A robust domain partitioning intrusion detection method

MWITONDI, Kassim <http://orcid.org/0000-0003-1134-547X>, SAID, Raed A and ZARGARI, Shahrzad <http://orcid.org/0000-0001-6511-7646>

A Robust Domain Partitioning Intrusion Detection Method

Kassim S. Mwitondi[1], Raed A. Said[2], and Shahrzad A. Zargari[1]

[1]Sheffield Hallam University, Faculty of Science, Technology and Arts
[2]Canadian University Dubai

Abstract

The capacity for data mining algorithms to learn rules from data is influenced by, *inter-alia*, the random nature of training and test data as well as by the diversity of domain partitioning models. Isolating normal from malicious data traffic across networks is one regular task that is naturally affected by that randomness and diversity. We propose a robust algorithm **Sample-Measure-Assess (SMA)** that detects intrusion based on rules learnt from multiple samples. We adapt data obtained from a set of simulations, capturing data attributes identifiable by number of bytes, destination and source of packets, protocol and nature of data flows (normal and abnormal) as well IP addresses. A fixed sample of 82,332 observations on 27 variables was drawn from a superset of 2.54 million observations on 49 variables and multiple samples were then repeatedly extracted from the former and used to train and test multiple versions of classifiers, via the algorithm. With two class labels–binary and multi-class, the dataset presents a classic example of *masked* and *spurious* groupings, making an ideal case for concept learning. The algorithm learns a model for the underlying distributions of the samples and it provides mechanics for model assessment. The settings account for our method's novelty–i.e., ability to learn concept rules from highly masked to highly spurious cases while observing model robustness. A comparative analysis of Random Forests and individually grown trees show that we can circumvent the former's dependence on multicollinearity of the trees and their individual strength in the forest by proceeding from dimensional reduction to classification using individual trees. Given data of similar structure, the algorithm can order the models in terms of optimality which, means our work can contribute towards understanding the concept of *normal* and *malicious* flows across tools. The algorithm yields results that are less sensitive to violated distributional assumptions and, hence, it yields robust parameters and provides a generalisation that can be monitored and adapted to specific low levels of variability. We discuss its potential for deployment with other classifiers and potential for extension into other applications, simply by adapting the objectives to specific conditions.

Key Words: *Bagging, Bootstrapping, Classification, Cross-Validation, Cyber-Security, Data Mining, Decision Trees, Intrusion Detection, Over-fitting, Random Forests, Robustness, Supervised Modelling, Unsupervised Modelling*

1 Introduction

Modern communication networks are characterised by large volumes of data–highly dynamical, volatile and variable which, inevitably, complicates systems security. Successfully isolating normal from malicious traffic across networks is an integral function of predictive modelling, a fundamental objective of which is to attain accurate and reliable results. Striking a balance between the two attributes remains an interesting subject of research. While using the Maximum Likelihood Estimators (MLEs) to obtain distributional parameters from well-behaved data may be straightforward, problems arise when underlying distributional assumptions are violated. Under such circumstances the general practice is to deploy algorithms that learn rules from training data and apply them to new data. One common approach is to deploy **robust** methods–a general description of a set of statistical tools that are less sensitive to violation of distributional assumptions as described in Kent and Tyler [1], Mwitondi [2] and Mwitondi et al. [3]. The methods work well for mixtures of two normal distributions with different standard deviations; under this setting, parametric methods like the ***t***-test work poorly in separating the two densities. Various data mining methods for learning rules from data have been developed in recent years, examples include Wu and Banzhaf [4], Sommer and Paxson [5] and Mitchell and Chen [6]. The overall performance of many of these methods is constrained by variability in the data and model parameters, with **performance** used in its model fitting context to refer to metrics relating to accuracy and optimality, whereas domain partitioning refers to a decomposition of multiple training and test samples to be processed

by several classifiers. We propose a novel method based on a robust algorithm–**Sample-Measure-Assess (SMA)** that detects intrusion based on rules learnt from multiple samples. The approach considers the nature of data generation and domain partitioning as the converging point between data randomness and domain-knowledge.

1.1 Motivation, Problem Definition and Objectives

Learning rules from data relies heavily on prior knowledge, both in the form of historical data and expert knowledge, making attribute matching and adding features (see Section 2.1) a crucial part of the process. Such a process is susceptible to variability, its accuracy and reliability both depend on the multiplicity of data sharing and model evaluation, an area our paper seeks to contribute to. We set off from the overall Bayesian approach to classification, as discussed in Dunsmore [7], to set the scene for using prior knowledge to generate posterior information (Section 2.2.2). Thus, we define the problem as: **Using testbed configuration generated data to model variations due to data randomness**. To address the foregoing problem, we set the following objectives.

1. **To train and test classifiers that conform to data variability.**
2. **To carry out predictive modelling of intrusion using the UNSW-NB15 and ACCS [8] dataset.**
3. **To carry out a performance comparative analysis of the classifiers.**
4. **To highlight the potential role of interdisciplinarity in intrusion detection.**

The foregoing objectives are fairly standard and, conventionally, they have been pursued based on pre-defined ontologies with inherently highly dynamic parameters. Such an approach tends to randomise, not only the training and testing datasets, but also, as reported in Mwitondi and Said [9], the predictive power of classifiers. Standard solutions, in many applications, focus on competing models, voting and cross-validation–implying that results are conditional on the classifiers' internal mechanics. For instance, the Random Forests (Breiman [10]) error rate, typically, depends both on the multicollinearity of the trees and the strength of individual trees in the forest. In this case, subjecting the data to dimensional reduction potentially reduces both multicollinearity and tree strength, making the data dimension a crucial adjustable parameter. This paper exploits the well-documented natural, sequential relationship between unsupervised and supervised modelling as described in Chapmann [11] and Ashfaq et al. [12]. Reducing unsupervised to supervised learning is a standard practice, but as exhibited by Garg and Kalai [13], there is a need for a paradigm for mitigating subjectivity in unsupervised decision-making via leveraging prior knowledge. Their meta-clustering approach chooses the unsupervised algorithm with lowest empirical error on training set and, although they apply the method on high-dimensional data, limiting the applications to two clusters has a potentially masking effect. A more informative approach would be to explore as many natural groupings as possible, from the smallest to the largest number of clusters (Section 2). This step can then be followed by supervised modelling, allocating new cases to established classes. As well as data sources, another source of the randomness in Table 1 are model settings and these are the issues **SMA** seeks to address. As explained below, our approach provides a non-standard sequence of these methods.

1.2 Main Contribution

While the two foregoing sequence of activities may sound standard, they are based on novel mechanics. More specifically, the papers novelty is embedded in Sections 2.2.1 and 2.2.3. In particular, **Algorithm 1** explores as many naturally emerging groupings as possible and the results are used as inputs in training learning models. The activities between the two sections provide a generalisation that can be monitored and adapted to specific low levels of variability, as shown in lines 19 and 20 of the algorithm. These outcomes are less sensitive to violated assumptions for data drawn from various sources with disparate distributions, and as such they yield robust parameters, as described in Kent and Tyler [1]. This generalisation is particularly emphasised by the first objective, while the remaining three objectives not only provide an application basis for an authoritative dataset in intrusion detection, but they also provide scope for adapting the method to other applications by simply replacing objective 2.

The use of binary and multi-class labels, present a classic example of tackling masked and spurious clusters. The former arises when the model fails to isolate distinctive clusters in such as way that potentially heterogeneous clusters overlap while the latter is when the model adapts itself so well to the data, that even random variations are incorporated

into cluster formation. Both cases, also known as under-fitting and over-fitting respectively, are undesirable and, the two class labels help in determining optimality. That is, combining the power of automated learning techniques (via objectives 1, 2 and 3) and existing domain knowledge (via objectives 2, 3 and 4) to uncover networks intrusion empowers the method to learn concept rules from highly masked to highly spurious cases while observing model robustness. No existing work provides such a robust generalisations on variations due to data randomness, a well-documented challenge in data science as reported Bridges et al. [14], Mwitondi et al. [3] and Mwitondi and Said [9]. Finally, from an intrusion detection perspective, normal and malicious flows do not fit in any current concept definition across tools. This work is expected to enhance that understanding from a modelling point of view, as described below.

2 Methods

This section sets the scene for achieving the objectives in Section 1. We use testbed configuration generated data from UNSW-NB15 and ACCS [8] to explore variations for the purpose of addressing randomness in allocation rules in intrusion detection. The dataset, fully described in Section 2.1, consists of 82,332 observations on 27 variables, drawn from a large set of 2.54 million observations on 49 variables with binary and multi-class labels. Extracted samples are used to train and test multiple versions of classifiers, proceeding from dimensional reduction to classification via a specially-designed algorithm. The data and implementation strategy are described below.

2.1 Data Sources

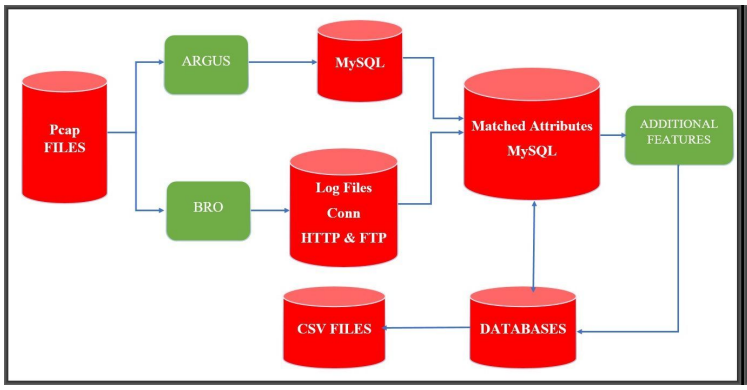
Data were obtained from thousands of raw network packets of the UNSW-NB 15 created by the IXIA PerfectStorm tool in the Cyber Range Lab of the Australian Centre for Cyber Security (ACCS). The data generation process was accomplished using twelve algorithms as described in UNSW-NB15 and ACCS [8] and in Moustafa and Slay [15, 16]. The process ran through a testbed configuration generating over 2.54 million records of *normal* and *malicious* flows through the IXIA traffic generator involving three servers–two *normal* and one *malicious*, passing the flow through a firewall configured to pass both types of traffic onto the Internet and a **tcpdump**. The latter, a packet analyzer allowing visibility of TCP/IP and other transmitted packets being transmitted or received over the testbed configuration, allows the process to visually validate the packets from IXIA. The process is followed by feature creation, graphically illustrated on the left hand side in Figure 1, matching attributes via a database and adding features before generating CSV files. The datasets are avaliable at UNSW-NB15 and ACCS [8] stored in multiple files. For computational and comparative convenience, we adopt only part of the data, with attributes shown in the right hand side panel of Figure 1, and denoted by

$$\mathbf{X} = [x_{i,j}]\,; \quad i = 1, 2, 3, ..., n-1, n \quad \text{and } j = 1, 2, 3, ..., p-1, p \tag{1}$$

where $n = 82,332$ is the number of observations and $p = 27$ is the number of variables. This setting does not invalidate our modelling strategy as described below. Given $n \gg p$ and the paper's motivation, the data size is good enough for domain-partitioning based on multiple sampling, training and testing, so we repeatedly sample from it. Our strategy is to randomly select m random observations from n, keeping p unchanged. We denote each sample by

$$[x_{\nu,\tau}]\,; \quad \nu = 1, 2, 3, ..., m-1, m < n \quad \text{and } \tau = p \text{ (or } p^* \leq p) \tag{2}$$

where ν and τ are the equivalents of i and j in Equation 1 and, as we shall see in Section 2.2.3, while for individual trees $\tau = p$ is given by design, it is a randomly sampled quantity, $\tau = p^* \leq p$, for random forests. Our strategy, detailed below, is to repeatedly sample $x_{\nu,\tau}$ from $\mathbf{X}$ over different combinations of ν and τ, for performance parameters.



NAME	TYPE	DESCRIPTION
dur	Float	Record total duration
spkts	integer	Source to destination packet count
dpkts	integer	Destination to source packet count
sbytes	Integer	Source to destination transaction bytes
dbytes	Integer	Destination to source transaction bytes
rate	Integer	Transmission rate
sttl	Integer	Source to destination time to live value
dttl	Integer	Destination to source time to live value
sload	Float	Source bits per second
dload	Float	Destination bits per second
sloss	Integer	Source packets retransmitted or dropped
dloss	Integer	Destination packets retransmitted or dropped
sintpkt	Float	Source interpacket arrival time (mSec)
dintpkt	Float	Destination interpacket arrival time (mSec)
sjit	Float	Source jitter (mSec)
djit	Float	Destination jitter (mSec)
swin	Integer	Source TCP window advertisement value
dwin	Integer	Destination TCP window advertisement value
stcpb	Integer	Source TCP base sequence number
dtcpb	Integer	Destination TCP base sequence number
smean	Integer	Source transmitted packet size mean
dmean	Integer	Destination transmitted packet size mean
tcprtt	Float	TCP connection setup round-trip time ('synack' plus 'ackdat').
synack	Float	TCP connection setup time between the SYN and the SYN_ACK packets.
ackdat	Float	TCP connection setup time between the SYN_ACK and the ACK packets.
attacka	Nominal	Attack type
label	Binary	Binarised transmission – normal or attack

Figure 1: Data generation process and CSV file output (left) and generated data attributes (right)

We work on a labelled data scenario, both binary and multi-class, as graphically illustrated in Figure 2. The binary case distinguishes the highest frequency bar as *normal* data flows versus the remaining nine attributes categorised as *malicious* while the multi-class splits the *malicious* into nine different levels. The scenario provides a classic example of *masked* and *spurious* groupings, hence rendering itself readily for concept learning as outlined in Section 2.2.

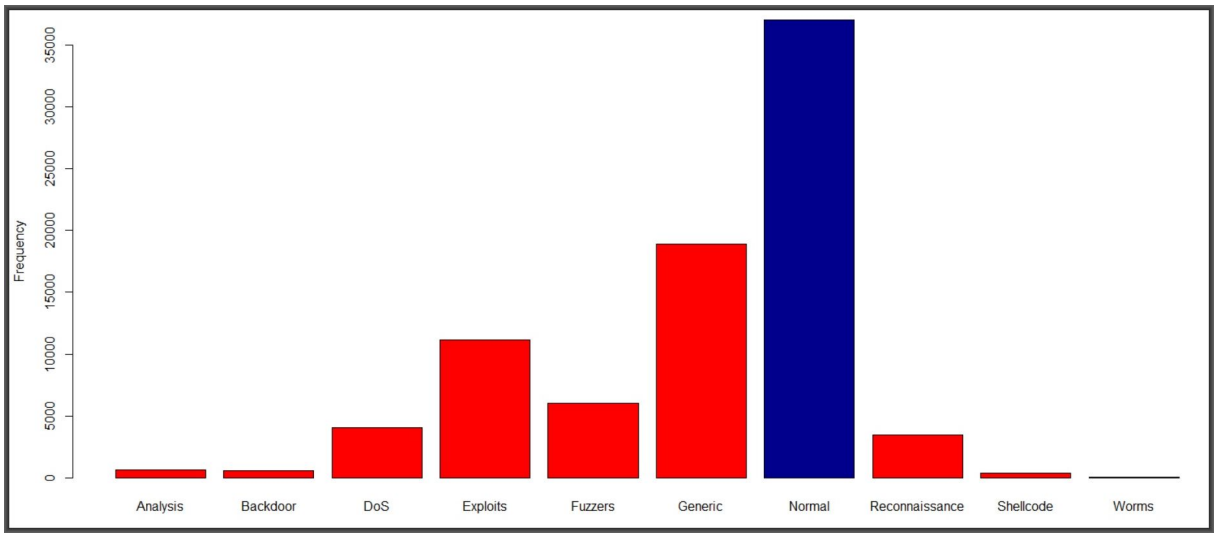


Figure 2: Binary variable *normal* flows are represented by the highest bar while the remaining nine are *attacks*

The data flow types are graphically illustrated in Figure 2. Normal attacks, represented by the blue bar, comprise of 44.94% of the total flow and of the remaining 55.06% categorised as attacks, *Generic* make 41.63%, *Exploits* make 24.56% and *Fuzzers* 13.37%. Other attack types, *DoS*, *Reconnaissance*, *Analysis*, *Backdoor*, *Shellcode* and *Worms* make 9.02%, 7.71%, 1.49%, 1.29%, 0.83% and 0.1% respectively. They represent different types of attacks, for example, *Fuzzers* will attempt to suspend a program or network by feeding it the randomly generated data, whereas *Analysis* consists of different attacks of port scan, spam and html files penetrations. A full description of these data attributes is provided in Moustafa and Slay [15]. Our modelling strategy is outlined in the following exposition.

2.2 Modelling Strategy

Our modelling approach is to train, test and assess multiple models on data sampled from Equation 1, using multiple versions of domain-partitioning algorithms. The process, via **Algorithm** 1, is designed to attain consistency in uncovering intrusion types' partitioning rules and their dynamics. Thus, our goal is two-fold–to identify cyber-attack

predictors through unsupervised modelling and extending their potential into predicting likelihoods of future attacks. As noted above, the two class label scenarios–binary and multi-class, present a classic example of masked and spurious groupings, making an ideal case for concept learning. The main idea is to combine the power of automated learning techniques and existing domain knowledge to uncover networks intrusion patterns–an idea that derives naturally from the overall objective of data mining–extraction knowledge from data. The settings account for the method's novelty, i.e., ability to learn concept rules from highly masked to highly spurious cases while observing model robustness.

2.2.1 Unsupervised Modelling

Two natural challenges to dealing with data of this magnitude is that variables may be highly correlated and sampling from the data typically yields high variability. One standard approach to addressing these issues is through dimensional reduction–i.e., reducing the number of variables by combining them. Two popular approaches to dimensional reduction are data clustering, as described in Maechler et al. [17], in many variants, and Principal Component Analysis, as described in Kambhatla and Leen [18]. As we shall be extracting multiple samples, we are concerned about several data-dependent variations across samples–among them, the proportion of explained variance due to dimensional reduction. This variation is captured by the eigenvalues for each component and it can be visualised via scree plots which provide an indication as to how many components to retain. Figure 3 provides simple illustrations of data–dependent variations. The panels on the left and right hand side are based on samples of size 20 and 1500 respectively, drawn from the subset in Equation 1. The former suggests that we retain just over 10 components, while the latter suggests just over 20 components. The corresponding bi-plots show the influence of the variables in constructing the components–the first two components account for 73.86% of the variation in data for the smaller sample and 40.1% for the larger sample. Implementation through **Algorithm 1** records loadings and the eigenvalues for each retained component, from each sample. The process is repeated many times, paying particular attention to variations in the samples. Over repeated runs, consistency of each variable's contribution to component formation will be observed and stored to determine the final structure which we can then compare to the two known class labels.

2.2.2 A Bayesian Approach to Supervised Modelling

This sub-section presents the Bayesian framework to illustrate its role in classification as originally discussed in Dunsmore [7]. We do not specifically seek to apply the method for intrusion detection as in Sharma and Mukherjee [19], but rather as a tool for using existing prior knowledge to learn about the data behaviour and generate new posterior information, as implied in steps 15 to 17 of **Algorithm 1**. That is, train a classifier on part of $[x_{i,j}]$, test it on $[x_{l\neq i,j}]$ and, given an unlabelled sample $[x_{\nu,\tau}]$ not used in the training, we allocate each flow in the sample, into one of the known classes. The problem is that of **predicting class** k given m **observations** in **sample** $[x_{\nu,\tau}]$ as follows

$$p\left(k|x\right) = \frac{p\left(x|k\right)p\left(k\right)}{p\left(x\right)} \propto \frac{\pi_{k.m} f_{k.m}\left(x\right)}{\sum_{j=1}^{k} \pi_{k.m} f_{k.m}\left(x\right)} \tag{3}$$

where conditional probability $p\left(k|x\right)$ represents the posterior information generated from existing prior knowledge of class membership–i.e., $p(k) \propto \pi_{k.m}$ and $p(x|k) \propto f_{k.m}\left(x\right)$ is the data distribution in the k^{th} class as observed from the m^{th} sample –i.e., the probability of observing x given that we are in the k^{th} class of sample m. Assuming a correct classification incurs no loss, then given data $[x_{\nu,\tau}]$ and classes $\{k_1, k_2\}$ a prediction rule is defined as

$$\frac{p\left(x|k_1\right)}{p\left(x|k_2\right)} > \frac{C_{k_2,k_1} p\left(k_2\right)}{C_{k_1,k_2} p\left(k_1\right)} \implies P(k_1|x) > \frac{C_{k_2,k_1}}{C_{k_1,k_2} + C_{k_2,k_1}} \tag{4}$$

where $p(k_1)$ and $p(k_2)$ are class priors and $C_{k_2,k_1/k_1,k_2}$ represent the cost of incorrectly allocating an observation to a class, which also implies that the probability of class given data is greater than the corresponding loss i.e., $\frac{C_{2,1}}{C_{1,2}+C_{2,1}}$. The cost function was introduced via the Bayesian framework in order to highlight the importance of updating prior with posterior information. It can be shown that the Bayesian decision rule for minimum risk is the weighted sum

$$\Psi = C_{k_1,k_2} p\left(k_1\right)\omega_1 + C_{k_2,k_1} p\left(k_2\right)\omega_2 \tag{5}$$

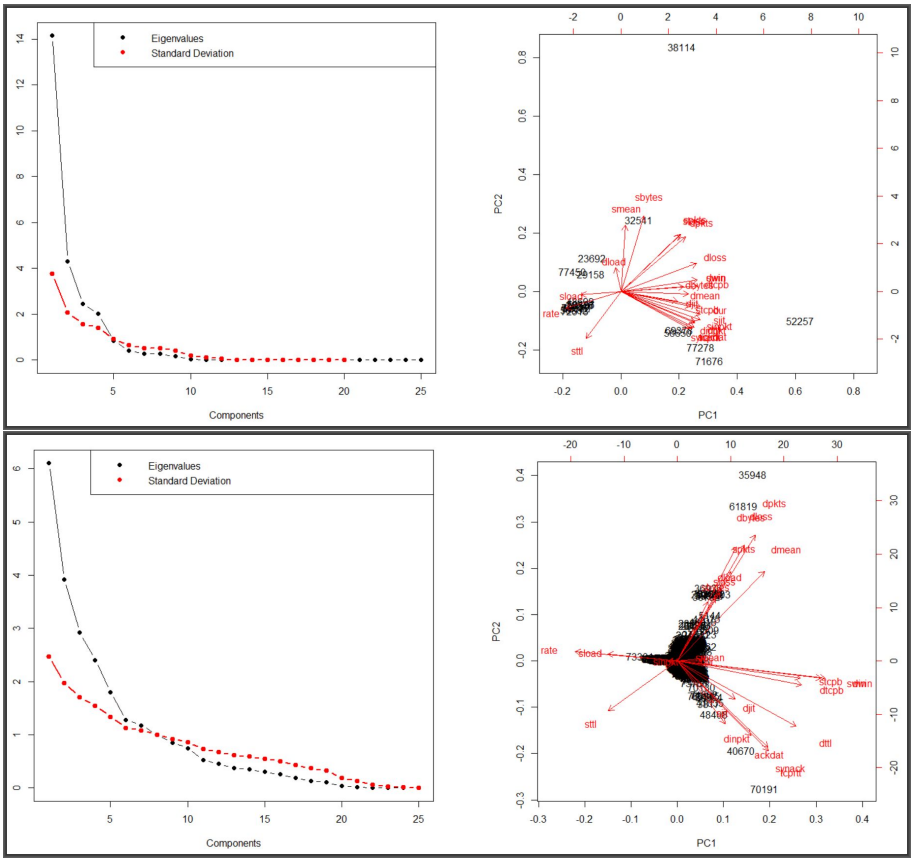


Figure 3: Eigenvalues and standard deviation patterns for sample sizes 20 (left) and 1500 (right)

where ω_1 and ω_2 are the probabilities of misclassifying observations from k_1 and k_2 respectively. The decision rule is

$$\Psi = \frac{1}{n}\sum_{i=1}^{n}\mathcal{L}\left(\hat{y}_i, y_i\right) \quad \text{where} \quad \mathcal{L}\left(\hat{y}_i, y_i\right) = \begin{cases} 0, & \text{if} \quad \hat{y}_i = y_i \\ 1, & \text{otherwise} \end{cases} \tag{6}$$

This framework epitomises the general concept of learning rules from data as applied by a wide range of classifiers, an interesting area of research in Machine Learning (ML) and all Data Science related areas. In their literature survey of various ML methods in cyber security applications, Buczak and Guven [20] focus on the variety and complexity of the methods and note that it is impossible to make one recommendation for each method, based on the type of attack the system is supposed to detect. They identify several criteria for consideration, including predictive accuracy, model complexity and timeliness, and conclude that more work needs to be done in search of effective methods for cyber applications. Classifiers like decision trees, neural networks and Support Vector Machines (SVM) can always be driven to high training accuracy by tuning appropriate parameters, yielding unreliable models. According to Mwitondi and Said [9], striking a balance between model accuracy and reliability is a major challenge in data mining, i.e., the challenge to avoid over-fitting. Bridges et al. [14] consider thresholding of multiple heterogeneous streaming anomaly detectors and they go on to define anomalies as events with low **p**-values. Their algorithm relies heavily on the distributional assumption that the data are sampled from a known distribution. However, Mwitondi et al. [3] show that even with a reasonable consideration of probability distribution of the data and the bounding likelihood of an anomaly, challenges relating to data and model randomness remain. Here it suffices to first recognise that the parameters used in the fitting and, hence, computation of Ψ in Equation 6 are data-dependent. Mwitondi and Said [9] show that the empirical rule is fully associated with randomness due to the allocation region and that the allocation rule is trained, validated and tested on random data. This randomness, exhibited in Table 1, implies that our posterior knowledge is random, and therefore susceptible to variability. This paper seeks to contribute towards addressing this issue.

ALLOCATION RULE ERRORS DUE TO DATA RANDOMNESS			
Population	Training	Cross-Validation	Testing
Ψ_{POP}	Ψ_{TRAIN}	Ψ_{XVALID}	Ψ_{TEST}

Table 1: Data-dependent allocation errors impinging on model fitting and on the computation of Ψ

Notice that Ψ in Equation 6 is a function of the loss function, *priors* and errors. Any domain partitioning algorithm using rules learnt from data is susceptible to allocation rule errors in Table 1. Our work, through **Algorithm 1**, heavily depends on managing the variability in Table 1. Its main idea is to combine existing domain knowledge and automated learning techniques for intrusion detection. Thus, decisions on key parameters such as assessing the misclassification costs would have to be made based on domain knowledge–i.e., with full involvement of system administrators, say.

2.2.3 Implementation Mechanics

This section outlines the implementation mechanics for training, validation and assessing Decision Trees and Random Forests. With two different class labels describing the nature of the data flow, each of the data points in the sampled data in Equation 2 is forced into only one of the classes. In other words, the entire sample $x_{\nu,\tau}$ consists of two structures, which makes it susceptible to obscured classes. Our analyses are based on individually grown Decision Trees (DT) and Random Forests, both due to Breiman et al. [21][10]. With individual trees, unlike Random Forests, the number of variables, τ, is not altered to enable identification of the importance of predictor variables which also minimises the influence of multicollinearity and strength of individual trees. To minimise Ψ, we propose an algorithm that repeatedly samples from the sub-space in Equation 1 and applies multiple Decision Tree models for both training and testing. If the data attributes are used one at a time to split the data into normal and malicious flows by only considering the number of observations at node N^* then, given the attribute $x_{.\tau}$ and the threshold, the rule is

$$\begin{cases} k_1 = & \{\eta \in N^* : x_{.\tau} \leq w \\ k_2 = & \{\eta \in N^* : x_{.\tau} > w \end{cases} \tag{7}$$

The observations in each of the two sets lie on either side of the hyper-plane $x_{.\tau} = w$ chosen in such a way that a given measure of impurity is minimised. While training and testing this rule on random datasets are the main causes of the variations in Table 1, other variations in decision tree model results derive from setting model parameters.

On the other hand, Random Forests are constructed from the training samples drawn from Equation 1 with replacement and with the number of variables also sampled from p. The procedure involves no pruning and so the error rate depends both on the multicollinearity of the trees as well as on the strength of individual trees in the forest. Dimensional reduction (i.e., reducing the number of predictor variables) reduces both multicollinearity and tree strength which makes τ a crucial adjustable parameter. We circumvent this complex scenario by applying random forests with unaltered p and generating multiple versions of DT models to capture the consistency of predictor variables across the repeated runs as outlined in the algorithm below. This approach derives from bagging Breiman [22] but rather than just dividing $[x_{i,j}]$ into fixed training and test sets, multiple bootstrap training and testing samples are repeatedly drawn from it with a fixed p. The two models are repeatedly trained and tested on these multiple samples, recording the key performance parameters, accuracy and reliability. Model optimisation and selection are finally done by harmonising data variability through cross-validation. The algorithm learns a model $F(\phi) = \underbrace{(P)}_{x,y\sim D} [\phi(x) \neq y]$, where D is the underlying distribution, and it provides the mechanics for assessing the models. Particularly when class labels are known, its outputs provide great insights into the overall behaviour of the data particularly how the attributes relate to the target variable as illustrated in the following sections.

Algorithm 1 SMA-Sample, Measure, Assess

1: **procedure** SMA
2: Set $\mathbf{X} = [x_{i,j}]$: Accessible Data Source
3: Learn $F(\phi) = \underbrace{(P)}_{x,y\sim D} [\phi(x) \neq y]$ based on a chosen learning model
4: Set the number of iterations to a large number K
5: **Initialise:** $\Theta_{tr} := \Theta_{tr}(.)$: Training Parameters
6: **Initialise:** $\Theta_{ts} := \Theta_{ts}(.)$: Testing Parameters
7: **Initialise:** $\Pi_{cp} := \Pi_{cp}(.)$: Comparative Parameters
8: **Initialise:** s as a percentage of $[x_{\nu,\tau}]$, say 1%
9: s_{tr} : Training Sample $[x_{\nu,\tau}] \leftarrow [x_{i,j}]$ extracted from $\mathbf{X} = [x_{i,j}]$
10: s_{ts} : Test Sample $[x_{\nu,\tau}] \leftarrow [x_{l\neq i,j}]$ extracted from $\mathbf{X} = [x_{i,j}]$
11: **for** $i := 1 \rightarrow K$ **do**: Set K large and iterate in search of optimal values
12: **while** $s \leq 50\%$ of $[x_{\nu,\tau}]$ **do** Vary sample sizes to up to the nearest integer 50% of X
13: **Sampling for Training:** $s_{tr} \leftarrow X$
14: **Sampling for Testing:** $s_{ts} \leftarrow X$
15: **Fit Training and Testing Models** $\hat{\mathcal{L}}_{tr,ts} \propto \Phi(.)_{tr,ts}$ with current parameters
16: **Update Training Parameters:** $\Theta_{tr}(.) \leftarrow \Theta_{tr}$
17: **Update Testing Parameters:** $\Theta_{ts}(.) \leftarrow \Theta_{ts}$
18: **Compare:** $\Phi(.)_{tr}$ with $\Phi(.)_{ts}$: Plotting or otherwise
19: **Update Comparative Parameters:** $\Pi(.)_{cp} \leftarrow \Phi(.)_{tr,ts}$
20: **Assess:** $P(\Psi_{D,POP} \geq \Psi_{B,POP}) = 1 \Longleftrightarrow \mathbb{E}[\Psi_{D,POP} - \Psi_{B,POP}] = \mathbb{E}[\Delta] \geq 0$
21: **end while**
22: **end for**
23: **Output the Best Models** $\hat{\mathcal{L}}_{tr,ts}$ based on $\mathbb{E}[\Delta] \geq 0$
24: **end procedure**

Initialisation of the algorithm is problem–specific and focuses on the two main problems in data mining–data clustering and classification. Thus, the learning model in line 3 may be initialised without or with class labels. The initial training and testing parameters in lines 5 and 6 are determined by the investigator–these may be from sources ranging from experimental to expert knowledge. A simple example for unlabelled data would be the starting points for running the **K**-Means algorithm or in the case of random forest classification, the number of bagged trees. The comparative parameters in line 6 provide comparative performance measures after each successive iterations, updating the set as the algorithm proceeds and selecting the best performance at the end of the algorithm execution.

3 Implementation, Results and Discussions

Optimal results of random forests obtained from multiple runs are presented in Figure 4 with an estimated Out-of-Bag (OOB) error of 18.41% obtained from training sample aggregation of 1500 trees in the left hand side panel while the predicted traffic structures are in the right hand side panel. Notice that since the random forest classifier aggregates individual trees based on a generic bagging procedure, it may not always be possible to find the best split. Bootstrap aggregation without underlying model relies only on sample reprsentativeness which may not always be guaranteed. It becomes a serious issue with imbalanced class as then, the model will learn more of the highly weighted classes than of others.

As noted above, we circumvent the shortcomings of random forests by applying **Algorithm** 1. One of its key outputs is the tree partitioning in Figure 5 with its overall results showing that the importance of the splitting variables is in the order **sbytes (20), smean (11), dload (11), rate (11), sload (8), dur7, sttl (6), dbytes (6), dmean (5), dpkts (4), dloss (4), synack (3), tcprtt (3), sjit (2), ackdat (2), dinpkt (2), sloss (1), swin (1)** and **djit (1).** Note that the order of the attack types–the bar charts at the bottom of Figure 5 is the same as that in the right hand side panel of Figure 4.

The binary response version of the tree is shown in Figure 6 the overall results of which order the importance of

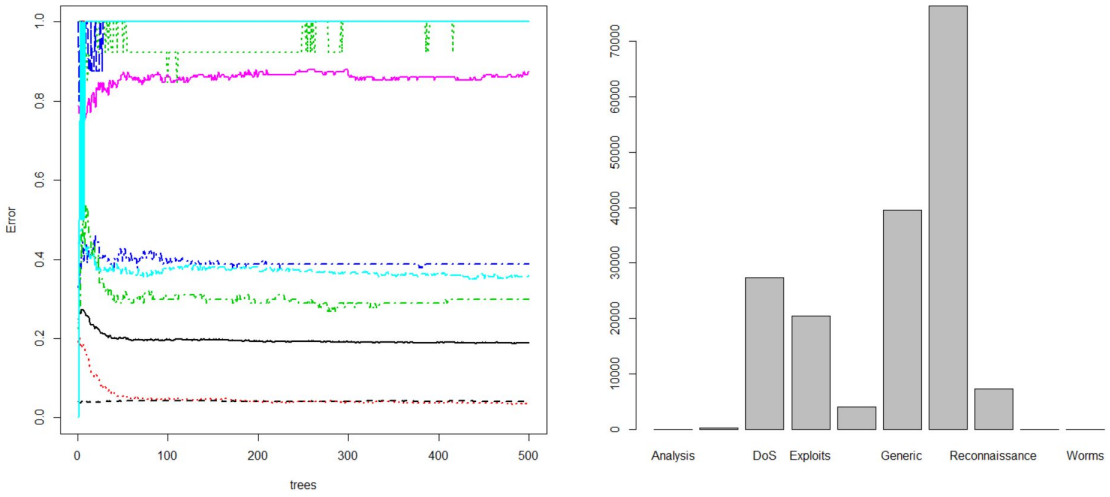


Figure 4: Optimal results of random forests obtained after multiple runs yielded a 18.41% OOB error

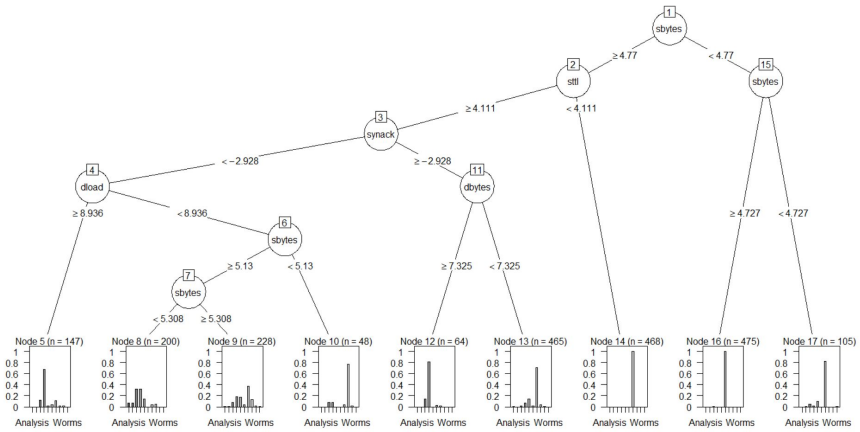


Figure 5: Individual random trees showing the main tree splitting variables and proportions at nodes

variables as follows: **sttl** (14), **dload** (12), **dbytes** (7), **dpkts**, **dloss**, **tcprtt**, **dmean, smean, dinpkt** and **synack** (6), **sjit**, **djit** and **ackdat** (5), **sbytes** (3), **sload** (2) and **rate**, **sinpkt, dttl, dur, spkts** (1). This type of discretisation pools together all types of attacks versus normal flow and while it may look like masking information, it has greater protection potential than breaking them down in that the number of attributes required to identify intrusion is minimised. Performance comparisons of the two targets can be contextualised as follows. System administrators will typically be interested in identifying the specific nature of the attacks and while this knowledge may help them develop specialist deterrents, Zargari and Voorhis [23] note that the dynamic behaviour of attacks can add an extra burden on them, technically and financially. The foregoing concern, further highlights the need for developing enhanced variability monitoring tools. For example, the nine attack types identified in this paper are likely to be highly dynamic and so it is reasonable to expect some attacks evolving into other previously unknown variants or two or more merging.

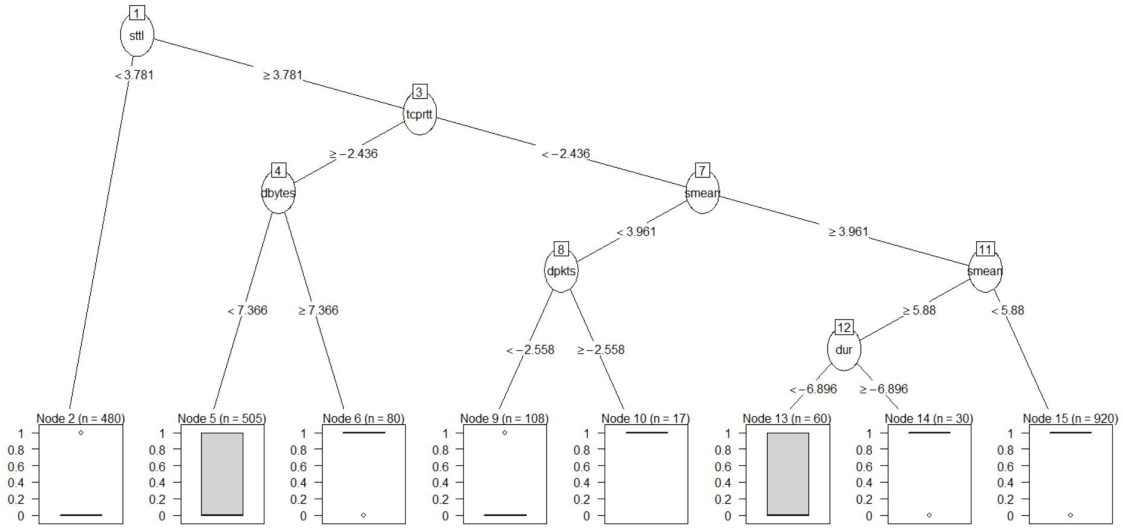


Figure 6: The binary target version of the Individual Decision Trees model

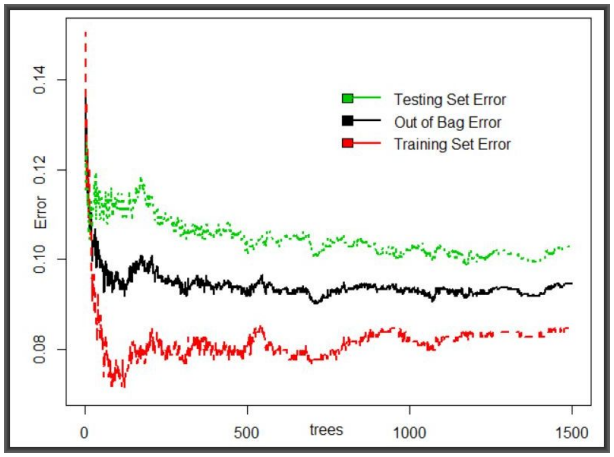


Figure 7: Multiple Decision Tree models on the binary variable

A better option is probably being able to generalise–i.e., isolating normal from malicious data flows. Below is a comparative performance based on specific and general target variables. Figure 7 exhibits error plots based on the binary variable label with an OOB of just over 6% compare this to the 18.41% for the multi-class target above. While it is technically obvious to see why the binary target yields more accurate results than the multi-class target variable, it is imperative to focus on the analytical impact of randomness in the intrusion types as implied in Table 1. One way of achieving that goal is to focus on the variability of the model results which is what this paper sought to achieve.

Various model variability outputs can be captured for comparative purposes. For DT models, these may include the key parameters in Table 2–the complexity parameter, number of splits and variation in the model validation process. We can assess the predictive performance of the model by looking at the root node error in conjunction with the values in this table. The relative error is equivalent to $1 - R^2$ and it represents the error on the observations used to estimate the tree model. The training and validation errors are obtained by multiplying relative error RE and CVE by the $XSTD$ respectively. To avoid over-fitting, the two errors and the standard deviation provide a guide as to when to stop growing the tree–typically, $RE + XST < CVE$, and will feature in $\Theta_{tr,ts}$ in **Algorithm** 1.

Complexity Parameter	Splits	Relative Error (RE)	Cross-Validation Error (CVE)	XSTD
0.479838710	0	1.000000000	1.000000000	0.023526986
0.117943548	1	0.520016129	0.520016129	0.020034240
0.073588710	2	0.422177420	0.439516129	0.018848184
0.072580645	3	0.328629032	0.349798387	0.017233736
0.017137097	4	0.256048387	0.283266129	0.015782203
0.014112903	5	0.238911290	0.246975806	0.014874193
0.010000000	6	0.210685484	0.230846774	0.014439031

Table 2: Performance of the algorithm on binary target data

Table 2 epitomises some of the results from the **SMA Algorithm,** which addresses the randomness issues in Table 1, by searching for optimal values for line 23. The algorithm works with a wide range of unsupervised and supervised models and Table 2 illustrates predictive performance assessment of the model, based on DT's complexity parameter, number of splits and variation in the model validation process. Different complexity parameters and other relevant metrics would be tabled for different models–such as the number of layers, number of neurons and learning rate, in the case of Neural Networks or the margins on hyperplanes, for Support Vector Machines. These parameters are fundamental in determining the quantities in line 19 which, ultimately, determine the desired optimal values in line 23.

4 Concluding Remarks

This paper was motivated by the random nature of analytical studies which has previously inspired many comparative analyses-based classifier design, datasets used and other experimental setups, particularly Moustafa and Slay [15, 16] and Mwitondi et al. [24]. We presented an iterative algorithm that is trained and tested on multiple random datasets with the ultimate objectives being to identify key predictors of intrusion and predict likelihoods of future attacks. The algorithm generates an ensemble model–a derivative data mining technique embedded with data adaptation capabilities for intrusion detection and it is adaptable to various learning algorithms. Its main idea is to combine existing domain knowledge and automated learning techniques for intrusion detection which fits in nicely with the overall objective of data mining–extraction knowledge from data as highlighted in Wu and Banzhaf [4]. The paper provided a unified *unsupervised-supervised* approach to modelling of cyber intrusion dataset. The two examples drawn from binary and multi-class target variable were motivated by the fact that frameworks for attaining the two objectives are based on pre-defined ontologies with inherently highly dynamic parameters. As reported in Mwitondi and Said [9], these parameters tend to randomise not only the training and testing datasets, but also the predictive power of the models. The proposed algorithm can be applied with many learning algorithms and, as we seek to achieve generalisation rules in isolating *malicious* from *normal* data flows, we expect that this work will pave the way for more model comparisons across applications. Results show that the ensemble model conforms to data variability and yields more insightful predictions on multinomial targets. We sought to combine the power of automated learning techniques and existing domain knowledge to uncover networks intrusion patterns.

The nature and purpose of data generation and the two-class label scenario provided perfect settings for our method's novelty–ability to learn concept rules from highly masked to highly spurious cases while observing model robustness. From an intrusion detection perspective, normal and malicious flows do not fit in any current concept definition across tools, and so our work is expected to enhance that understanding from a modelling point of view. The objectives of the paper confined it to developing, training and testing an ensemble models that conform to data variability; carry out predictive modelling of intrusion using historical data and carrying out comparative analysis of the predictive models. It is imperative to envision some of the attributes in Equations 1 and 2 as being encrypted, since encryption is always going to remain part of IT security. For the purpose of this paper, a plain text or numerical variable jumbled into unreadable code will still retain its key feature, describing its type as a data attribute. Thus, while the paper pays no particular focus on Deep Packet Inspection (DPI), **Algorithm 1** has the potential for dealing with encrypted packages, given minor adaptation and the right data attributes. One example of such adaptation is Roni et al. [25] who propose

using the K-Nearest Neighbour method to classify real-time encrypted data.

Finally, the **SMA Algorithm** performed fine on a single machine, but it is reasonably assumed that it is suitably applicable to large, distributed situations. We did not delve further into this, as it is an infrastructural design problem. Thus, while **SMA** has neither been tested for the number of users on a distributed system, nor on multiple processors, it provides new potential for scalable systems for intrusion detection. The algorithm is adaptable to multiple users and concurrent database access–i.e., running distributed applications on multiple servers across geographical and time zones as in Neill and Carloni [26]. The work was completed using open source tools–**RStudio** and **TexStudio** as the authors commitment to promoting **Open Science**. It is expected that our findings will open new research directions into cyber-security and related areas of open science through sharing of data and research findings.

References

[1] Kent, J. T.; Tyler, D. E. Constrained M -estimation for multivariate location and scatter. *Ann. Statist.* **1996**, *24*, 1346–1370.

[2] Mwitondi, K. S. Robust Methods in Data Mining. PhD Thesis. *School of Mathematics, University of Leeds, Leeds: University Press* **2003**,

[3] Mwitondi, K.; Moustafa, R.; Hadi, A. A Data-Driven Method for Selecting Optimal Models Based on Graphical Visualisation of Differences in Sequentially Fitted ROC Model Parameters. *Data Science* **2013**, *12*, WDS247–WDS253.

[4] Wu, S.; Banzhaf, W. The use of computational intelligence in intrusion detection systems: A review. *Applied Soft Computing* **2001**, *10*, 1–35.

[5] Sommer, R.; Paxson, V. Outside the Closed World: On Using Machine Learning for Network Intrusion Detection. *IEEE Symposium on Security and Privacy* **2010**,

[6] Mitchell, R.; Chen, I.-R. Behavior Rule Specification-Based Intrusion Detection for Safety Critical Medical Cyber Physical Systems. *IEEE Transactions on Dependable and Secure Computing* **2015**, *12*, 16–30.

[7] Dunsmore, I. R. A Bayesian Approach to Classification. *Journal of the Royal Statistical Society. Series B (Methodological)* **1966**, *28*.

[8] UNSW-NB15,; ACCS, The UNSW-NB15 data set description. *Cyber Range Lab of the Australian Centre for Cyber Security (ACCS)* **2015**,

[9] Mwitondi, K.; Said, R. A Data-based Method for Harmonising Heterogeneous Data Modelling Techniques Across Data Mining Applications. *Statistics Applications and Probability* **2013**, *Pro 2*, 293–305.

[10] Breiman, L. Random Forests. *Machine Learning* **2001**, *45*, 5–32.

[11] Chapmann, J. *Machine Learning Algorithms*; CreateSpace Independent Publishing Platform, 2017.

[12] Ashfaq, R. A. R.; Wang, X.-Z.; Huang, J. Z.; Abbas, H.; He, Y.-L. Fuzziness based semi-supervised learning approach for intrusion detection system. *Information Sciences* **2017**, *378*, 484 – 497.

[13] Garg, V.; Kalai, A. Meta-Unsupervised-Learning: A supervised approach to unsupervised learning. *CoRR* **2016**,

[14] Bridges, R. A.; Jamieson, J. D.; Reed, J. W. Setting the threshold for high throughput detectors A mathematical approach for ensembles of dynamic, heterogeneous, probabilistic anomaly detectors. 2017.

[15] Moustafa, N.; Slay, J. UNSW-NB15: A comprehensive data set for network intrusion detection systems. *Cyber Range Lab of the Australian Centre for Cyber Security (ACCS)* **2015**,

[16] Moustafa, N.; Slay, J. The evaluation of Network Anomaly Detection Systems: Statistical analysis of the UNSW-NB15 data set and the comparison with the KDD99 data set. *Security Journal: A Global Perspective* **2016**, 1–14.

[17] Maechler, M.; Rousseeuw, P.; Struyf, A.; Hubert, M.; Hornik, K. cluster: Cluster Analysis Basics and Extensions. 2013; R package version 3.4.0.

[18] Kambhatla, N.; Leen, T. Dimension Reduction by Local Principal Component Analysis. *Neural Computation* **1997**, *9*, 1493–1516.

[19] Sharma, N.; Mukherjee, S. Layered approach for intrusion detection using nave Bayes classifier. *ACM Digital Library* **2012**, 639–644.

[20] Buczak, A. L.; Guven, E. A Survey of Data Mining and Machine Learning Methods for Cyber Security Intrusion Detection. IEEE COMMUNICATIONS SURVEYS and TUTORIALS. 2016.

[21] Breiman, L.; Friedman, J.; Stone, C.; Olshen, R. *Classification and Regression Trees*; 1984.

[22] Breiman, L. Bagging Predictors. *Machine Learning* **1996**, *24*, 123–140.

[23] Zargari, S.; Voorhis, D. Feature Selection in the Corrected KDD-dataset. *Third International Conference on Emerging Intelligent Data and Web Technologies* **2012**, *x*.

[24] Mwitondi, K.; Said, R.; Zargari, S. An Ensemble Method for Intrusion Detection with Conformity to Data Variability. Proceedings of the 8th Annual International Conference on ICT : Big Data, Cloud and Security (ICT-BDCS), Singapore, 21-22 August 2017. pp 1–4.

[25] Roni, M., B-Y. Langberg; ; Peleg, D.; Roditty, L. Realtime Classification for Encrypted Traffic. Experimental Algorithms. 2010; pp 373–385.

[26] Neill, R.; Carloni, L. P. A scalable architecture for intrusion-detection systems based on a broadband network of embedded set-top boxes. 2011 IEEE 54th International Midwest Symposium on Circuits and Systems (MWSCAS). 2011; pp 1–4.