National College of Ireland

# Using feature selection to improve intrusion detection

MSc Research Project
Master's in cyber security

Jordan Cogan
Student ID: 15392331

School of Computing
National College of Ireland

Supervisor: Vikas Sahni

**National College of Ireland**

**MSc Project Submission Sheet**

**School of Computing**

| | |
|---|---|
| **Student Name:** | Jordan Cogan |
| **Student ID:** | 15392331 |
| **Module:** | Research Project/Internship |
| **Programme:** | Master's in cyber security |
| **Year:** | 1 |
| **Supervisor:** | Vikas Sahni |
| **Submission Due Date:** | 6th September 2021 |
| **Project Title:** | Using feature selection to improve intrusion detection? |
| **Word Count:** | 5026 |
| **Page Count:** | 19 |

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

**Signature:** Jordan Cogan

**Date:** 02/09/2021

**PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST**

| | |
|---|---|
| Attach a completed copy of this sheet to each project (including multiple copies) | □ |
| **Attach a Moodle submission receipt of the online project submission,** to each project (including multiple copies). | □ |
| **You must ensure that you retain a HARD COPY of the project**, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer. | □ |

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

| **Office Use Only** | |
|---|---|
| Signature: | |
| Date: | |
| Penalty Applied (if applicable): | |

# Using feature selection to improve intrusion detection

Jordan Cogan
15392331

**Abstract**

Recent high profile cyber-attacks have further showed the importance of cyber security in society, with many smaller organizations now starting to employ external security assistance. The basis of any successful cyber security team is the Security Operations Centre (SOC). The SOC team are responsible for monitoring and mitigating potential threats. The key to a successful SOC team is their Security Information and Event Management system (SIEM) which can be used to monitor Intrusion Detection traffic. The more reliable that a SIEM is at detecting intrusion traffic, the more successful the SOC team can be. Although many researchers have proposed a variety of ways to improve intrusion detection, this paper proposes a model which makes use of multi-layer feature selection. The Random Forest and K-nearest neighbours classifiers were applied after the multi-layer feature selection and evaluated. The Random Forest outperformed K-nearest neighbours overall and achieved 98% for its accuracy, while also achieving 99% for precision, recall and F1-score. Therefore, the multi-layer feature selection model can be considered an appropriate solution at improving intrusion detection.

**Keywords:** Intrusion detection, Machine Learning, Feature Selection, Random Forest, KNN, Chi-2, KDD99.

# 1 Introduction

## 1.1 Key Terms

| Term | Definition |
|---|---|
| False Positive | Falsely recognising traffic as Attacking Intrusion traffic |
| True Positive | Correctly recognising traffic as Attacking Intrusion traffic |
| False Negative | Falsely recognising traffic as Non-Attacking Intrusion traffic |
| True Negative | Correctly recognising traffic as Non-Attacking Intrusion traffic |
| DOS traffic | Denial of Service traffic |
| Probe traffic | Probing traffic |
| R2L traffic | Root to Local traffic |
| U2R traffic | User to Root traffic |

## 1.2 Background and Motivation

The use of technology is ever increasing. This brings with it huge benefits to society but also some major challenges. In 2020 alone businesses saw a 20% increase in the number of cyber security related threats compared to 2019.[1] Along with the increasing number of

threats, attackers are continuing to become more sophisticated and are able to evolve in the manner in which they attack. They utilise multiple attacking vectors to compromise systems.

Recent high profile cyber-attacks have further showed the importance of cyber security in society, with many smaller organizations now starting to employ external security assistance, as the skillset to ward off these threats is specialised in nature and may not be present in their organisation. The basis of any successful cyber security team is the Security Operations Centre (SOC).

The SOC team are responsible for monitoring, analysing, and mitigating potential threats. The key to a successful SOC team is their Security Information and Event Management system (SIEM), which acts as a centralised platform for all cyber security related notifications. The SIEM, gathers security notifications from various security related technologies such as firewalls, Intrusion Prevention Systems (IPS), Intrusion Detection Systems(IDS) and Anti-Virus logs. The effectiveness of a SOC team is dependent on the reliability and how effectively the SIEM platform filters its logs to generate alarms, this is where the key problem for SOC lies.

SOC analysts must ensure proper alarm rules are created. To trigger alerts for True Positive events, if alarms are incorrectly configured, they can lead to an overwhelming number of False Positive events which may cause a True Positive to be missed by the SOC team, which could have costly implications.

The performance of the SIEM can be improved using a variety of methods such as Threat Hunting, Threat Intelligence and Malware Identification. Implementation of such methods can reduce the number of false positive alarms.

In undertaking research for this paper, the objective is to further improve the efficiency of a SOC team by implementation of a machine learning algorithm.

## 1.3 Research Question

*"Using feature selection to improve intrusion detection"*

The purpose of this research paper is to provide a solution to the above statement by implementing a solution which utilises a multi-layer feature selection model. This research makes use of the KDD99 dataset, which has been widely used for intrusion detection traffic. The KDD99 dataset is prelabelled and an accompanying document can be found which identifies each label into the following categories: 'DOS', 'Probe', 'R2L', 'U2R', 'Other' and 'Normal'.

In order to answer this research question, the process was spilt into four key stages:

1. Data Preparation
2. Feature Selection
3. Classification
4. Visualisation

## 1.4 Summary of contents

This research paper is designed in the following order:

- Section 2 discusses previous works which are relevant and related to the multi-layer feature selection model which this paper puts forward.
- Section 3 outlines the methods employed during this research.

- Section 4 summarises the four key stages which this research process was created upon.
- Section 5 illustrates the feature selection and classification processes and their implementation.
- Section 6 highlights the results and evaluations for this research.
- Section 7 concludes and discusses the potential future workings.

# 2 Related Work

This piece of the research paper discusses previous works which put forward an approach of a similar nature to the multi-layer feature selection technique this paper presents. Much of the material discussed, relates to the KDD 99 dataset which is the same evaluation dataset this research paper is also basing its results on. Previous iterations which have proposed intrusion detection models and used different evaluation methods have been discussed also.

## 2.1 Supervised and Unsupervised learning

The goal of intrusion detection is to correctly identify abnormal network activity. As technology continuously advances, intrusion detection using different machine learning techniques have become the most popular approach [2]. Within machine learning there are two main approaches, supervised learning and unsupervised learning. Supervised learning models, learn from a number of training examples, where each example has a label which indicates its truth output [3]. Supervised learning can be further separated into two problems, classification and regression. Classification algorithms accurately assign the testing data into specific categories, such as filtering spam into a sperate folder in email. Regression algorithms understand the relationship between dependant and independent variables, these models are more useful for numerical data, such as sales projections.

Unsupervised learning uses unlabelled data and applies machine learning algorithms to analyse and cluster the data [4]. The three main problems unsupervised learning can be separated into are clustering, association and dimensionality. Clustering algorithms group unlabelled data based on their similarities and their differences. Association algorithms apply rules in order to find relationships between variables. Dimensionality algorithms are used when the number of features in a dataset is large, it can reduce the number of inputs while maintaining data integrity, this process is often used for data pre-processing. The previous research discussed during this literature review make use of both supervised and unsupervised learning models.

## 2.2 KDD99 Dataset approaches

In the 2020 HPBD&IS international conference, Jin et al. put forward KC-IDS, a multi-layer intrusion detection system [5] which is based upon the KDD 99 dataset. In this approach Jin et al. describe how the dataset is broken into ‘Normal’ and ‘Attack’, with ‘Attack’ being further split into ‘DOS’, ‘Probe’, ‘R2L’ and ‘U2R’. The paper puts forward the KC-IDS model, which acts as a multi-layer intrusion detection system by applying four layers of machine learning detection.

- Layer one makes use of KNN and is applied to the DOS traffic.
- Layer two makes use of KNN and is applied to the Probing traffic.
- Layer three makes use of Catboost and is applied to the U2R traffic.
- Layer four makes use of KNN and is applied to the R2L traffic.

For the future work of KC-IDS, Jin et al. suggest exploring the use of feature selection to further improve the multi-class detection, which is what this research explores.

In 2019, Gao et al. published their approach to the KDD 99 dataset, “An Adaptive Ensemble Machine Learning Model for Intrusion Detection” [6]. In this publication Gao et al. put forward an approach that utilises a MultiTree algorithm which is composed of several base classification models such as decision tree, random forest, KNN and DNN. In their findings it was noted that the quality of the data features was the key factor in determining the detection efficiency of the MultiTree. In their future recommendations they also highlighted idea for feature selection to further improve the efficiency.

In 2018, Nathan Shone proposed a novel approach to the KDD 99 dataset by means of a deep learning technique, which proposes nonsymmetric deep autoencoder (NDAE) for unsupervised learning. [7] Wu et al. published their novel intrusion detection model in 2018, in this publication they put forward an approach which uses CNN to solve the issue of an imbalance dataset, CNN selects the features from traffic in raw dataset automatically and then sets the cost function weight coefficient of each class based on its number [8]. XU et al. applied a deep learning theory and developed a model which uses a deep network which automatically applies feature extraction [9].

Farnaaz and Jabbar makes use of a random forest model on to detect the key types of attacks in the NSL-KDD dataset (DDOS, Probe, U2R and R2L), based on this study it was determined the model had a higher accuracy than a decision tree-based algorithm [10].

## 2.3 Other intrusion detection approaches

Other approaches to intrusion detection have been explored and their benchmarks established without the use of the KDD 99 dataset. Hsu et al. proposed an anomaly based network intrusion detection system which can be implemented for large scale networks [11]. Their approach threats and anomalies are detected based on their model which “consists of autoencoder (AE), support vector machine (SVM), and random forest (RF) models”. This model was then evaluated using UNSW-NB15.

Another approach which utilises an autoencoder was Aminanto et al. who proposed a solution with employs an isolation forest for unsupervised learning and adaptability combined with an autoencoder to reduce the number of false positive results. [12]

Although theses previous iterations have all proposed various models and approaches to intrusion detection, many of them have highlighted the concept of feature selection as a future work to further improve the detection rates of their proposed models. This is one of the key reasons why this research paper aims to an intrusion detection system which makes use of two-layer feature selection.

# 3 Research Methodology

For this research project the methodology which was follows was the Knowledge Discovery in Databases methodology (KDD). The KDD methodology comprises of four key stages, Selection, Pre-processing, Extraction and Evaluation. Figure 1 below shows the four stages of the KDD methodology. [13]

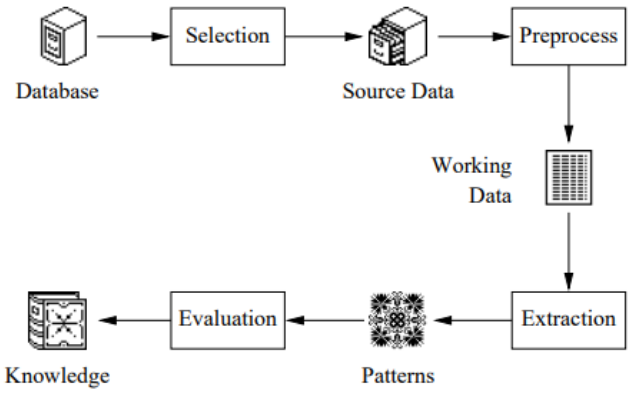


*Figure 1. Knowledge Discovery in Databases methodology (KDD) Stages* [13]

## 3.1 Stage One – Selection

The dataset chosen for this research project was the KDD99 dataset, although it is more than 20 years old the KDD99 dataset is still the most commonly used dataset for intrusion detection and machine learning within this research area. [14]

The KDD99 dataset was taken from the UCI KDD Archive, according to the UCI archive the full dataset contains around 18 million records, however due to hardware constraints the smaller version of the dataset which is available was chosen for use during this research project.

The dataset which was used for this project consisted of 494,022 rows and 41 columns of records. A 42$^{nd}$ column contains a label for the type of traffic that record is related to, i.e., Normal, Apache, Nmap, Back etc. An accompanying document provided by the UCI KDD Archive, indicates which of the attacking traffic types belongs to which of the following attacking categories:

- **Denial of Service (DOS):** DOS attacks are when an attacker overwhelms a computer resource or memory with traffic, which causes the resource to become too full to handle legitimate requests.
- **Probing (Probe):** Probing attacks are when an attacker attempts to gather information about a network in order to later attempt to avoid security controls.
- **User to Root (U2R):** U2R attacks occur when an attacker has already gained access to a normal user account by previous methods and is then able to perform an exploit which gains root access to the system. [15]
- **Remote to Local (R2L):** R2L attacks occur when an attacker who does not have access to an account on the network but is able to send data packets directly to a machine, performs an exploit which gains the attacker local access to that machine. [15]

All traffic which is not considered Normal or does not fall under the categories previously described, is considered Other traffic.

## 3.2 Stage two – Pre-process

The most important step when it comes to data is the preparation stage, as "the data pre-processing can often have a significant impact" on the performance of a machine learning algorithm [16]. In the data preparation stage, the KDD99 dataset is transformed into a format which can be better understood by machine learning.

For the purpose of this research project the first step of the data pre-processing was to split the dataset into subsets based on their categories as previously described. These categories are as follows:

- Normal Traffic
- DOS Traffic
- Probe Traffic
- U2R Traffic
- R2L Traffic
- Other Traffic

Once each of the traffic subsets were created, the next step was to remove the column which contained the traffic label, this label is only for human identification and as such serves no purpose for the machine learning algorithm later.

The next step of the data pre-processing was to make each of the subsets more readable for the machine learning algorithm. Each of the subsets contain string values which needed to be converted to numerical values in order for the machine learning algorithm to interpret them. The table below is an example of the conversions for the 'Protocol' variable

| Original String Value | Assigned Numerical Value |
|---|---|
| icmp | 1 |
| tcp | 2 |
| udp | 3 |

The full list of String to Numerical conversions is available in Appendix 9.1.

Finally, another column was manually added to each dataset called 'Attacking Or Non' with the Value 1 being assigned to all the attacking traffic and the value 0 being assigned to all Normal traffic. This was done for use by the machine learning algorithm later.

## 3.3 Stage Three – Extraction

Once the dataset was prepared for usage, the next stage was the extraction stage. In this stage the extraction is performed in two key steps, Feature Selection and Classification models.

In the feature selection stage, a multi-layer feature selection model is applied as follows:

**Layer One:**

1. Initial feature selection is applied to each individual subset.
2. The 10 key features of each subset are extracted.
3. Each subsets key features are combined, with duplicates being excluded.
4. These combined key features create the 'Initial Features'.

**Layer Two:**

1. Feature selection is applied to the 'Initial Features' created during layer one.

2. The ‘Final Features’ are created from the 10 key features extracted from ‘Initial Features ‘dataset.
3. These ‘Final Features’ are the key features in identifying any of the attacking categories.

The next stage of the extraction process was to use the Final Features extracted and apply this to the classification models chosen. The two chosen classification models were:

- Random Forest
- K-nearest neighbours (KNN)

The final features extracted were combined with the normal traffic, including the previously created column added to each dataset called ‘Attacking Or Non’ with the Value 1 being assigned to all the attacking traffic and the value 0 being assigned to all Normal traffic.

The two classification models were applied to this combined dataset and the results were analysed in the Evaluation stage next.

### 3.4 Stage Four – Evaluation

The evaluation stage is where the results from the classification models are analysed. The performance of each model was calculated with the outputs of the confusion matrix, the Accuracy, Sensitivity and Precision can be determined using the confusion matrix for each model.

A graph, confusion matrix and ROC curve, were created for a better understanding of the results of each classification model.

## 4 Design Specification

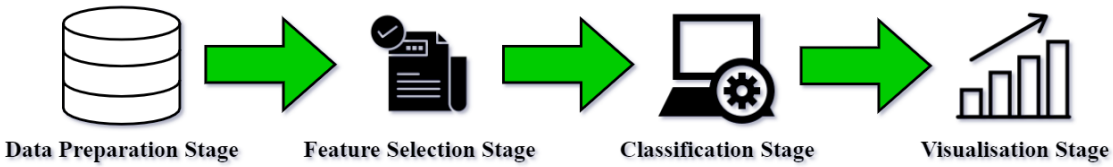


*Figure 2. Four key design stages*

The design of this research can be separated into four key stages such as can be seen in figure 2 above.

### 4.1 Data Preparation Stage

The most important step when it comes to data is the preparation stage, as “the data pre-processing can often have a significant impact” on the performance of a machine learning algorithm [16]. In the data preparation stage, the KDD99 dataset is transformed into a format which can be better understood by machine learning.

The dataset was to split into subsets based on their traffic category using the datasets accompanying documentation which highlights which traffic types falls under each category i.e., ‘DOS’, ‘Probe’, ‘R2L’, ‘U2R’, ‘Other’ or ‘Normal’. These datasets also contained string values which needed to be converted into numerical values in order for the machine learning algorithms to understand them.

Another column was manually added to each dataset called ‘Attacking Or Non’ with the Value 1 being assigned to all the attacking traffic and the value 0 being assigned to all Normal traffic. This was done for use by the machine learning algorithm later.

## 4.2 Feature selection Stage


*Figure 3. Multi-layer Feature Selection*

Figure 3 above, shows the multi-layer feature selection approach for this research. This multi-layer methodology utilises the benefits of the KDD99 dataset being prelabelled and identifiable under the categories: ‘DOS’, ‘Probe’, ‘R2L’, ‘U2R’, ‘Other’ and ‘Normal’. The ‘Normal’ traffic refers to all non-attacking traffic types, while ‘DOS’, ‘Probe’, ‘R2L’ and ‘U2R’ are attacking traffic types and ‘Other’ refers to any traffic which cannot be directly classified within any of those traffic types.

The steps of the multi-layer feature selection are as follows:

1. Initial feature selection is applied to each individual category.
2. The 10 key features of each category are extracted.
3. Each Categories key features are combined, with duplicates being excluded.
4. These combined key features create the ‘Initial Features’.
5. Feature selection is applied to the ‘Initial Features’.
6. The ‘Final Features’ is created from the 10 key features extracted from ‘Initial Features’.
7. These ‘Final Features’ are the key features in identifying any of the attacking categories.

The feature selection is conducted in two layers, the initial feature selection is applied to each traffic category to determine their top 10 key features. These 10 features from each category are combined to create the ‘Initial Features’, duplicate features are removed during this process. Next, feature selection is again applied, this time to the ‘Initial Features’, the outcome this process determined the ‘Final Features’, which are deemed to be the most prominent features of attacking traffic based on this dataset.

## 4.3 Classification Models Stage

The classification models chosen were Random Forest and K-nearest neighbours (KNN), random forests make use of "several randomized decision trees and aggregates their predictions by averaging" [17]. It has been highly adopted in recent years due to its ease and flexibility.

KNN is a simple but effective method for classification [18] that can be used for both classification and regression problems, however it is commonly used for classification problems, such as for this research project.

## 4.4 Visualisation Stage

The visualisation stage graphically represents the evaluation of the classification models in the form of a graph, confusion matrix and ROC curve, for a better understanding of the results of the classification model.

# 5 Implementation

Google Collaboratory was used for all coding throughout this research project. Google collab was used as it run in the cloud which makes the proposed model more agile. Since Google collab is run in the cloud it also makes the programmes accessible from any device, not just the local machine. Google collab also makes use Googles dedicated Graphics processing unit (GPU) and Tensor Processing Unit (TPU) which allows programmes to run faster.

Python coding language was used throughout this project also. Python was used for a variety of reasons such as the number of prebuilt libraries which could be made use of in order to create programmes more easily than other languages. In Python memory allocation is not an issue as it is done by default. Python is also considered a highly readable language which allows for easier reading and edits to be made where needed.

The implementation of this research project was divided into three key stages, Data Preparation Implementation, Feature selection Implementation and Classification model Implementation.

## 5.1 Data Preparation Implementation

As previously noted ,the most important step when it comes to data is the preparation stage, as "the data pre-processing can often have a significant impact" on the performance of a machine learning algorithm [16]. In the data preparation stage, the KDD99 dataset is transformed into a format which can be better understood by machine learning.

The first preparation step taken was to split the dataset into subsets based on their traffic category using the datasets accompanying documentation which highlights which traffic types falls under each category i.e., 'DOS', 'Probe', 'R2L', 'U2R', 'Other' or 'Normal'.

Once the subsets have been complied, the next step is to remove the column which contains the traffic label, these labels are only useful for human identification and as such serves no purpose for the machine learning algorithm.

Finally, in order for the machine learning algorithm to understand the data, string values must be converted into a numerical value instead. In the KDD99 dataset, there are

three variables which are made up of string values, ‘Protocol’, ‘Service and ‘Flag’. As there as so many data entries which all contain these values, a python script was created in order to transform these strings into numerical values. These numerical values were assigned incrementally for each variable, for instance:

‘Protocol’ variable:

| Original String Value | Assigned Numerical Value |
|---|---|
| icmp | 1 |
| tcp | 2 |
| udp | 3 |

This process of converting the string values to numerical values was applied to the remaining ‘Service’ and ‘Flag’ variables and then applied across each attacking traffic category (‘DOS’, ‘Probe’, ‘R2L’, ‘U2R’and ‘Other’) which can be seen in figure 4 below, which is a code snippet for the replacing of string values to numerical values for the DOS dataset.

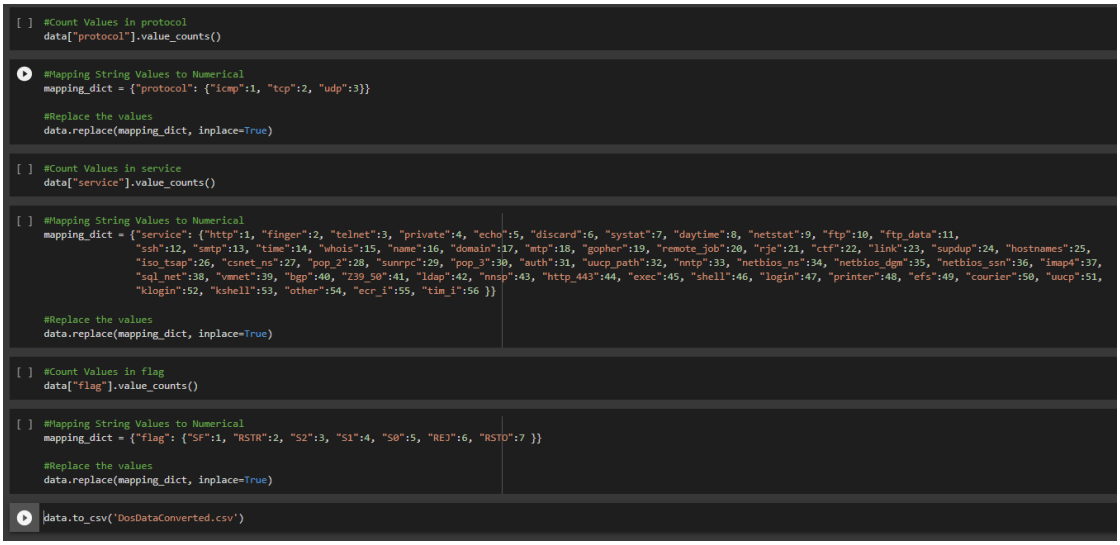


*Figure 4. Converting Strings to Numerical value code snippet*

## 5.2 Feature selection Implementation

Chi-2 was used for feature selection during this research, it is a simple and general algorithm that can automatically select features according to the characteristics of the data. [19]

Feature Selection was applied to each of the sub datasets created during the data preparation phase initially in order to find the top 10 features of each sub dataset. The figure 5 below shows the code snippet for how feature selection was performed on the DOS dataset.

```
[ ] #Import Dependencies
    import pandas as pd
    import numpy as np

[ ] #Import Feature libraries
    from sklearn.feature_selection import SelectKBest
    from sklearn.feature_selection import chi2

[ ] #Read CSV File
    data = pd.read_csv("/content/drive/MyDrive/Masters/Thesis/Dataset/DosDataConverted.csv")

    #Create Dataframe
    X = data.iloc[:,0:40]
    y = data.iloc[:,0]

[ ] #Select 10 features with highest Chi2 Value
    bestfeatures = SelectKBest(score_func=chi2, k=10)
    fit = bestfeatures.fit(X,y)

[ ] dfscores = pd.DataFrame(fit.scores_)
    dfcolumns = pd.DataFrame(X.columns)

[ ] #Add Features to table
    featureScores = pd.concat([dfcolumns,dfscores],axis=1)
    featureScores.columns = ['Heading','Chi2 Score']

[ ] #Print All Feature Scores
    featureScores

[ ] #Print Top 10 Feature Scores
    print(featureScores.nlargest(10,'Chi2 Score'))
```

*Figure 5. Feature Selection Code Snippet*

Following the feature selection of the DOS dataset, the same process was applied to the remaining datasets in order to identify the key features of each. The figures below show the outputs of each feature selection process.

```
[11] #Print Top 10 Feature Scores
     print(featureScores.nlargest(10,'Chi2 Score'))

                 Heading    Chi2 Score
4             src_bytes  2.585959e+07
5             dst_bytes  1.260810e+07
0              duration  1.280612e+06
22                count  5.088164e+03
23            srv_count  4.746115e+03
11            logged_in  2.819402e+03
9                   hot  2.261863e+03
2               service  5.609249e+02
12    num_compromised    4.537381e+02
32  dst_host_srv_count   4.510814e+02
```

*Figure 6. Dos data top features*

```
[10] #Print Top 10 Feature Scores
     print(featureScores.nlargest(10,'Chi2 Score'))

                 Heading    Chi2 Score
0              duration  17236.086957
11            logged_in    1582.000703
22                count     994.816354
32  dst_host_srv_count       64.530951
31      dst_host_count       39.536631
2               service      38.945066
3                  flag      12.117974
4             src_bytes      10.069439
23            srv_count        3.165628
27     srv_rerror_rate         3.130270
```

*Figure 7. Probe data top features*

```
[10] #Print Top 10 Feature Scores
     print(featureScores.nlargest(10,'Chi2 Score'))

                 Heading    Chi2 Score
4             src_bytes  4.376901e+06
5             dst_bytes  3.864745e+06
0              duration  4.082377e+04
15             num_root  3.270289e+03
32  dst_host_srv_count   2.033403e+03
31      dst_host_count   1.758611e+03
12    num_compromised    1.224656e+03
9                   hot  1.241875e+02
8                urgent  1.140000e+02
16  num_file_creations   1.121735e+02
```

*Figure 8. U2R data top features*

```
[10] #Print Top 10 Feature Scores
     print(featureScores.nlargest(10,'Chi2 Score'))

                 Heading    Chi2 Score
4             src_bytes  1.523884e+09
5             dst_bytes  4.241469e+08
0              duration  1.336579e+06
32  dst_host_srv_count   2.483504e+05
31      dst_host_count   1.559174e+05
15             num_root  1.295672e+05
12    num_compromised    7.225153e+04
2               service  2.889722e+04
16  num_file_creations   2.042467e+04
18    num_access_files   1.697495e+04
```

*Figure 9. R2L data top features*

```
[30] #Print Top 10 Feature Scores
     print(featureScores.nlargest(10,'Chi2 Score'))

                    Heading    Chi2 Score
4                 src_bytes  6.560093e+11
5                 dst_bytes  1.580534e+09
0                  duration  1.345022e+08
15                 num_root  2.067216e+06
12        num_compromised    1.979361e+06
23              srv_count    1.639019e+05
22                  count    1.486894e+05
32     dst_host_srv_count    5.959979e+04
9                      hot   3.821989e+04
10      num_failed_logins    3.769023e+04
```

*Figure 10. Other data top features*

Once these top features for each dataset were calculated they were combined, and duplicating features were removed to create the top 18 unique initial features as follows:

| Feature Number | Feature Name |
|---|---|
| 1 | count |
| 2 | dst_bytes |
| 3 | dst_host_count |
| 4 | dst_host_srv_count |
| 5 | duration |
| 6 | flag |
| 7 | hot |
| 8 | logged_in |
| 9 | num_access_files |
| 10 | num_compromised |
| 11 | num_failed_logins |
| 12 | num_file_creations |
| 13 | num_root |
| 14 | service |
| 15 | src_bytes |
| 16 | srv_count |
| 17 | srv_rerror_rate |
| 18 | urgent |

The attacking datasets which these features were extracted from (‘DOS’, ‘Probe’, ‘R2L’, ‘U2R’ and ‘Other’) were then combined and any features which were not part of the list above were removed, this created the Initial Features attacking dataset.

Feature selection was then performed on the Initial Features attacking dataset that was created, to find the Final top 10 attacking features, these features can be seen in figure 11 below.

```
[10] #Print Top 10 Feature Scores for Final Feature Selection
     print(featureScores.nlargest(10,'Chi2 Score'))

                   Heading    Chi2 Score
3              src_bytes  2.533436e+12
4              dst_bytes  7.841052e+09
0               duration  1.737232e+09
10              num_root  1.671366e+07
9       num_compromised  5.649862e+06
13                count  3.772507e+06
14            srv_count  3.286356e+06
11  num_file_creations  7.417003e+05
12    num_access_files  3.217854e+05
17  dst_host_srv_count  2.789314e+05
```

*Figure 11. Final top features of attacking data*

These final features are found to be the top 10 features of attacking data within the KD99 dataset.

| Feature Number | Feature Name |
|---|---|
| 1 | count |
| 2 | dst_bytes |
| 3 | dst_host_srv_count |
| 4 | duration |
| 5 | num_access_files |
| 6 | num_compromised |
| 7 | num_file_creations |
| 8 | num_root |
| 9 | src_bytes |
| 10 | srv_count |

Once these features were chosen as the 10 key attacking features, the normal traffic was then combined with the attacking data using these features to create the Combined dataset which would train the proposed model.

## 5.3 Classification model Implementation

### 5.3.1 Random Forest

The first classification model used during this research was the Random Forest model, figure below shows the code snippet used to compute the Random Forest model. In this study, the random forest model was optimized using the hyperparameters like n_estimator and random_state .The below figure 12 shows the random forest model fine-tuned with n_estimator as 100 and random_state as 0.

```
#Import Dependencies and functions
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn import metrics
import seaborn as sn
from sklearn.metrics import confusion_matrix

[30] #Read CSV File
dataset = pd.read_csv("/content/drive/MyDrive/Masters/Thesis/Dataset/FeatureSelectionCombined.csv")

[31] #Create dataframe
X = dataset.iloc[:,0:11]
y = dataset.iloc[:,0]

[32] #Split Features and Labels
X=dataset[['duration', 'src_bytes', 'dst_bytes', 'num_compromised', 'num_root', 'num_file_creations', 'num_access_files', 'count', 'srv_count', 'dst_host_srv_count' ]] #Split Features
y=dataset['attack_or_non'] #Split Labels

[33] #Split dataset into training set and test set
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)

[34] #Perfrom Random Forest
rfc = RandomForestClassifier(n_estimators=100, random_state=0)

[35] rfc.fit(X_train,y_train)

[36] y_pred = rfc.predict(X_test)

[37] #Display Accuracy
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

*Figure 12. Random Forest Code Snippet*

### 5.3.2 K-nearest neighbours (KNN)

The second classification model used was K-nearest neighbours (KNN) which is a simple but effective method that is commonly used for classification problems, such as for this research project. The number of n_neighbors chosen to run for this algorithm was 10.The default number of n_neighbors is 5, so double was chosen for this research project.Figure 13 below shows the code snippet of the KNN implementation.

```
[2] #Read CSV File
dataset = pd.read_csv("/content/drive/MyDrive/Masters/Thesis/Dataset/FeatureSelectionCombined.csv")

#Create dataframe
X = dataset.iloc[:,0:11]
y = dataset.iloc[:,0:311030]

[4] #Split Features and Labels
X=dataset[['duration', 'src_bytes', 'dst_bytes', 'num_compromised', 'num_root', 'num_file_creations', 'num_access_files', 'count', 'srv_count', 'dst_host_srv_count' ]] #Split Features
y=dataset['attack_or_non'] #Split Labels

[5] #Split dataset into training set and test set
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)

[6] #Perfrom KNN
classifier = KNeighborsClassifier(n_neighbors = 10)

[7] classifier.fit(X_train, y_train)

[8] y_pred = classifier.predict(X_test)
```

*Figure 13. KNN Code Snippet*

# 6 Evaluation

In this section the accuracy and performance of the classification models are evaluated. The performance of each model was calculated with the outputs of the confusion matrix, the Accuracy, Sensitivity and Precision can be determined using the confusion matrix for each model.

Accuracy, Sensitivity and Precision are defined and can be calculated as follows:

**Accuracy**
Accuracy can be defined as the number of times the model correctly classified all normal traffic and all attacking traffic.

$$Accuracy = \left(\frac{TP+TN}{TP+TN+FP+FN}\right)$$

*Figure 14. Accuracy Formula* [20]

**Precision**
Precision a ratio based on the number of True Positive predictions by the total Positive prediction.

$$Precision = \left(\frac{TP}{TP+FP}\right)$$

*Figure 15. Precision Formula* [20]

**Sensitivity**
Sensitivity also known as Recall or the True Positive rate, is a ratio based on the number of True Positive predictions by the Total Number of positives.

$$Recall = \left(\frac{TP}{TP+FN}\right)$$

*Figure 16. Recall Formula* [20]

## 6.1 Experiment for Random Forest Classifier

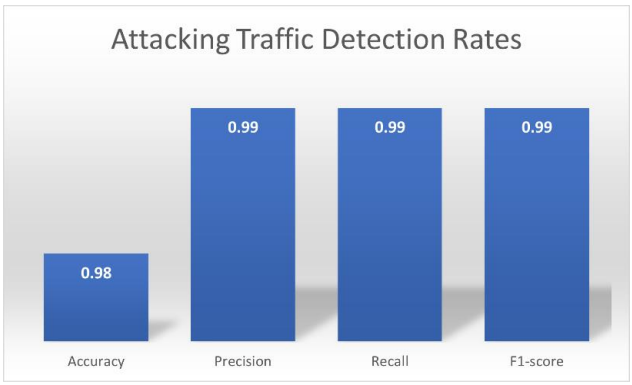


*Figure 17. Random Forest Detection Graph*

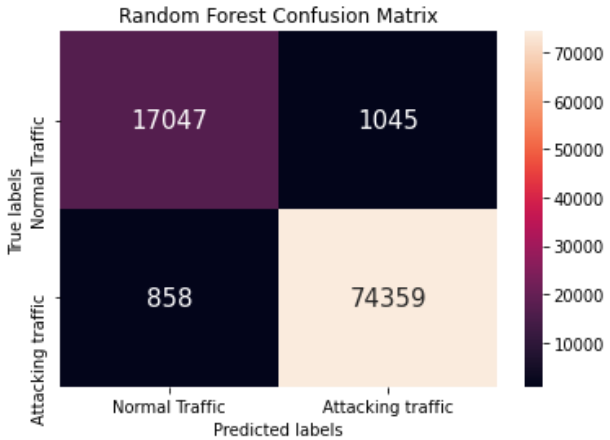


*Figure 18. Random Forest Confusion Matrix*

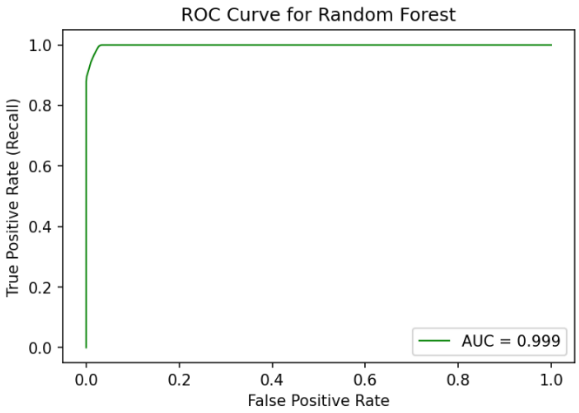


*Figure 19. Random Forest ROC Curve*

## 6.2 Experiment for KNN

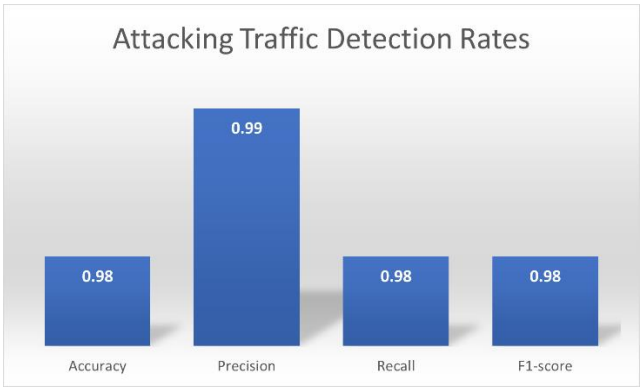


*Figure 20. KNN Detection Graph*

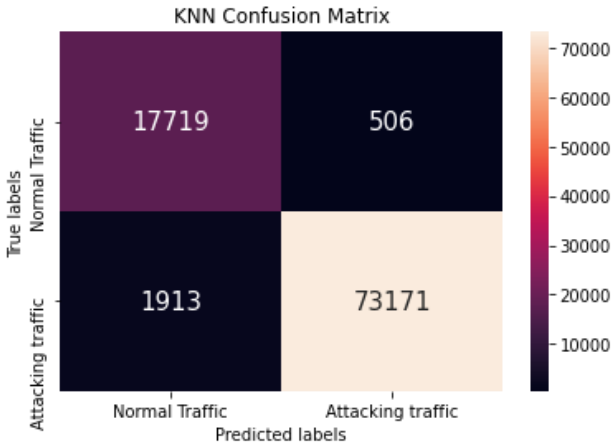


*Figure 21. KNN Confusion Matrix*

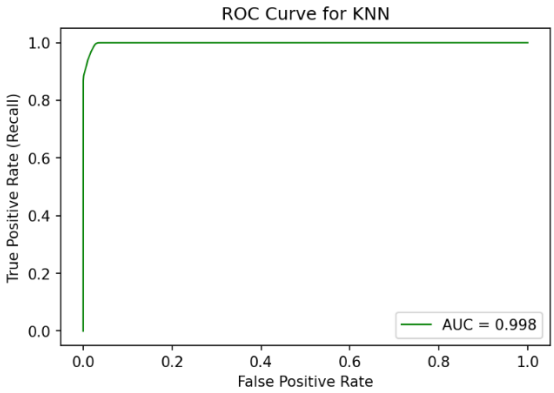


*Figure 22. KNN ROC Curve*

## 6.3 Discussion

Figure 23 below shows a comparison evaluation from the results of the Random forest algorithm and the K-nearest neighbours algorithm.

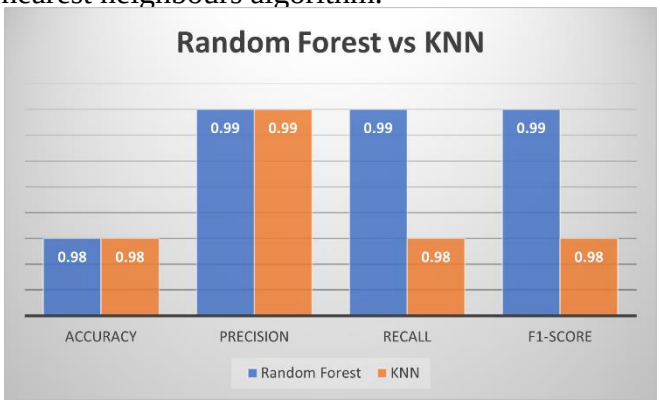


*Figure 23. Random Forest vs KNN graph*

As can be seen in figure 23, both algorithms scored 98% for their accuracy and 99% for their precision. However, when it comes to the recall and F1-score, the Random Forest algorithm outperformed the K-nearest neighbours algorithm with 99% versus 98% in both categories.

Using the Random Forest results to compare to previous approaches as it out preformed KNN overall. Feng et al. proposed a User-Centric Machine Learning Framework [21], when Random forest was applied during the approach the model achieved a detection rate of 80%, which is significantly less than the 99% achieved during this research project which can be seen in figure 19 based on the Random Forest ROC Curve.

Jin et al. put forward KC-IDS, a multi-layer intrusion detection system [5] which applies machine learning algorithms to each attacking category separately achieving the following results for each:

| **Traffic Type** | **Detection Rate** | **False Positive Rate** | **F-Measure** |
|---|---|---|---|
| DOS | 99.94 | 0.60 | 99.92 |
| Probe | 94.14 | 0.16 | 96.19 |
| U2R | 50.00 | 0.00 | 66.67 |
| R2L | 55.23 | 0.11 | 64.83 |

While the Detection Rate and F-Measure for Dos traffic archives a significate score for KC-IDS, the remaining traffic types achieve significantly lower scores than the scores achieved by the model conducted in this research project.

Furthermore KC-IDS preforms classification models for each category separately which increases the complexity and number of resources needed to conduct, the model proposed during this research utilises the classification model on the combined traffic while maintaining impressive scores for accuracy, precision, recall and F1-score, thus reducing the number of resources required.

# 7 Conclusion and Future Work

## 7.1 Conclusion

For this research project the impact of feature selection in combination with classification models for intrusion detection was assessed. A multi-layer feature selection model was introduced to the KDD99 dataset in order to identify the top key attacking traffic features. The model utilises Chi2 for its feature selection process. The first layer of the model preforms feature selection on each subset in order to find the top 10 attacking features for each individual attacking type within the KDD99 dataset. The second layer of the model combines the previously found features from each subset, once combined the duplicate features are removed and then feature selection is applied to this dataset to determine the final ten key features of attacking traffic.

The feature selection model was then applied to two classification models, the Random Forest algorithm and the K-nearest neighbours algorithm. These classification models were evaluated using a confusion matrix and ROC graph in order to better analyse the results. The results determined that the Random Forest out preformed KNN in recall and f1-score categories with 99% versus 98% in each. However, in the accuracy and precision categories, Random Forest and KNN scored equally with 99% and 98% respectively.

## 7.2 Limitations

Due to time and resource constraints, it was not possible to generate a unique intrusion dataset by simulation or other methods, which is why the commonly used KDD99 dataset was chosen for this research project. Also due to resource constraints the full 18 million record KDD99 dataset was no used during this project, the small version provided was chosen.

## 7.3 Future Work

For the future work of this research, live Security Information and Event Management system (SIEM) traffic could be used and evaluated rather than the KDD99 dataset. Upon successfully utilising real world traffic, should the results continue to have a high accuracy, then further work would be to implement the model into a live SIEM system for usage.

# 8 Acknowledgment

I would like to thank my supervisor, Vikas Sahni, for his constant support and guidance. Throughout the entire process Vikas Sahni continuously provided encouragement and assistance whenever there may have been any queries or concerns on my behalf. I must also thank my family for their constant support throughout this master's programme. Finally, I would like to also thank my friends which have supported me during this research and provided me with feedback and advice from their own computing knowledge backgrounds.

# Bibliography


[1] Keumars Afifi-Sabet, '2020 the busiest year on record for cyber attacks against UK firms', *IT PRO*, Jan. 11, 2021. https://www.itpro.co.uk/security/cyber-attacks/358276/2020-the-busiest-year-on-record-for-cyber-attacks-against-uk-firms (accessed Jun. 16, 2021).

[2] R. U. Khan, X. Zhang, M. Alazab, and R. Kumar, 'An Improved Convolutional Neural Network Model for Intrusion Detection in Networks', in *2019 Cybersecurity and Cyberforensics Conference (CCC)*, May 2019, pp. 74–77. doi: 10.1109/CCC.2019.000-6.

[3] Z.-H. Zhou, 'A brief introduction to weakly supervised learning', *National Science Review*, vol. 5, no. 1, pp. 44–53, Jan. 2018, doi: 10.1093/nsr/nwx106.

[4] K. Hsu, S. Levine, and C. Finn, 'Unsupervised Learning via Meta-Learning', *arXiv:1810.02334 [cs, stat]*, Mar. 2019, Accessed: Jul. 27, 2021. [Online]. Available: http://arxiv.org/abs/1810.02334

[5] D. Jin, Y. Lu, J. Qin, Z. Cheng, and Z. Mao, 'KC-IDS : Multi-layer Intrusion Detection System', in *2020 International Conference on High Performance Big Data and Intelligent Systems (HPBD IS)*, May 2020, pp. 1–5. doi: 10.1109/HPBDIS49115.2020.9130573.

[6] X. Gao, C. Shan, C. Hu, Z. Niu, and Z. Liu, 'An Adaptive Ensemble Machine Learning Model for Intrusion Detection', *IEEE Access*, vol. 7, pp. 82512–82521, 2019, doi: 10.1109/ACCESS.2019.2923640.

[7] N. Shone, T. N. Ngoc, V. D. Phai, and Q. Shi, 'A Deep Learning Approach to Network Intrusion Detection', *IEEE Trans. Emerg. Top. Comput. Intell.*, vol. 2, no. 1, pp. 41–50, Feb. 2018, doi: 10.1109/TETCI.2017.2772792.

[8] K. Wu, Z. Chen, and W. Li, 'A Novel Intrusion Detection Model for a Massive Network Using Convolutional Neural Networks', *IEEE Access*, vol. 6, pp. 50850–50859, 2018, doi: 10.1109/ACCESS.2018.2868993.

[9] C. Xu, J. Shen, X. Du, and F. Zhang, 'An Intrusion Detection System Using a Deep Neural Network With Gated Recurrent Units', *IEEE Access*, vol. 6, pp. 48697–48707, 2018, doi: 10.1109/ACCESS.2018.2867564.

[10] N. Farnaaz and M. A. Jabbar, 'Random Forest Modeling for Network Intrusion Detection System', *Procedia Computer Science*, vol. 89, pp. 213–217, Jan. 2016, doi: 10.1016/j.procs.2016.06.047.

[11] Y.-F. Hsu, Z. He, Y. Tarutani, and M. Matsuoka, 'Toward an Online Network Intrusion Detection System Based on Ensemble Learning', in *2019 IEEE 12th International Conference on Cloud Computing (CLOUD)*, Jul. 2019, pp. 174–178. doi: 10.1109/CLOUD.2019.00037.

[12] M. E. Aminanto, T. Ban, R. Isawa, T. Takahashi, and D. Inoue, 'Threat Alert Prioritization Using Isolation Forest and Stacked Auto Encoder With Day-Forward-Chaining Analysis', *IEEE Access*, vol. 8, pp. 217977–217986, 2020, doi: 10.1109/ACCESS.2020.3041837.

[13] G. J. Williams and Z. Huang, 'A Case Study in Knowledge Acquisition for Insurance Risk Assessment using a KDD Methodology', in *Kdd Methodology, Data Mining Portfolio - Tr Dm 96023, Csiro*, 1996, pp. 117–129.

[14] A. Özgür and H. Erdem, 'A review of KDD99 dataset usage in intrusion detection and machine learning between 2010 and 2015', PeerJ Preprints, preprint, Apr. 2016. doi: 10.7287/peerj.preprints.1954v1.

[15] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, 'A detailed analysis of the KDD CUP 99 data set', in *2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications*, Jul. 2009, pp. 1–6. doi: 10.1109/CISDA.2009.5356528.

[16] S. B. Kotsiantis, D. Kanellopoulos, and P. E. Pintelas, ‘Data Preprocessing for Supervised Leaning’, vol. 1, no. 1, p. 8, 2006.
[17] G. Biau and E. Scornet, ‘A random forest guided tour’, *TEST*, vol. 25, no. 2, pp. 197–227, Jun. 2016, doi: 10.1007/s11749-016-0481-7.
[18] G. Guo, H. Wang, D. Bell, Y. Bi, and K. Greer, ‘KNN Model-Based Approach in Classification’, in *On The Move to Meaningful Internet Systems 2003: CoopIS, DOA, and ODBASE*, Berlin, Heidelberg, 2003, pp. 986–996. doi: 10.1007/978-3-540-39964-3_62.
[19] H. Liu and R. Setiono, ‘Chi2: feature selection and discretization of numeric attributes’, in *Proceedings of 7th IEEE International Conference on Tools with Artificial Intelligence*, Nov. 1995, pp. 388–391. doi: 10.1109/TAI.1995.479783.
[20] F. Rahmad, Y. Suryanto, and K. Ramli, ‘Performance Comparison of Anti-Spam Technology Using Confusion Matrix Classification’, *IOP Conf. Ser.: Mater. Sci. Eng.*, vol. 879, p. 012076, Aug. 2020, doi: 10.1088/1757-899X/879/1/012076.
[21] C. Feng, S. Wu, and N. Liu, ‘A user-centric machine learning framework for cyber security operations center’, in *2017 IEEE International Conference on Intelligence and Security Informatics (ISI)*, Jul. 2017, pp. 173–175. doi: 10.1109/ISI.2017.8004902.

# 9 Appendix

## 9.1 Full List of String to Numerical Conversions

**‘Flag’** Conversions:

| **Flag** | **Conversion** |
|---|---|
| SF | 1 |
| RSTR | 2 |
| S2 | 3 |
| S1 | 4 |
| S0 | 5 |
| REJ | 6 |
| RSTO | 7 |
| S3 | 8 |
| SH | 9 |
| OTH | 10 |
| RSTOS0 | 11 |

**‘Service’** Conversions:

| **Service** | **Conversion** |
|---|---|
| http | 1 |
| finger | 2 |
| telnet | 3 |
| private | 4 |
| echo | 5 |
| discard | 6 |
| systat | 7 |
| daytime | 8 |

| | |
|---|---|
| netstat | 9 |
| ftp | 10 |
| ftp_data | 11 |
| ssh | 12 |
| smtp | 13 |
| time | 14 |
| whois | 15 |
| name | 16 |
| domain | 17 |
| mtp | 18 |
| gopher | 19 |
| remote_job | 20 |
| rje | 21 |
| ctf | 22 |
| link | 23 |
| supdup | 24 |
| hostnames | 25 |
| iso_tsap | 26 |
| csnet_ns | 27 |
| pop_2 | 28 |
| sunrpc | 29 |
| pop_3 | 30 |
| auth | 31 |
| uucp_path | 32 |
| nntp | 33 |
| netbios_ns | 34 |
| netbios_dgm | 35 |
| netbios_ssn | 36 |
| imap4 | 37 |
| sql_net | 38 |
| vmnet | 39 |
| bgp | 40 |
| Z39_50 | 41 |
| ldap | 42 |
| nnsp | 43 |
| http_443 | 44 |
| exec | 45 |
| shell | 46 |
| login | 47 |
| printer | 48 |
| efs | 49 |
| courier | 50 |
| uucp | 51 |
| klogin | 52 |
| kshell | 53 |
| other | 54 |
| ecr_i | 55 |

| | |
|---|---|
| tim_i | 56 |
| urp_i | 57 |
| eco_i | 58 |
| domain_u | 59 |
| IRC | 60 |
| pm_dump | 61 |
| X11 | 62 |
| ntp_u | 63 |
| icmp | 64 |
| tftp_u | 65 |

**‘Protocol’** Conversions:

| **Protocol** | **Conversion** |
|---|---|
| icmp | 1 |
| tcp | 2 |
| udp | 3 |