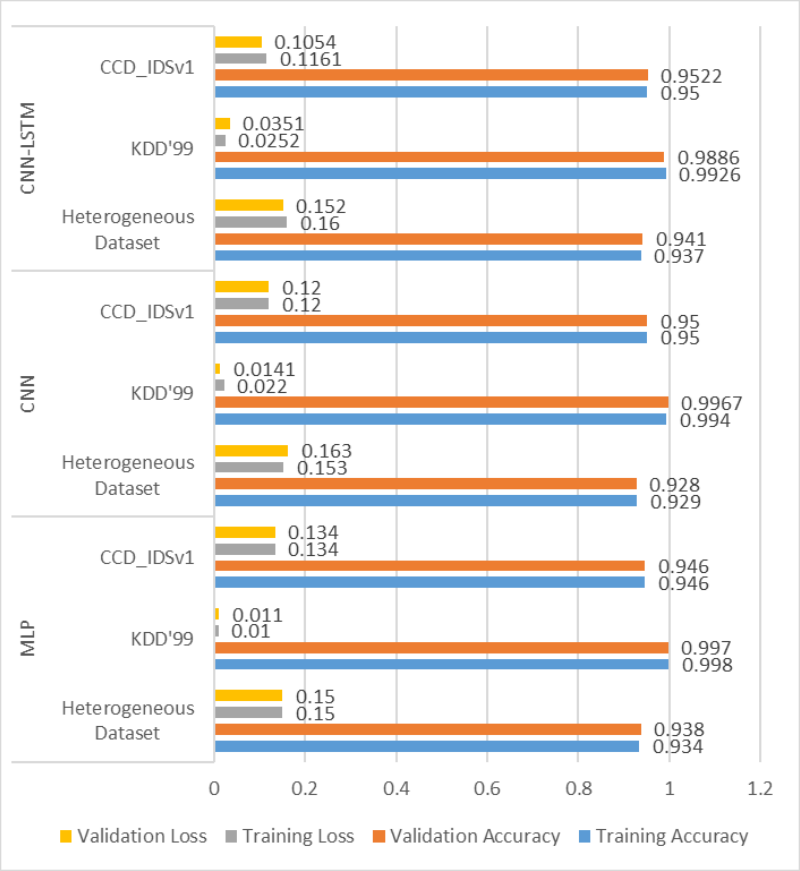



**Figure 10. Multiclass Classification - Training Results (MLP, CNN, CNN-LSTM)**

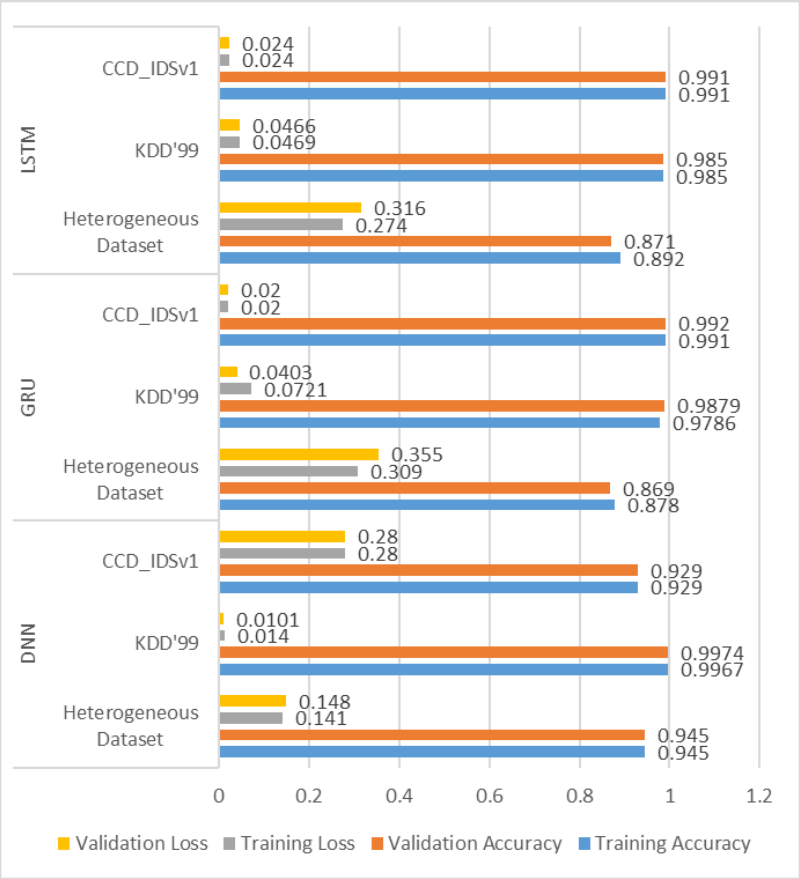



**Figure 11. Multiclass Classification - Training Results (DNN, GRU, LSTM)**

### *Multiclass Classification – Post-Training Evaluation Results*

Figures 12 and 13 showcase the post-training evaluation results for every model. In multiclass classification the models that were trained with the heterogeneous dataset have the highest scoring results with an average of **90**% correct benign traffic predictions and **90**% correct malicious traffic predictions. For KDD'99-trained models, there is an average of 61% correct benign traffic predictions and 55% correct malicious traffic predictions. Lastly, CCD-IDSv1-trained models showcase an average of 52% correct benign traffic predictions and 58% correct malicious traffic predictions. CCD-IDSv1-trained models have a significant increase on the average of correct malicious traffic predictions.

Overall, most of the models showcase a minor decrease in their post-training performance in comparison with binary classification. This indicates that those machine learning algorithms need further modification. However, for this experiment, the machine learning algorithms are considered fixed variables to eliminate bias, and the differences in models' performance that are trained with different datasets are observed.

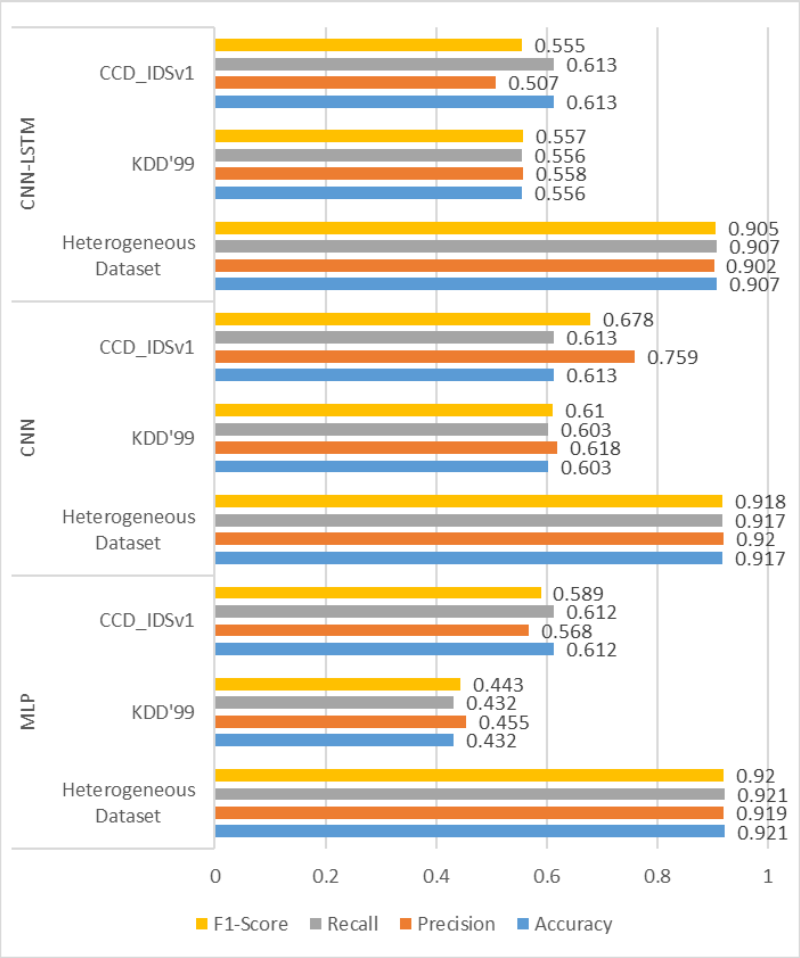



**Figure 12. Multiclass Classification - Post-Training Evaluation Results (MLP, CNN, CNN-LSTM)**

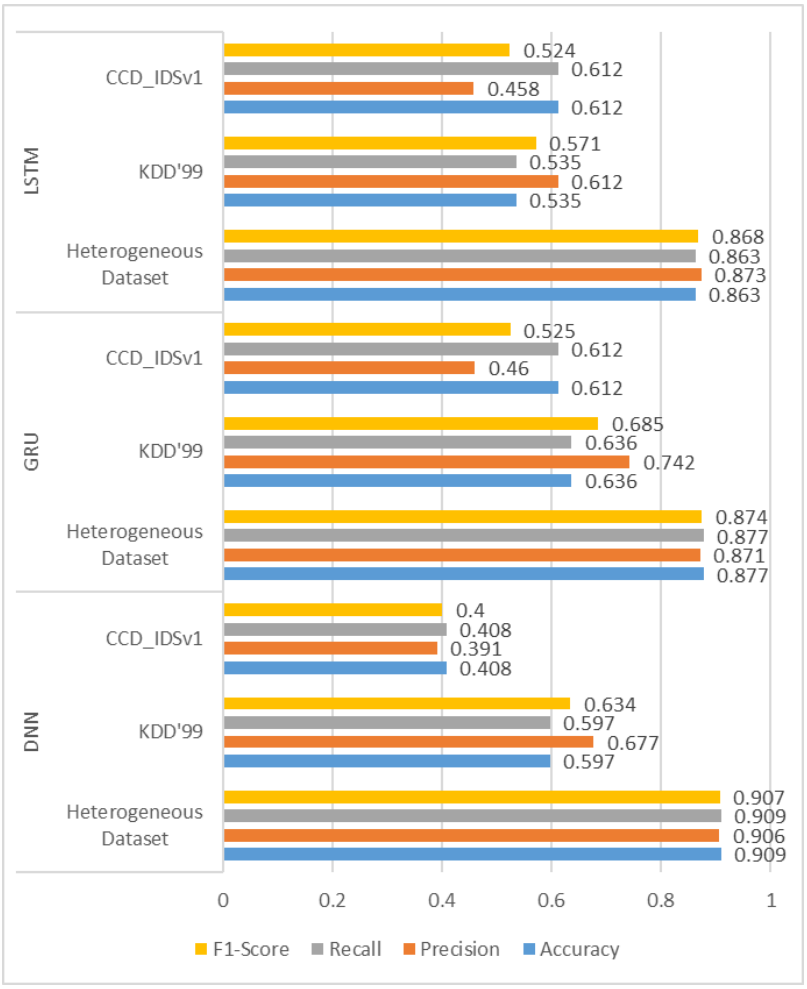



**Figure 13. Multiclass Classification - Post-Training Evaluation Results (DNN, GRU, LSTM)**

### *Evaluation Summary*

- KDD'99-trained models produced the best training results for both binary and multiclass classification. However, the post-training performance demonstrated a decreased overall prediction accuracy for both classifications.
- CCD-IDSv1-trained models produced high training results, but the post-training predictions have an average of 65% accuracy for binary classification, and under 60% average accuracy for multiclass classification.
- The heterogeneous dataset-trained models showcase the most consistent accuracy results with an average of over 90% for both training and post-training prediction accuracy.

# Chapter 5 - Conclusion

In this thesis, the creation steps and the evaluation of a CSA-oriented heterogeneous dataset were presented. ML-based NIDSs can utilize this dataset and, thus, take advantage of the CSA principles (perception, comprehension, projection). The dataset was created using three data types: numerical information from captured network traffic; categorical information from a vulnerability assessment; and firewall and system categorical logs. The collected data were processed, correlated, and ultimately fused.

Furthermore, a more realistic evaluation process for NIDS-related datasets was proposed. Initially, two NIDS-related datasets (KDD'99 and CCD-IDSv1) were selected for comparison with the heterogeneous dataset. Six ML algorithms were trained using every dataset, and their training metrics were compared. The next step of the evaluation phase, which to the best of our knowledge has not yet been proposed, was the deployment of the trained ML models for post-training assessment. Malicious and benign data were captured, forming a new dataset that ML models have not encountered during their training. The dataset was then processed to match the input features of every ML model, resulting in three different versions. Conclusively, the ML models that were trained with the heterogeneous dataset showcased the best overall prediction accuracy (**98**% and **90**%) post-training for both binary and multiclass classification.

## Future Work

Even though the ML models that were trained with the heterogeneous dataset showcased high accuracy results, some improvements can be made to further enhance its performance. First, the network traffic format that was used is NetFlow v5. This is a common protocol for a network traffic dataset, but the newer NetFlow v9/IPFIX, which provides more information about the network traffic, could be used for future investigation [45]. Furthermore, additional research is suggested for the sentiment analysis method used to analyze the firewall and system logs. The logs could be further modified by using the aspect-based sentiment analysis process [46]. This process allows an ML algorithm to be trained with a categorical dataset that is context-specific (e.g., cybersecurity-related keywords) and has a predefined score for each categorical value. After training the ML algorithm with this dataset, the trained ML model can provide more

accurate sentiment analysis for the context-specific keywords that it receives, such as firewall and system logs in this case.

# Bibliography


[1] IETF, “Cisco Systems NetFlow Services Export Version 9,” 2004. [Online]. Available: https://datatracker.ietf.org/doc/html/rfc3954.

[2] E. Parliament, “Cybersecurity: main and emerging threats in 2021 (infographic),” 2022. https://www.europarl.europa.eu/news/en/headlines/society/20220120STO21428/cybersecurity-main-and-emerging-threats-in-2021-infographic#ssh (accessed May 23, 2022).

[3] R. Karlović and I. Sučić, “Security as the basis behind community policing: Croatia’s community policing approach,” in *Community Policing-A European Perspective*, Springer, 2017, pp. 125–138.

[4] J. P. Anderson, “Computer security threat monitoring and surveillance,” *Tech. Report, James P. Anderson Co.*, 1980.

[5] M. Sazzadul Hoque, “An Implementation of Intrusion Detection System Using Genetic Algorithm,” *Int. J. Netw. Secur. Its Appl.*, vol. 4, no. 2, pp. 109–120, 2012, doi: 10.5121/ijnsa.2012.4208.

[6] R. Prasad and V. Rohokale, “Artificial intelligence and machine learning in cyber security,” in *Cyber Security: The Lifeline of Information and Communication Technology*, Springer, 2020, pp. 231–247.

[7] J. Cai, J. Luo, S. Wang, and S. Yang, “Feature selection in machine learning: A new perspective,” *Neurocomputing*, vol. 300, pp. 70–79, 2018, doi: 10.1016/j.neucom.2017.11.077.

[8] D. Chou and M. Jiang, “A Survey on Data-driven Network Intrusion Detection,” *ACM Comput. Surv.*, vol. 54, no. 9, 2022, doi: 10.1145/3472753.

[9] 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,” *Trans. Emerg. Telecommun. Technol.*, vol. 32, no. 1, pp. 1–29, 2021, doi: 10.1002/ett.4150.

[10] M. L. Mathews, P. Halvorsen, A. Joshi, and T. Finin, “A collaborative approach to situational awareness for cybersecurity,” *Collab. 2012 - Proc. 8th Int. Conf. Collab. Comput. Networking, Appl. Work.*, pp. 216–222, 2012, doi: 10.4108/icst.collaboratecom.2012.250794.

[11] M. R. Endsley, “Design and evaluation for situation awareness enhancement,” in *Proceedings of the Human Factors Society annual meeting*, 1988, vol. 32, no. 2, pp. 97–101.

[12] U. Franke and J. Brynielsson, “Cyber situational awareness – A systematic review of the literature,” *Comput. Secur.*, vol. 46, pp. 18–31, Oct. 2014, doi: 10.1016/j.cose.2014.06.008.

[13] K. Shaukat, S. Luo, S. Chen, and D. Liu, “Cyber Threat Detection Using Machine Learning Techniques: A Performance Evaluation Perspective,” *1st Annu. Int. Conf. Cyber Warf. Secur. ICCWS 2020 - Proc.*, 2020, doi: 10.1109/ICCWS48432.2020.9292388.

[14] A. Arqane, O. Boutkhoum, H. Boukhriss, and A. El Moutaouakkil, “A review of intrusion detection systems: Datasets and machine learning methods,” *ACM Int. Conf. Proceeding Ser.*, 2021, doi: 10.1145/3454127.3456576.

[15] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, “A detailed analysis of the KDD CUP 99 data set in Computational Intelligence for Security and Defense Applications,” *Comput. Intell. Secur. Def. Appl.*, no. Cisda, pp. 1–6, 2009.

[16] J. Song, H. Takakura, Y. Okabe, M. Eto, D. Inoue, and K. Nakao, “Statistical analysis of honeypot data and building of Kyoto 2006+ dataset for NIDS evaluation,” *Proc. 1st Work. Build. Anal. Datasets Gather. Exp. Returns Secur. BADGERS 2011*, pp. 29–36, 2011, doi: 10.1145/1978672.1978676.

[17] N. Moustafa and J. Slay, “UNSW-NB15: A comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set),” *2015 Mil. Commun. Inf. Syst. Conf. MilCIS 2015 - Proc.*, 2015, doi: 10.1109/MilCIS.2015.7348942.

[18] I. Sharafaldin, A. Habibi Lashkari, and A. A. Ghorbani, “Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization,” in *Proceedings of the 4th International Conference on Information Systems Security and Privacy*, 2018, vol. 2018-Janua, pp. 108–116, doi: 10.5220/0006639801080116.

[19] N. Moustafa, “A new distributed architecture for evaluating AI-based security systems at the edge: Network TON_IoT datasets,” *Sustain. Cities Soc.*, vol. 72, no. April, p. 102994, 2021, doi: 10.1016/j.scs.2021.102994.

[20] R. Damasevicius *et al.*, “LITNET-2020: An Annotated Real-World Network Flow Dataset for Network Intrusion Detection,” *Electronics*, vol. 9, no. 5, p. 800, May 2020, doi:

10.3390/electronics9050800.

[21] N. Thapa, Z. Liu, A. Shaver, A. Esterline, B. Gokaraju, and K. Roy, “Secure cyber defense: An analysis of network intrusion-based dataset ccd-idsv1 with machine learning and deep learning models,” *Electron.*, vol. 10, no. 15, pp. 1–13, 2021, doi: 10.3390/electronics10151747.

[22] M. S. Al-Daweri, S. Abdullah, and K. A. Zainol Ariffin, “An adaptive method and a new dataset, UKM-IDS20, for the network intrusion detection system,” *Comput. Commun.*, vol. 180, no. February, pp. 57–76, Dec. 2021, doi: 10.1016/j.comcom.2021.09.007.

[23] C. Bailey Lee, C. Roedel, and E. Silenok, “Detection and Characterization of Port Scan Attacks,” *Univeristy California, Dep. Comput. Sci. Eng.*, pp. 1–7, 2003, [Online]. Available: http://cseweb.ucsd.edu/~clbailey/PortScans.pdf.

[24] A. Zien, N. Krämer, S. Sonnenburg, and G. Rätsch, “The feature importance ranking measure,” *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 5782 LNAI, no. PART 2, pp. 694–709, 2009, doi: 10.1007/978-3-642-04174-7_45.

[25] E. Cambiaso, G. Papaleo, and M. Aiello, “Taxonomy of slow DoS attacks to web applications,” *Commun. Comput. Inf. Sci.*, vol. 335 CCIS, no. i, pp. 195–204, 2012, doi: 10.1007/978-3-642-34135-9_20.

[26] H. F. González Robledo, “Types of hosts on a Remote File Inclusion(RFI) botnet,” *Proc. - Electron. Robot. Automot. Mech. Conf. CERMA 2008*, pp. 105–109, 2008, doi: 10.1109/CERMA.2008.60.

[27] M. M. Najafabadi, T. M. Khoshgoftaar, C. Calvert, and C. Kemp, “Detection of SSH brute force attacks using aggregated netflow data,” *Proc. - 2015 IEEE 14th Int. Conf. Mach. Learn. Appl. ICMLA 2015*, pp. 283–288, 2016, doi: 10.1109/ICMLA.2015.20.

[28] A. Kumar, A. K Sharma, and A. Singh, “Performance Evaluation of DVMRP Multicasting Network over ICMP Ping Flood for DDoS,” *Int. J. Appl. Inf. Syst.*, vol. 4, no. 6, pp. 1–6, 2012, doi: 10.5120/ijais12-450106.

[29] G. Taleck, “Synscan: Towards complete tcp/ip fingerprinting,” *CanSecWest, Vancouver BC, Canada*, 2004, [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.161.6737&rep=rep1&type=pdf.

[30] A. G. Karegowda, A. S. Manjunath, G. Ratio, and C. F. Evaluation, “Comparative study

of Attribute Selection Using Gain Ratio," *Int. J. Inf. Technol. Knowl. Knowl. Manag.*, vol. 2, no. 2, pp. 271–277, 2010, [Online]. Available: https://pdfs.semanticscholar.org/3555/1bc9ec8b6ee3c97c524f9c9ceee798c2026e.pdf%0A http://csjournals.com/IJITKM/PDF 3-1/19.pdf.

[31] A. Von Eye and C. C. Clogg, *Categorical variables in developmental research: Methods of analysis*. Elsevier, 1996.

[32] K. Potdar, T. S., and C. D., "A Comparative Study of Categorical Variable Encoding Techniques for Neural Network Classifiers," *Int. J. Comput. Appl.*, vol. 175, no. 4, pp. 7–9, 2017, doi: 10.5120/ijca2017915495.

[33] C. Guo and F. Berkhahn, "Entity Embeddings of Categorical Variables," no. 1, pp. 1–9, 2016, [Online]. Available: http://arxiv.org/abs/1604.06737.

[34] J. T. Hancock and T. M. Khoshgoftaar, "Survey on categorical data for neural networks," *J. Big Data*, vol. 7, no. 1, 2020, doi: 10.1186/s40537-020-00305-w.

[35] W. Medhat, A. Hassan, and H. Korashy, "Sentiment analysis algorithms and applications: A survey," *Ain Shams Eng. J.*, vol. 5, no. 4, pp. 1093–1113, 2014, doi: 10.1016/j.asej.2014.04.011.

[36] D. Hussen Maulud, S. R. M. Zeebaree, K. Jacksi, M. Mohammed Sadeeq, and K. Hussein Sharif, "State of Art for Semantic Analysis of Natural Language Processing," *Qubahan Acad. J.*, vol. 1, no. 2, pp. 21–28, 2021, doi: 10.48161/qaj.v1n2a40.

[37] Y. Nikoloudakis *et al.*, "Towards a machine learning based situational awareness framework for cybersecurity: An SDN implementation," *Sensors*, vol. 21, no. 14, 2021, doi: 10.3390/s21144939.

[38] R. Vinayakumar, K. P. Soman, and P. Poornachandran, "Evaluating effectiveness of shallow and deep networks to intrusion detection system," *2017 Int. Conf. Adv. Comput. Commun. Informatics, ICACCI 2017*, vol. 2017-Janua, pp. 1282–1289, 2017, doi: 10.1109/ICACCI.2017.8126018.

[39] 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," *Trans. Emerg. Telecommun. Technol.*, vol. 32, no. 1, pp. 1–29, Jan. 2021, doi: 10.1002/ett.4150.

[40] O. Atilla and E. Hamit, "A review of KDD99 dataset usage in intrusion detection and

machine learning between 2010 and 2015," *PeerJ*, pp. 0–21, 2016.

[41] T. D1Etterich, "Overfitting and Undercomputing in Machine Learning," *ACM Comput. Surv.*, vol. 27, no. 3, pp. 326–327, 1995, doi: 10.1145/212094.212114.

[42] J. Brownlee, *Better Deep Learning: Train Faster, Reduce Overfitting, and Make Better Predictions*. Machine Learning Mastery, 2018.

[43] C. Kaufman, R. Perlman, and B. Sommerfeld, "DoS protection for UDP-based protocols," p. 2, 2003, doi: 10.1145/948109.948113.

[44] S. Y. Nam, D. Kim, and J. Kim, "Enhanced ARP: Preventing ARP poisoning-based man-in-the-middle attacks," *IEEE Commun. Lett.*, vol. 14, no. 2, pp. 187–189, 2010, doi: 10.1109/LCOMM.2010.02.092108.

[45] M. Sarhan, S. Layeghy, N. Moustafa, and M. Portmann, "NetFlow Datasets for Machine Learning-Based Network Intrusion Detection Systems," *Lect. Notes Inst. Comput. Sci. Soc. Telecommun. Eng. LNICST*, vol. 371 LNICST, pp. 117–135, 2021, doi: 10.1007/978-3-030-72802-1_9.

[46] H. H. Do, P. W. C. Prasad, A. Maag, and A. Alsadoon, "Deep Learning for Aspect-Based Sentiment Analysis: A Comparative Review," *Expert Syst. Appl.*, vol. 118, pp. 272–299, 2019, doi: 10.1016/j.eswa.2018.10.003.

[47] S. Shin and Y. Seto, "Development of IoT security exercise contents for cyber security exercise system," *Int. Conf. Hum. Syst. Interact. HSI*, vol. 2020-June, pp. 281–286, 2020, doi: 10.1109/HSI49210.2020.9142678.

[48] R. M. Muhammad, I. D. Irawati, and M. Iqbal, "Integrated Security System Implementation for Network Intrusion," 2021.

[49] F. Mulyadi, L. A. Annam, R. Promya, and C. Charnsripinyo, "Implementing Dockerized Elastic Stack for Security Information and Event Management," *InCIT 2020 - 5th Int. Conf. Inf. Technol.*, pp. 243–248, 2020, doi: 10.1109/InCIT50588.2020.9310950.

# Appendix A - Confusion Matrices

## Binary Classification Predictions – KDD'99

**Table A 1. KDD'99 - Binary Classification - MLP Confusion Matrix**

| MLP | | |
|---|---|---|
| **True \ Predictions** | **Benign** | **Malicious** |
| **Benign** | 3962 | 3058 |
| **Malicious** | 2797 | 5855 |

**Table A 2. KDD'99 - Binary Classification - CNN Confusion Matrix**

| CNN | | |
|---|---|---|
| **True \ Predictions** | **Benign** | **Malicious** |
| **Benign** | 3670 | 3350 |
| **Malicious** | 1430 | 7222 |

**Table A 3. KDD'99 - Binary Classification - CNN-LSTM Confusion Matrix**

| CNN-LSTM | | |
|---|---|---|
| **True \ Predictions** | **Benign** | **Malicious** |
| **Benign** | 7018 | 2 |
| **Malicious** | 4787 | 3685 |

**Table A 4. KDD'99 - Binary Classification - DNN Confusion Matrix**

| DNN | | |
|---|---|---|
| **True \ Predictions** | **Benign** | **Malicious** |
| **Benign** | 6967 | 53 |
| **Malicious** | 1922 | 6730 |

**Table A 5. KDD'99 - Binary Classification - GRU Confusion Matrix**

| GRU | | |
|---|---|---|
| **True \ Predictions** | **Benign** | **Malicious** |
| **Benign** | 3668 | 3352 |
| **Malicious** | 5247 | 3405 |

**Table A 6. KDD'99 - Binary Classification - LSTM Confusion Matrix**

| LSTM | | |
|---|---|---|
| **True \ Predictions** | **Benign** | **Malicious** |
| **Benign** | 7018 | 2 |
| **Malicious** | 4988 | 3664 |

## Binary Classification Prediction – CCD-IDSv1

**Table A 7. CCD-IDSv1 - Binary Classification - MLP Confusion Matrix**

| MLP | | |
|---|---|---|
| **True \ Predictions** | **Benign** | **Malicious** |
| **Benign** | 4058 | 12 |
| **Malicious** | 2263 | 313 |

**Table A 8. CCD-IDSv1 - Binary Classification - CCN Confusion Matrix**

| CNN | | |
|---|---|---|
| **True \ Predictions** | **Benign** | **Malicious** |
| **Benign** | 4053 | 18 |
| **Malicious** | 2085 | 491 |

**Table A 9. CCD-IDSv1 - Binary Classification - CCN-LSTM Confusion Matrix**

| CNN-LSTM | | |
|---|---|---|
| **True \ Predictions** | **Benign** | **Malicious** |
| **Benign** | 4067 | 4 |
| **Malicious** | 2468 | 107 |

**Table A 10 . CCD-IDSv1 - Binary Classification - DNN Confusion Matrix**

| DNN | | |
|---|---|---|
| **True \ Predictions** | **Benign** | **Malicious** |
| **Benign** | 4063 | 8 |
| **Malicious** | 2434 | 142 |

**Table A 11. CCD-IDSv1 - Binary Classification - GRU Confusion Matrix**

| GRU | | |
|---|---|---|
| **True \ Predictions** | **Benign** | **Malicious** |
| **Benign** | 4036 | 35 |
| **Malicious** | 2194 | 381 |

**Table A 12. CCD-IDSv1 - Binary Classification - LSTM Confusion Matrix**

| LSTM | | |
|---|---|---|
| **True \ Predictions** | **Benign** | **Malicious** |
| **Benign** | 4067 | 36 |
| **Malicious** | 2509 | 66 |

## Binary Classification Predictions – Heterogeneous Dataset

**Table A 13. Heterogeneous Dataset - Binary Classification - MLP Confusion Matrix**

| MLP | | |
|---|---|---|
| **True \ Predictions** | **Benign** | **Malicious** |
| **Benign** | 3367 | 133 |
| **Malicious** | 337 | 4837 |

**Table A 14. Heterogeneous Dataset - Binary Classification - CNN Confusion Matrix**

| CNN | | |
|---|---|---|
| **True \ Predictions** | **Benign** | **Malicious** |
| **Benign** | 3483 | 17 |
| **Malicious** | 573 | 2067 |

**Table A 15. Heterogeneous Dataset - Binary Classification - CNN-LSTM Confusion Matrix**

| CNN-LSTM | | |
|---|---|---|
| **True \ Predictions** | **Benign** | **Malicious** |
| **Benign** | 3329 | 171 |
| **Malicious** | 545 | 2095 |

**Table A 16. Heterogeneous Dataset - Binary Classification - DNN Confusion Matrix**

| DNN | | |
|---|---|---|
| **True \ Predictions** | **Benign** | **Malicious** |
| **Benign** | 3435 | 65 |
| **Malicious** | 397 | 2243 |

**Table A 17. Heterogeneous Dataset - Binary Classification - GRU Confusion Matrix**

| GRU | | |
|---|---|---|
| **True \ Predictions** | **Benign** | **Malicious** |
| **Benign** | 3166 | 334 |
| **Malicious** | 377 | 2263 |

**Table A 18. Heterogeneous Dataset - Binary Classification - LSTM Confusion Matrix**

| LSTM | | |
|---|---|---|
| **True \ Predictions** | **Benign** | **Malicious** |
| **Benign** | 3368 | 132 |
| **Malicious** | 763 | 1877 |

## Multiclass Classification Predictions – KDD'99

**Table A 19. KDD'99 - Multiclass Classification - MLP Confusion Matrix**

| MLP | | | |
|---|---|---|---|
| **True \ Predictions** | **Benign** | **SSH BruteForce** | **SlowLoris DoS** |
| **Benign** | 3781 | 51 | 3240 |
| **SSH BruteForce** | 860 | 1 | 64 |
| **SlowLoris DoS** | 4728 | 3 | 3038 |

**Table A 20. KDD'99 - Multiclass Classification - CNN Confusion Matrix**

| CNN | | | |
|---|---|---|---|
| **True \ Predictions** | **Benign** | **SSH BruteForce** | **SlowLoris DoS** |
| **Benign** | 3753 | 51 | 3303 |
| **SSH BruteForce** | 673 | 21 | 213 |
| **SlowLoris DoS** | 2005 | 21 | 5752 |

**Table A 21. KDD'99 - Multiclass Classification - CNN-LSTM Confusion Matrix**

| CNN-LSTM | | | |
|---|---|---|---|
| **True \ Predictions** | **Benign** | **SSH BruteForce** | **SlowLoris DoS** |
| **Benign** | 3653 | 51 | 3320 |
| **SSH BruteForce** | 1308 | 20 | 375 |
| **SlowLoris DoS** | 1953 | 0 | 5112 |

**Table A 22. KDD'99 - Multiclass Classification - DNN Confusion Matrix**

| DNN | | | |
|---|---|---|---|
| **True \ Predictions** | **Benign** | **SSH BruteForce** | **SlowLoris DoS** |
| **Benign** | 7046 | 51 | 2 |
| **SSH BruteForce** | 933 | 52 | 138 |
| **SlowLoris DoS** | 4328 | 913 | 2329 |

**Table A 23. KDD'99 - Multiclass Classification - GRU Confusion Matrix**

| GRU | | | |
|---|---|---|---|
| **True \ Predictions** | **Benign** | **SSH BruteForce** | **SlowLoris DoS** |
| **Benign** | 6726 | 294 | 2 |
| **SSH BruteForce** | 1570 | 10 | 85 |
| **SlowLoris DoS** | 3732 | 73 | 3300 |

**Table A 24. KDD'99 - Multiclass Classification - LSTM Confusion Matrix**

| LSTM | | | |
|---|---|---|---|
| **True \ Predictions** | **Benign** | **SSH BruteForce** | **SlowLoris DoS** |
| **Benign** | 7078 | 0 | 309 |
| **SSH BruteForce** | 1348 | 0 | 32 |
| **SlowLoris DoS** | 5657 | 0 | 1368 |

## Multiclass Classification Predictions – CCD-IDSv1

**Table A 25. CCD-IDSv1 - Multiclass Classification - MLP Confusion Matrix**

| MLP | | | |
|---|---|---|---|
| **True \ Predictions** | **Benign** | **SSH BruteForce** | **SlowLoris DoS** |
| **Benign** | 4065 | 228 | 12 |
| **SSH BruteForce** | 268 | 6 | 67 |
| **SlowLoris DoS** | 0 | 0 | 0 |

**Table A 26.CCD-IDSv1 - Multiclass Classification - CNN Confusion Matrix**

| CNN | | | |
|---|---|---|---|
| **True \ Predictions** | **Benign** | **SSH BruteForce** | **SlowLoris DoS** |
| **Benign** | 4062 | 2 | 7 |
| **SSH BruteForce** | 1969 | 0 | 342 |
| **SlowLoris DoS** | 83 | 170 | 12 |

**Table A 27. CCD-IDSv1 - Multiclass Classification - CNN-LSTM Confusion Matrix**

| CNN-LSTM | | | |
|---|---|---|---|
| **True \ Predictions** | **Benign** | **SSH BruteForce** | **SlowLoris DoS** |
| **Benign** | 4063 | 1 | 7 |
| **SSH BruteForce** | 2395 | 0 | 45 |
| **SlowLoris DoS** | 87 | 36 | 13 |

**Table A 28. CCD-IDSv1 - Multiclass Classification - DNN Confusion Matrix**

| DNN | | | |
|---|---|---|---|
| **True \ Predictions** | **Benign** | **SSH BruteForce** | **SlowLoris DoS** |
| **Benign** | 2624 | 1 | 1447 |
| **SSH BruteForce** | 1491 | 0 | 153 |
| **SlowLoris DoS** | 838 | 4 | 90 |

**Table A 29. CCD-IDSv1 - Multiclass Classification - GRU Confusion Matrix**

| GRU | | | |
|---|---|---|---|
| **True \ Predictions** | **Benign** | **SSH BruteForce** | **SlowLoris DoS** |
| **Benign** | 4067 | 2 | 2 |
| **SSH BruteForce** | 2447 | 0 | 2 |
| **SlowLoris DoS** | 109 | 4 | 14 |

**Table A 30. CCD-IDSv1 - Multiclass Classification - LSTM Confusion Matrix**

| LSTM | | | |
|---|---|---|---|
| **True \ Predictions** | **Benign** | **SSH BruteForce** | **SlowLoris DoS** |
| **Benign** | 4067 | 2 | 3 |
| **SSH BruteForce** | 2445 | 0 | 3 |
| **SlowLoris DoS** | 108 | 6 | 13 |

## Multiclass Classification Predictions – Heterogeneous Dataset

**Table A 31. Heterogeneous Dataset - Multiclass Classification - MLP Confusion Matrix**

| MLP | | | |
|---|---|---|---|
| **True \ Predictions** | **Benign** | **SSH BruteForce** | **SlowLoris DoS** |
| **Benign** | 4555 | 151 | 0 |
| **SSH BruteForce** | 303 | 507 | 188 |
| **SlowLoris DoS** | 31 | 0 | 398 |

**Table A 32. Heterogeneous Dataset - Multiclass Classification - CNN Confusion Matrix**

| CNN | | | |
|---|---|---|---|
| **True \ Predictions** | **Benign** | **SSH BruteForce** | **SlowLoris DoS** |
| **Benign** | 4362 | 278 | 57 |
| **SSH BruteForce** | 476 | 324 | 198 |
| **SlowLoris DoS** | 34 | 0 | 403 |

**Table A 33. Heterogeneous Dataset - Multiclass Classification - CNN-LSTM Confusion Matrix**

| CNN-LSTM | | | |
|---|---|---|---|
| **True \ Predictions** | **Benign** | **SSH BruteForce** | **SlowLoris DoS** |
| **Benign** | 4658 | 5 | 1 |
| **SSH BruteForce** | 692 | 245 | 96 |
| **SlowLoris DoS** | 36 | 0 | 404 |

**Table A 34. Heterogeneous Dataset - Multiclass Classification - DNN Confusion Matrix**

| DNN | | | |
|---|---|---|---|
| **True \ Predictions** | **Benign** | **SSH BruteForce** | **SlowLoris DoS** |
| **Benign** | 4356 | 339 | 3 |
| **SSH BruteForce** | 252 | 522 | 224 |
| **SlowLoris DoS** | 33 | 0 | 398 |

**Table A 35. Heterogeneous Dataset - Multiclass Classification - GRU Confusion Matrix**

| GRU | | | |
|---|---|---|---|
| **True \ Predictions** | **Benign** | **SSH BruteForce** | **SlowLoris DoS** |
| **Benign** | 4274 | 217 | 106 |
| **SSH BruteForce** | 333 | 373 | 400 |
| **SlowLoris DoS** | 31 | 0 | 403 |

**Table A 36. Heterogeneous Dataset - Multiclass Classification - LSTM Confusion Matrix**

| LSTM | | | |
|---|---|---|---|
| **True \ Predictions** | **Benign** | **SSH BruteForce** | **SlowLoris DoS** |
| **Benign** | 4474 | 56 | 91 |
| **SSH BruteForce** | 581 | 256 | 241 |
| **SlowLoris DoS** | 37 | 0 | 403 |

# Appendix B - Netflow v5 Initial Feature Set

| Feature Name | Description |
|---|---|
| fl_dur | Flow duration |
| tot_fw_pk | Total packets in the forward direction |
| tot_bw_pk | Total packets in the backward direction |
| tot_l_fw_pkt | Total size of packet in forward direction |
| fw_pkt_l_max | Maximum size of packet in forward direction |
| fw_pkt_l_min | Minimum size of packet in forward direction |
| fw_pkt_l_avg | Average size of packet in forward direction |
| fw_pkt_l_std | Standard deviation size of packet in forward direction |
| Bw_pkt_l_max | Maximum size of packet in backward direction |
| Bw_pkt_l_min | Minimum size of packet in backward direction |
| Bw_pkt_l_avg | Mean size of packet in backward direction |
| Bw_pkt_l_std | Standard deviation size of packet in backward direction |
| fl_byt_s | flow byte rate that is number of packets transferred per second |
| fl_pkt_s | flow packets rate that is number of packets transferred per second |
| fl_iat_avg | Average time between two flows |
| fl_iat_std | Standard deviation time two flows |
| fl_iat_max | Maximum time between two flows |
| fl_iat_min | Minimum time between two flows |

**Figure B 14. Initial Netflow v5 Feature Set (cont. in Figure 15)**

| Feature Name | Description |
|---|---|
| fw_iat_tot | Total time between two packets sent in the forward direction |
| fw_iat_avg | Mean time between two packets sent in the forward direction |
| fw_iat_std | Standard deviation time between two packets sent in the forward direction |
| fw_iat_max | Maximum time between two packets sent in the forward direction |
| fw_iat_min | Minimum time between two packets sent in the forward direction |
| bw_iat_tot | Total time between two packets sent in the backward direction |
| bw_iat_avg | Mean time between two packets sent in the backward direction |
| bw_iat_std | Standard deviation time between two packets sent in the backward direction |
| bw_iat_max | Maximum time between two packets sent in the backward direction |
| bw_iat_min | Minimum time between two packets sent in the backward direction |
| fw_psh_flag | Number of times the PSH flag was set in packets travelling in the forward direction (0 for UDP) |
| bw_psh_flag | Number of times the PSH flag was set in packets travelling in the backward direction (0 for UDP) |
| fw_urg_flag | Number of times the URG flag was set in packets travelling in the forward direction (0 for UDP) |
| bw_urg_flag | Number of times the URG flag was set in packets travelling in the backward direction (0 for UDP) |

**Figure 15 Initial Netflow v5 Feature Set (cont. in Figure 16)**

| Feature Name | Description |
|---|---|
| fw_hdr_len | Total bytes used for headers in the forward direction |
| bw_hdr_len | Total bytes used for headers in the forward direction |
| fw_pkt_s | Number of forward packets per second |
| bw_pkt_s | Number of backward packets per second |
| pkt_len_min | Minimum length of a flow |
| pkt_len_max | Maximum length of a flow |
| pkt_len_avg | Mean length of a flow |
| pkt_len_std | Standard deviation length of a flow |
| pkt_len_va | Minimum inter-arrival time of packet |
| fin_cnt | Number of packets with FIN |
| syn_cnt | Number of packets with SYN |
| rst_cnt | Number of packets with RST |
| pst_cnt | Number of packets with PUSH |
| ack_cnt | Number of packets with ACK |
| urg_cnt | Number of packets with URG |
| cwe_cnt | Number of packets with CWE |
| ece_cnt | Number of packets with ECE |

**Figure 16 Initial Netflow v5 Feature Set (cont. in Figure 17)**

| Feature Name | Description |
|---|---|
| down_up_ratio | Download and upload ratio |
| pkt_size_avg | Average size of packet |
| fw_seg_avg | Average size observed in the forward direction |
| bw_seg_avg | Average size observed in the backward direction |
| fw_byt_blk_avg | Average number of bytes bulk rate in the forward direction |
| fw_pkt_blk_avg | Average number of packets bulk rate in the forward direction |
| fw_blk_rate_avg | Average number of bulk rate in the forward direction |
| bw_byt_blk_avg | Average number of bytes bulk rate in the backward direction |
| bw_pkt_blk_avg | Average number of packets bulk rate in the backward direction |
| bw_blk_rate_avg | Average number of bulk rate in the backward direction |
| subfl_fw_pk | The average number of packets in a sub flow in the forward direction |
| subfl_fw_byt | The average number of bytes in a sub flow in the forward direction |
| subfl_bw_pkt | The average number of packets in a sub flow in the backward direction |
| subfl_bw_byt | The average number of bytes in a sub flow in the backward direction |
| fw_win_byt | Number of bytes sent in initial window in the forward direction |
| bw_win_byt | # of bytes sent in initial window in the backward direction |
| Fw_act_pkt | # of packets with at least 1 byte of TCP data payload in the forward direction |
| fw_seg_min | Minimum segment size observed in the forward direction |
| atv_avg | Mean time a flow was active before becoming idle |
| atv_std | Standard deviation time a flow was active before becoming idle |
| atv_max | Maximum time a flow was active before becoming idle |
| atv_min | Minimum time a flow was active before becoming idle |
| idl_avg | Mean time a flow was idle before becoming active |
| idl_std | Standard deviation time a flow was idle before becoming active |
| idl_max | Maximum time a flow was idle before becoming active |
| idl_min | Minimum time a flow was idle before becoming active |

**Figure 17 Initial Netflow v5 Feature Set**

# Appendix C - Technology Enablers

## Virtual Box

VirtualBox[11] is a free and open-source virtualization software. It can be adjusted to provide custom network topologies and host many virtual machines that can operate simultaneously. Because of its ease of use and low resource requirements, VirtualBox has been used to host complex networks in cybersecurity-related research initiatives [47]. In this thesis, it was utilized for the virtualization and hosting of all the systems that were part of the data capture network topology. Furthermore, the network these systems were part of was isolated from the rest of the network but had access to the internet, so that benign traffic could be captured as realistically as possible.

## Kali Linux

Kali Linux[12] is a free and open-source penetration testing and ethical hacking-oriented operating system. It contains many tools that can test a system's defense abilities. Some of those tools are usually used to emulate malicious behavior and cyberattacks. Because of the tools it has, this operating system was used so that malicious network traffic could be generated.

## Windows Server

For an attack to be successfully deployed, benign systems need to be present in the network. Windows Server 2008 R2, which is a server implementation by Microsoft[13], was employed as a one of the benign systems. The choice of this server as a benign system was made because it is still very popular choice among enterprises[14] [15] [16] between the available Microsoft Servers. Older versions of software were installed on this server so that it could serve as a vulnerable machine, thus, susceptible to cyberattacks. On top of that, Windows' firewall is

[11] https://www.virtualbox.org/

[12] https://www.kali.org/

[13] https://www.microsoft.com/en-us/windows-server?SilentAuth=1&wa=wsignin1.0

[14] https://www.cio.com/article/201559/not-dead-yet-windows-server-2008-users-have-options.html

[15] https://www.softchoice.com/blogs/cloud-migration-adoption-management/windows-server-2008-end-of-life-risks-and-opportunities

[16] https://community.spiceworks.com/networking/articles/2462-server-virtualization-and-os-trends

triggered by some malicious network activities, leading to the creation of system logs. Even though a firewall cannot detect every cyberattack, it was enabled for the scope of this thesis so that system logs could be used as a heterogeneous source of input in the heterogeneous dataset.

## Ubuntu Server

Ubuntu Server 20.04, a free and open-source server implementation by Canonical[17], was used as a benign system entity. Among Linux servers, Ubuntu Server is the most popular one[18]. This server was targeted by cyberattacks that can be deployed even if there is no system vulnerability present. As a result, up-to-date versions of software were installed on this server. The system logs that were produced during some cyberattacks were extracted and were used in the heterogeneous dataset.

## Wazuh

As mentioned before, cyberattacks do not always trigger firewall rules or default operating systems' cyber defense software. As a result, host intrusion detection systems can be utilized to monitor the file integrity or the privilege alterations of files that may occur during a cyberattack. Therefore, Wazuh[19], a free and open-source host monitoring software, was employed. Based on the software that was present in benign systems, custom rules were created, so that in the case of a cyber incident, they could be triggered and produce system logs [48] [49]. These logs were also utilized for the creation of the heterogeneous dataset.

## Python Programming Language

Even though Kali Linux was exclusively used for the emulation of malicious behavior, it did not contain the tools to emulate certain attacks. As a result, custom scripts to emulate those attacks were created with Python[20], a programming language that is highly extensible due to a wide variety of libraries. In addition to cyber incident emulation, Python was also used for data and dataset pre-processing, as well as for the development of machine learning algorithms and

[17] https://canonical.com/

[18] https://w3techs.com/technologies/details/os-linux

[19] https://wazuh.com/

[20] https://www.python.org/

neural network models. The models were created with the TensorFlow 2[21] python library. TensorFlow 2 is a free and open-source library that is developed by Google and enables the creation of machine learning and custom neural network models. Python scripts were also utilized for the conversion of captured network traffic data to a KDD'99-compatible dataset, which was used for the heterogeneous dataset evaluation process.

## Argus Network Audit System

Argus[22] is a free and open-source network flow extractor. It can convert captured network traffic to a network flow dataset. Argus was utilized by Thapa et al. [21] for the creation of the CCD-IDSv1 dataset, and for the scope of this thesis, it is used for the conversion of captured network traffic data to a CCD-IDSv1-compatible dataset. This dataset was used in the heterogeneous dataset evaluation process.

## Weka

Weka[23] is a free and open-source tool that provides a wide variety of machine learning algorithms to test a dataset with, dataset statistical analysis functions, as well as functions for feature importance calculation [24]. Weka was used for its latter feature in this thesis.

## Text2Data

System and firewall logs were collected to be used as part of the heterogeneous dataset. As a result, that categorical data needed to be converted to numerical data to be integrated into the dataset. All the logs were converted to numerical data with a sentiment analysis process, provided by Text2Data. Text2Data [24] is a tool that can provide a sentiment score for a sentence, with values between -1 and 1, with -1 being negative sentiment and 1 being positive sentiment. Ultimately, the scores of each sentence were used for the dataset.

---

[21] https://www.tensorflow.org/

[22] https://openargus.org/

[23] https://www.cs.waikato.ac.nz/ml/weka/index.html

[24] https://text2data.com/

## Wireshark

Wireshark[25], which is free and open source, is a network protocol analyzer and network packet capturing tool. Wireshark, and its command-line version, tcpdump, were exclusively used for the capture of network traffic. This tool can capture the traffic in data files that contain every packet that exists in a network transaction. Such files can be further processed so that network flow transactions can be derived.

## CICFlowMeter

For the scope of this thesis, network traffic was utilized as network flows instead of raw captured packet transactions. CICFlowMeter[26] is a free and open-source tool that can convert network traffic that was captured by Wireshark, into network flow transactions, specifically in the form of the NetFlow v5 standard. The network flow transactions are the emitted into a Comma Separated Values (CSV) file that contains 83 network-related features.

## OpenVAS

A vulnerability assessment is a method of determining if a system, or a set of systems, is vulnerable to cyberattacks and, as such, can be a valuable source of information for cybersecurity experts. The vulnerability assessment report is examined by an analyst, and appropriate measures are performed. The following items are included in the report: the detected vulnerability description, the severity of the vulnerability, the port on which the vulnerability was identified, and the official enumeration of the vulnerability. Data from a vulnerability assessment report was used to broaden the sources from which the evaluation dataset will obtain information while automating the review process of the vulnerability assessment report. Information stemming from a vulnerability assessment tool was used as an extra input source for the heterogeneous dataset. OpenVAS[27], a free and open-source software, was utilized to scan the benign systems and produce a holistic vulnerability assessment report.

---

[25] https://www.wireshark.org/

[26] https://www.unb.ca/cic/research/applications.html#CICFlowMeter

[27] https://openvas.org/