International Journal of Security and Its Applications
Vol. 9, No. 4 (2015), pp. 63-80
http://dx.doi.org/10.14257/ijsia.2015.9.4.08

# A Hybrid Mitigation Technique for Malicious Network Traffic based on Active Response

Ayei E. Ibor[1] and Gregory Epiphaniou[2]

*[1]Department of Computer Science, Cross River University of Technology, Calabar, Nigeria*
*[2]Cyber Security Technical Consultant, QA Limited, St. Katharine's Way, London, United Kingdom*
*E-mail: ayei.ibor@gmail.com, epiphaniou@gmail.com*

***Abstract***

*The rapid increase in advanced persistent threats in the cyber space engenders full attention to the use of intrusion detection with emphasis on Artificial Intelligence-based intrusion detection systems as a mitigation mechanism. The sharp increase in attack surfaces can be partially attributed to the fact that Internet becomes the de facto means of converged communications and online transactions accommodating different types of services under the same scheme. Most current intrusion detection systems (IDS) deploy signature patterns of known attacks and anomaly detection approaches in detecting intrusions in an attempt to reduce the computational complexity introduced by large scale data sets. However, these approaches have been proved to be inadequate to detect novel attacks often resulting in a high false positive rate. This research will therefore seek to address the issue of detecting persistent network threats by combining the approaches of misuse and anomaly detection in one system. Our algorithm incorporates the concept of active response against all four broad attack types analyzed in the literature to realize another algorithm for intrusion detection and prevention as well as active response called HYBRITQ-4. The algorithm introduces a mechanism for classifying packets based on protocol information to enhance pattern searches and matching when detecting abnormal packets. Findings from our investigation suggest that the proposed algorithm can efficiently improve the detection rate, false positive rate and accuracy of detecting intrusions in patterns of known and novel attacks.*

***Keywords:*** *Intrusion detection, security, data mining, algorithm, attack patterns.*

## 1. Introduction

More and more computers are being connected to networks across the globe. Most of these networks are part of the global grid – The Internet. As technology evolves every day, the desire to transmit data and do business online at high speeds becomes greater. This desire has brought with it the use of the Internet to reach distances that may in normal circumstances be difficult or may lead to very high transmission overheads. The ease of use of computer networks (local and wide area networks respectively), though elaborately advantageous to data transmission poses various degrees of risks for all users, companies, organizations, institutions and the like.

Malicious activities are on the increase and more information systems of organizations are being compromised. These activities have evolved from mere unauthorized access and modification of data to cyber warfare. The use of a monitoring and prevention system is therefore indispensable as it would help to detect, report and prevent external (and internal) attacks against a network

In addition, systems and networks can be made more secure by learning from previous and recent scenarios of attacks and taking the appropriate measures to eliminate or reduce future occurrence(s) of such events. As discussed in [1], intrusion detection involves the intelligent monitoring of patterns in network traffic or log files for signs of violations of security policies, some of which are based on if-then rules. These if-then rules are based on expert systems, which are constructed using artificial intelligence based techniques. Several artificial intelligence based techniques have been deployed for both misuse and anomaly detection including intrusion detection expert system (IDES), which uses if-then rules to construct an expert knowledge of known attack patterns and vulnerabilities in a system [1].

In this paper, we will deploy the use of an artificial intelligence based data mining approach in extracting patterns of normal data from network traffic and audit data as discussed in [2]. This approach, which uses an algorithm called HYBRITQ-4 will allow the quick detection of traffic patterns by filtering the traffic based on protocol information. This helps to reduce exhaustive search for patterns of known and unknown signatures during the detection phase of the algorithm. In this way, the detection speed and accuracy is enhanced. One limitation of this approach is that the initial training time of the algorithm may affect the speed of detection although this is significantly improved on subsequent iterations.

## 2. Related Literature

An intrusion is an illegal action or transaction initiated by a hacker (malicious user) against a controlled environment in order to circumvent its security. As asserted by [3], intrusion detection and prevention systems (IDPS) were developed to counter the numerous cyber attacks perpetrated by these malicious users across the globe. Detecting and preventing network intrusions comes with a wide range of approaches. Most of these approaches are based on data mining, pattern matching, machine learning, classifier and genetic algorithms.

In [4, 5], the use of artificial immune system for intrusion detection is proposed, which is inspired by biological modeling and techniques. Also, [6] deployed the use of feed forward neural networks underpinned by the back propagation training algorithm to classify known and novel attack patterns. The use of a genetic algorithm in hybrid evolutionary neural network for intrusion detection was investigated by [7, 8] for identifying complex anomalous behaviors in network patterns. Data mining approaches are also considered to deploy an aspect of artificial intelligence in extracting patterns of normal data from network traffic and audit data as discussed in [2].

As described in [9, 10, 11] data mining involves the extraction of knowledge (knowledge discovery) from a set of mass data points that may be incomplete, noisy, fuzzy and random in order to discover patterns in the data needed for understanding the structure of a particular system. Data mining algorithms exploit the efficiency of this technology to enhance intrusion detection by fetching the required patterns including novel patterns from network data and host logs that can be used for identifying normal and malicious packets by an Intrusion detection and prevention system (IDPS).

Pattern matching algorithms use sequences and tree structures to discover the presence of the constituents of a pattern that may be classified as a normal or an abnormal string thereby improving the efficiency of intrusion detection systems (IDSs). According to [12, 13, 14], the increase in network traffic and data flow requires the services of IDSs, which will compare the strings in the rule-set with patterns in the network data flowing across a network. This comparison will help to discover patterns that may match attack signatures in the headers and payloads of transmitted packets.

Machine learning and classifier algorithms are based on inductive learning approaches in which a system is trained to identify patterns of strings from training data. Such patterns are then stored and used subsequently for identifying the presence or absence of an intrusion [2]. On the other hand, genetic algorithms employ the techniques of natural evolution, which depend on selection and genetics to perform heuristic searches. As claimed by [15] and [16], the strength of genetic algorithms is a function of its parallel traversal of the search space in such a way that randomly generated solutions are proposed at the beginning and continuously evaluated with a fitness function. A fitness function combines objectives and constraints in a single function with the use of arithmetic operators such as the weighted sum [16].

### 2.1. Intrusion Detection and Prevention Techniques

Detecting and preventing intrusions on a network is done based on some techniques. These techniques are discussed below:

#### i. Misuse Detection

In this technique, signatures of well known intrusions are stored in a database and used to detect unlabelled data sets by matching these patterns against the stored signatures. As asserted by [17], the fingerprints of known attacks are identified based on the stored signatures, which must undergo frequent updates to include information about novel intrusions. This reduces the rate of false alarms in IDSs and improves the detection efficiency. Figure 1 below illustrates the mechanism of a misuse detection technique.

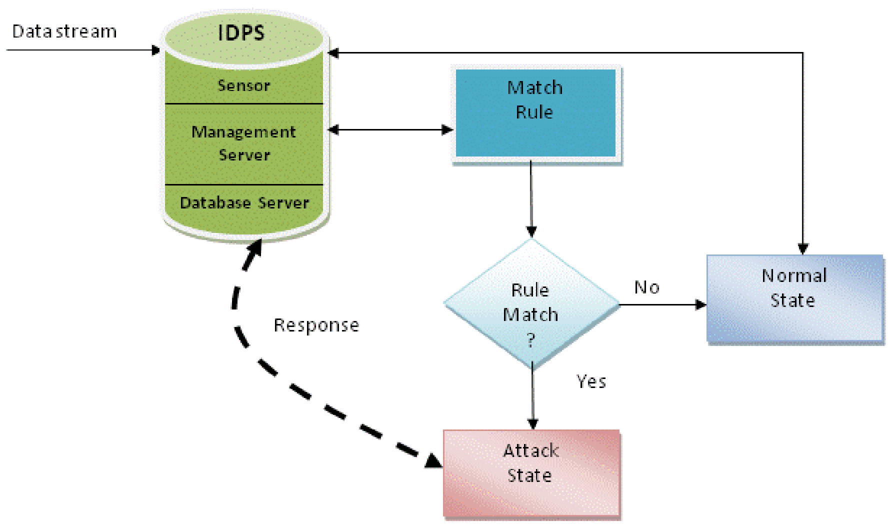

**Figure 1. Misuse Detection Technique for Intrusion detection**

As shown in Figure 1, network traffic that reaches the IDPS is captured and patterns in the data stream are matched with rules in the signature database. An intrusion is flagged when a match is found. Subsequently, the database is updated with this new information.

#### ii. Anomaly Detection

The anomaly detection approach stores profiles of normal behavior in its feature database using normal data. These stored profiles are compared with network activities or traffic and a deviation from the normal model is flagged as an intrusion [17, 18]. This approach can detect zero-day exploits though it can lead to a high rate of false alarms since legitimate but previously unseen behaviors can be classified as intrusions.

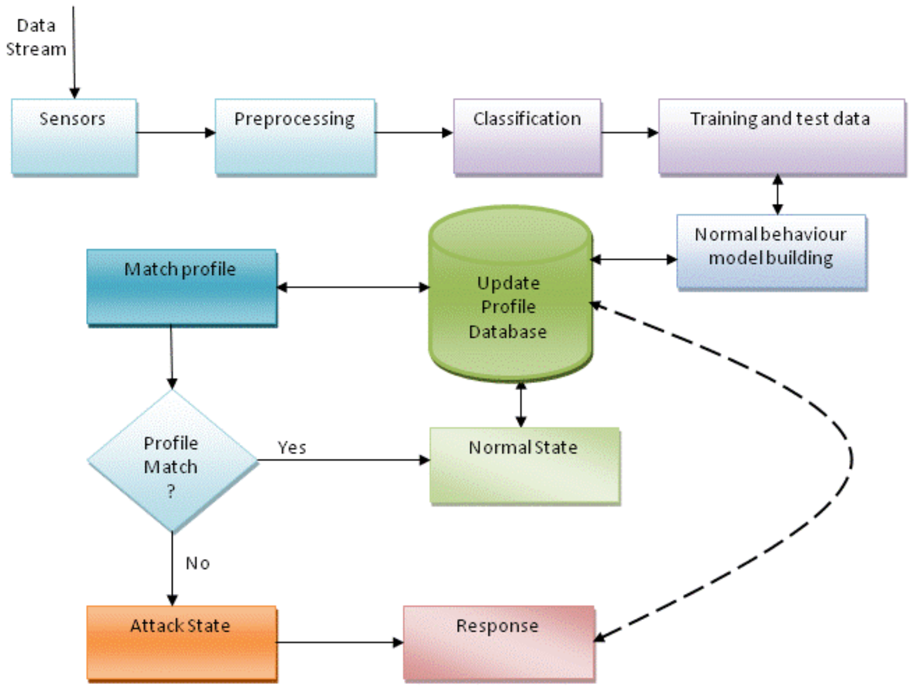

**Figure 2. Anomaly Detection Technique for Intrusion Detection**

Figure 2 depicts the anomaly detection process. Data streams captured by sensors undergo the following stages:

**Preprocessing**: the captured packets are normalized and the relevant features extracted. As indicated in [19], a structured dataset is created from these features while removing irrelevant and redundant information as well as noisy and unreliable data in the dataset. This is to enhance the process of knowledge discovery during the training phase.

**Classification**: Algorithms such as J48, Naïve Bayes, and ID3 are used for classification [2]. This process enables new observations or behavioral patterns to be put into the appropriate classes where a training set has pre-determined classes of datasets. These observed instances are analyzed into a set of quantifiable properties for building the normal behavior of the system. With classification, pattern recognition can be achieved, which is leveraged by assigning a specific output value to a given input value.

**Training data and test data**: these datasets are used to predict the output of a system based on a given set of inputs. A training set usually consists of a tuple of attributes for both the input and output that can be used either to train a knowledge-based system or predict the state of a system at any point in time. The test data is usually compared against the training data to determine the efficiency of the system while predicting an outcome based on the current events.

**Building model of normal behavior**: the normal behavior of the system is modeled based on the features selected using a classifier. The process of building a model of normal behavior is described in [18]. This usually can be based on unsupervised, supervised and semi-supervised techniques. In an unsupervised technique, an unlabelled test data is used with an assumption that most of the instances in the dataset exhibit normal system behavior. This can be achieved through searching the dataset for instances that may seem to fit least to the minority of the dataset.

In a supervised method, the dataset has to be labeled as normal or abnormal with a classifier being trained to identify these classes of behavior. With the semi-supervised approach, a given set of normal training data is used to construct a model of normal behavior. This model is then tested using test data.

The built model of normal behavior is subsequently stored in the profile database and used to match current events in the system in order to detect abnormal instances of the events. Where there is a profile match between the stored profile and the current event, the system is considered to be in a safe or normal state. A mismatch triggers an attack state, which activates the response module to take the appropriate countermeasure. In each case, the profile database is updated based on the current state of the system.

### 2.2. Existing intrusion detection systems and their limitations

A range of commercial and open source intrusion detection systems are currently being deployed in networks across the globe either as network or host based intrusion detection systems. Commonly used IDSs such as SNORT NIDS (Network Intrusion detection System) and OSSEC (open source security) are based on misuse detection models.

Identifying efficiency and accuracy, as the underlying problems of current IDSs, owing to the fact that they rely mostly on stored signatures of known attack vectors for detecting intrusions, underpinned the work of [17]. An attacker who knows the port a particular service is running on can scan a large block of addresses for which different machines are listening and evade the IDS.

Also, scrambling a packet with such techniques as encryption could make it difficult for IDSs to detect malicious traffic. A simple way of doing this is to change the first byte of a payload or encrypt the tunnel of communication so that the traffic becomes unintelligible to the IDS. Some of the drawbacks of SNORT are discussed in [20], who proposed a performance enhanced intrusion detection and prevention system called BSnort (Better Snort) for the detection and prevention of denial of service (DoS) attacks. Using OSSEC involves the use of rules, which define the respective legitimate traffic on a host. Updating these rules between different versions of the IDS may pose problematic as well as being able to co-ordinate pre-shared keys between the server and the client. Also, if the system is compromised, OSSEC may not perform well in alerting the system of intrusions. The processes of intrusion detection and prevention can be summarized using the simple process model in figure 3.

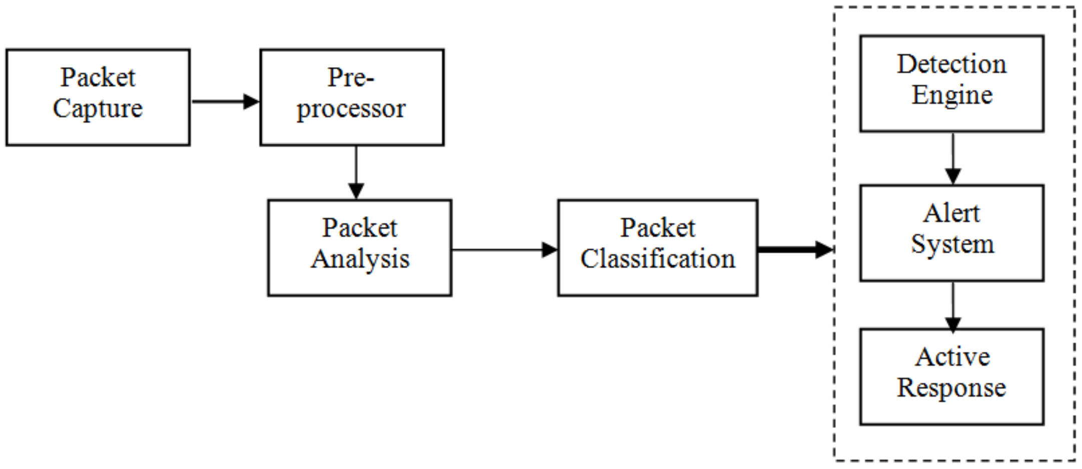

**Figure 3. A Simple Process Model of the Proposed Algorithm**

## 3. Classes of Attack Vectors and Vulnerabilities

The need to operate secure information systems requires the corresponding identification and classification of attack vectors and vulnerabilities. This will provide a platform to provide adequate measures against them. Security assessment may be a herculean task in most information systems and networks. This may arise from the fact that more and more vulnerabilities spring up as a result of expansion in resource sharing such that the detection and patching of known vulnerabilities do not in any way guarantee the absence of others [21]. Also, there may be no standard metrics to measure the security of a system.

An attack vector is a method or technique by which a user with malicious intent can penetrate a network without pre-assigned access rights and privileges. With attack vectors, hackers (or crackers) can exploit the vulnerabilities of a system in order to deliver malicious payloads meant to distort the functionality of the target system. Some common attack vectors include worms, Trojan horses, spyware, backdoors and rootkits, viruses, email attachments, pop-up windows, instant messaging and web links. Vulnerabilities can be identified as the inherent flaws in a system, which can allow a malicious user to take advantage of the system to the extent of reducing the system's information assurance. An investigation by [21] as well as [22] revealed that most vulnerabilities arise from software development and poor configuration of applications and services. There are also vulnerabilities that may arise from the physical environment of the system, management procedures and security policies, hardware, equipments and facilities used for communication, service delivery, and business operations. A summary of the four major types of attacks is given in Table 1.

**Table 1. A Description of Common Computer Attacks**

| **Attack type** | **Description** |
|---|---|
| **Probe** | Probing attack is a passive attack type based on footprinting and social engineering techniques. The main target of probe is to gather as much information about a particular network of computers or an organization as possible. In [23], it is stated that the information acquired through probing can help a malicious user circumvent the security controls of a network or system. |
| **Denial of Service (DoS)** | DoS attack is an attack on the availability of system resources. The attacker usually interrupts the normal state of a system by flooding the system (usually a host or server machine) with a huge number of requests that will completely use up the available computing or memory resources to the extent of disrupting the normal system operation or the suspension of available services [23] |
| **User to Root (U2R)** | This attack is underpinned by an attacker being able to exploit the vulnerabilities of a system after successfully compromising a normal user's account including the escalation of privileges in order to gain root access in the target system [24]. Having root access to a system means the ability to have total control of the system's resources including programs, data files, memory, processor, disk and registry. User to root attacks can result from bugs in the operating system such as buffer overflows, rootkits, loadmodule and perl. |
| **Remote to Local (R2L)** | In this attack, a malicious user transmits packets to the target machine in a network so that he can gain access to the machine as though he owns an account in the machine [23] [24]. Most remote to local attacks are achieved using techniques in social engineering. Common R2L attacks include ftp_write, multihop, guess_passwd, spy, warezclient and warezmaster attacks respectively. |

## 4. Experimentation

Several experiments were performed to simulate the architecture of HYBRITQ-4. These experiments were based on two samples of data, one as the training set and the other as the test set to validate the model's ability to predict new data, given training instances.

### 4.1. Components of the Algorithm

Table 2 will highlight the components of HYBRITQ-4

**Table 2. Components of HYBRITQ-4**

| **Component** | **Description** |
|---|---|
| Packet Capture module | Connection vectors are created to represent incoming connections, which undergo pre-processing. As described in [19], [25], the pre-processing phase involves the normalization of the captured packets to a form recognizable by the IDPS. The original data patterns will be converted to binary values and forwarded to the analysis engine.<br>During the pre-processing phase, the detection rules are classified based on the protocol information available on the captured data. Four classes of protocols are identified comprising TCP, UDP, ICMP and IP [20]. |
| Pre-processor and packet decoding | The initial pre-processing involves the grouping of packets on protocol information and has the following advantages:<br>i. It reduces the number of searches needed to detect an intrusion thereby reducing the detection time<br>ii. It simplifies the rule matching automata, which reduces the depth of the decision tree leading to minimal use of memory during searches. |
| Packet analysis | Performs deep packet inspection and extraction of packet header information for packet classification |
| Statistical classifier | The statistical classifier uses the protocol information already extracted from the packets to determine the type of intrusion that must have been injected into the packet. The classification uses the J48 algorithm. This algorithm supports the classification of a set of data points using decision trees as described in [2]. |
| Detection engine | The detection engine detects intrusions using two methods; misuse and anomaly detection. Misuse detection will be achieved using the Boyer-Moore algorithm, which is a perfect string matching algorithm that is able to match an entire string representing a data packet with the attack signatures stored in the database [20]. Anomaly detection is achieved using the k-NN (k Nearest Neighbor) algorithm. This is a machine learning algorithm that is used to classify objects with respect to the closest training examples in the feature space. |
| Intrusion alerting, reporting and response | When an event has been flagged as normal or anomalous, the output is forwarded to the alert system that sends a signal to the alert message generation module to trigger the appropriate action. The alert manager generates signals that notify the IDPS of the presence or absence of an intrusion. In this way, the IDPS can perform one of the following actions: |

| | |
|---|---|
| | i. Respond: the IDPS responds to the alert in order to take the appropriate countermeasure. This can be in the form of an email message, screen display, or a beep, which is sent to the system administrator to notify him/her of the presence or absence of an intrusion.<br>ii. Protect: the IDPS terminates the connection, blacklists the identified IP address with malicious contents or drops the packet.<br>iii. Trace: the IDPS traces the source of the intrusion to identify the IP address of the original packet. Log files can also be traced to verify the events leading to the intrusion.<br>iv. Detect: the IDPS detects an intrusion by flagging an event as anomalous. |

## 4.2. The Algorithm's Architecture

The architecture of the proposed algorithm is depicted in Figure 4. Packets are captured with sensors or agents that are placed at strategic points on the network and also from a firewall. These sensors (or agents) collect network audit data and/or log files from the activities in the network. The data collected by the sensors is fed to the network or host sensor manager, which populates the pre-processor with the unclassified (raw) data. The captured packets may be normal (legitimate network traffic) or abnormal (attack patterns). The protocol information extracted from the packets is used to redirect the packets to the rule set for each protocol, to quicken the search process.

Performing deep packet analysis will allow the algorithm to classify packets according to the normality principle. The normality principle will be defined in this work as follows:

$$x: x \in N \qquad (1)$$

N is chosen from a set of training data, identified as normal data.
Data is normal if it has no element (or traces) of intrusive behavior. Every data pattern that satisfies the normality principle is classified as normal traffic. An intrusive behavior is such that satisfies the following relation:

$$\mathbf{x{:}x \notin n;\ \forall\ n \in N} \qquad (2)$$

***n*** is a subset of **N**: n is a sum total of all data patterns that represent one form of abnormality or the other. However, this does not necessary imply that ***n*** represents an intrusion.

The algorithm examines packet headers to extract protocol information needed to classify the captured data and redirect the analysis of the packet based on the protocol identified in the packet. This will be modeled as follows:

$$T \rightarrow R_{set} : \forall\ Pr \in T, \exists r \in R_{set} \rightarrow Pr, Pr\{TCP, UDP, ICMP, IP\} \qquad (3)$$

T is a packet captured by the IDPS defined as a 9-tuple such that

$$T = \{Pr, SRC, DST, SYN, ACK, RST, FIN, PSH, URG\} \qquad (4)$$

Where ***Pr*** is the protocol; ***SRC*** is the source port; ***DST*** is the destination port; ***SYN*** is the SYN flag; ***ACK*** is the acknowledgement flag; ***RST*** is the TCP reset flag; ***FIN*** is the FIN flag
***PSH*** is the push flag and ***URG*** is the urgent flag

$R_{set}$ is a set of classification rules that identifies the state of a packet based on (1)
$R_{set}$ is a collection that will comprise of the following set of rules:

i. DST set to 0

ii. SYN, ACK, RST, FIN, PSH and URG set to 1
iii. SYN, ACK, RST, FIN, PSH and URG set to 0
iv. SYN and FIN set to 1
v. ACK set to 0 with ACK_NUM ≠ 0
vi. SRC and DST set to 21

***Pr*** is a 4-tuple of protocols of type ***TCP, UDP, ICMP or IP***, **r** is a rule that matches a particular protocol such that ***r→ Pr.***

Some packets can also be forwarded to the detection engine where such packets' protocol information cannot be explicitly ascertained. The inspection is carried out as a precautionary measure for identifying malformed packets that may be protocol non-compliant, viruses, intrusions, or those that do not satisfy the standard packet format. The resultant effect is to decide whether the packet should be allowed into the network or routed to another destination. This analysis helps the IDPS to capture statistical information about the packets and forward it to the statistical classifier.

The statistical classifier uses data mining techniques based on the J48 algorithm to extract features needed to give a clear classification of the packet. The classified packet strings are used to populate the detection engine for onward intrusion detection. Intrusion detection is done in two ways – misuse and anomaly detection respectively.

Misuse detection is underpinned by matching the packet strings of the captured packets against pre-defined rules using the Boyer-Moore perfect string matching algorithm. When a misuse is detected, an alert is generated and sent to the alert manager. For anomaly detection, the k-Nearest Neighbor (k-NN) algorithm is used and events are classified as either normal or anomalous after the training and test phases, in which the algorithm relates the captured packets to the defined rules based on some distance similarity function.

Anomalous events are classified as intrusions with a class value of -1. Such intrusions trigger alerts from the alert manager, which concurrently initiates actions to respond, protect, trace or detect the event (packet) in order to secure the resources in the network.

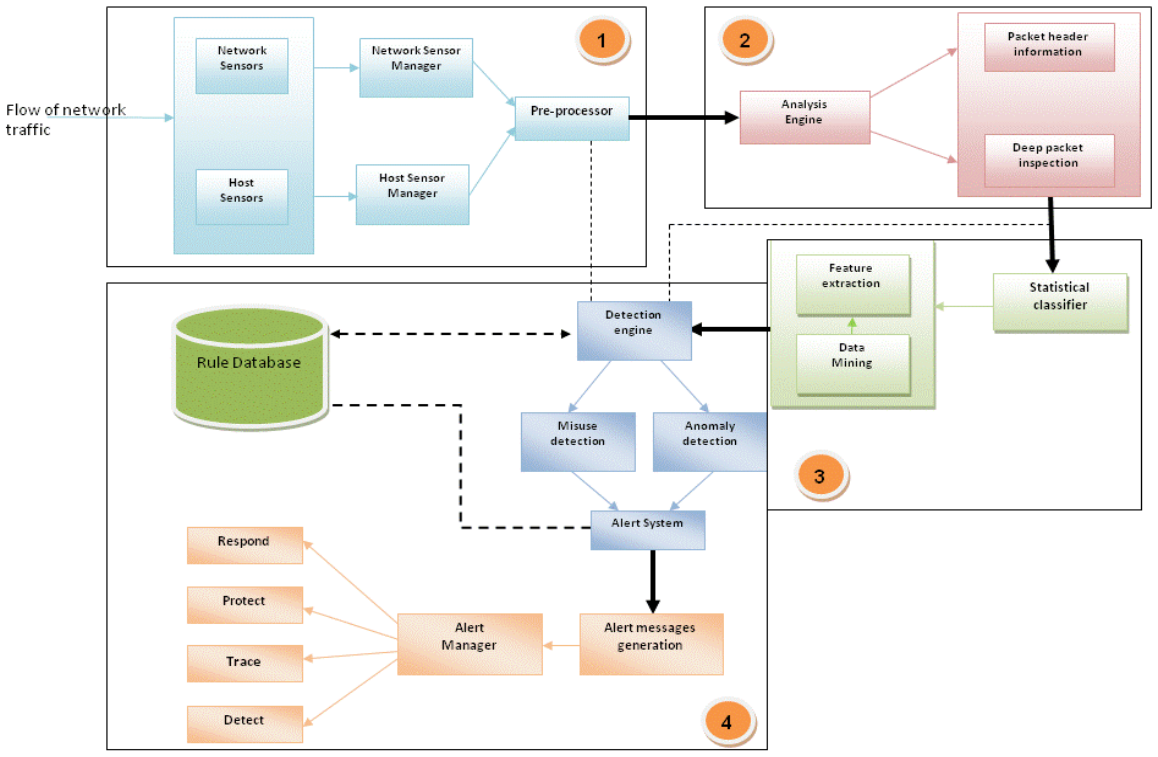

**Figure 4. Architecture of HYBRITQ-4**

As shown in Figure 4, the components of the algorithm are numbered as follows:

1. Packet capture and pre-processing
2. Analysis module
3. Statistical classifier
4. Detection and alert modules.

### 4.3. Cross Validation Test

The 10-fold Cross validation test (also referred to as rotation estimation) is used in this work to validate the proposed model. The purpose of this test is to partition the training data into 10-folds such that the classifier can be evaluated in terms of accuracy over one partition after it has learned from the other partitions. The classifier repeats this process on all partitions, evaluating them in turn [2]. This allows the classifier to make more accurate predictions when supplied with new data.

### 4.4. Experimental Results and Discussion

This section will discuss the series of experiments performed to validate the proposed algorithm. The performance of the algorithm will be evaluated using a randomly selected subset of the 10% KDD'Cup 99 dataset, which has known intrusions and normal data. This dataset is used for training while the corrected dataset is used as test data. WEKA (Waikato Environment for Knowledge Analysis) and MATLAB were used for the data preparation, classification and simulation of the proposed algorithm. The experiments are performed on a Pentium IV CPU running Microsoft Windows 8 Pro 32-bit operating system, dual core 2.13GHz and 2.0GB RAM.

#### i. Data Preparation

The KDD'Cup 99 dataset is one of the most widely used datasets for evaluating intrusion detection systems. This dataset is a result of the evaluation program of Defense Advanced Research Projects Agency - DARPA'98 for IDSs. As indicated in [23], the dataset is composed of training dataset of about 4.9 million single connection vectors. Each vector contains 41 features and is labeled either as normal data or an attack type. An attack type is either a Probe, DoS, U2R or R2L attack.

Due to the enormous volume of this dataset, a subset of the 10% KDD'Cup 99 dataset will be used in this work for the experiments. The 10% dataset has 22 known attacks in the training data and 39 attacks in the test data. The number of each attack type in the training dataset and its percentage score in the dataset is shown in Table 3.

**Table 3. Distribution of Connections in the Dataset**

| Class of connection | Number of connections | % of Total |
|---|---:|---:|
| Normal | 97,278 | 19.69 |
| Denial of Service (DoS) | 391,458 | 79.24 |
| Probe | 4,107 | 0.83 |
| Remote to Local (R2L) | 1,126 | 0.23 |
| User to Root (U2R) | 52 | 0.01 |
| **Total** | 494,021 | |

**ii. Feature Selection**

Due to computational complexity, not all the 41 features were used in the experiment. Out of the 41 features in the dataset, only nine (9) of the features were used for the experiments based on best performance evaluation. Two methods can be used for feature selection – the wrapper and filter methods respectively. This experiment will use the wrapper method, which uses a subset evaluator to create all possible subsets from the feature vector. The J48 classification algorithm is then used to determine the subset of features for which the classifier performs best and the classification of the training data [2, 26].

The evaluation of the attributes (features) in the dataset by the classifier produces a set of features for which the classification will have best performance. Each selected attribute has an associated class label. The class attribute is used only for predicting the attack type during the training and testing phases based on the contents of the feature set. The class of an attack can be one of the followng:

- DOS: denial of service attack
- PROBE: survellance such as port scanning
- R2L: remote to local attack
- U2R: user to root attack

Since the proposed technique in this research work uses the k-nearest neighbor algorithm for detecting outliers in the dataset, the class labels with nominal values DOS, PROBE, R2L and U2R were discredtized as follows: DOS = 1, PROBE = 2, R2L = 3 and U2R = 4.

The total number of instances (attack and normal) as well as percentage value of each used for the experiments is shown in Table 4.

**Table 4. Total Instances used in the Experiments**

| **Connection** | **Discrete value** | **Number of Instances** | **% of Total** |
|---|---|---|---|
| NORMAL | | 6578 | 37.18 |
| DOS | 1 | 6474 | 36.60 |
| PROBE | 2 | 3496 | 19.76 |
| R2L | 3 | 1088 | 6.15 |
| U2R | 4 | 54 | 0.31 |
| **Total** | | 17690 | 100.00 |

**iii. Training and Testing of Model with 10-fold Cross Validation:**

A 10-fold cross validation test is applied to the training data to build the model that is used to predict subsequent instances of the given labels. Each connection record has a class attribute that identifies its associated attack type or normality based on the parameters of the connection. The experiment is run over ten (10) iterations on the training set and the model built is tested using test data to validate its efficiency. The results of the experiments are analyzed with respect to the parameters discussed in Table 5.

**Table 5. Parameters for Describing the Results of Experimentation**

| **Parameter** | **Description** |
|---|---|
| Detection rate (DR) | The rate at which an attack is detected from a sample space of attacks |
| True Positive Rate (TPR) | This is the rate of correctly predicted instances of attacks or normal connection |
| False Positive Rate | This is the rate of wrongly predicted instances of attacks or |

| | |
|---|---|
| (FPR) | normal connection |
| Precision Rate (PR) | PR represents the ratio of instances in the dataset that are relevant |
| Recall Rate (RR) | The relevant instances, which are retrieved |
| F-Measure | Gives a measure of the accuracy of the model |
| Percentage error rates | • Mean absolute error: measures the difference between a predicted quantity and its actual value.<br>• Root mean square error: this gives a measure of the deviation of predicted values by a model from the actual observed instances. It is a good measure of accuracy for a given variable in a model based on different estimators.<br>• Relative absolute error: evaluates the performance of the model based on the size of its input data.<br>• Root relative squared error: evaluates the performance of the model based on its predictive capability [27] |

The error rates give an insight of how close the algorithm can make predictions in relation to true values as a measure of efficiency of the system. The data obtained from 10 different iterations with a 10-fold cross validation test is represented in Tables 6, 7 and 8 respectively. These data will be used to analyze the performance of the proposed algorithm and comparisons made with existing techniques. Since there are 10 different runs each based on 10-fold cross validation, it implies that we have 100 instances of the results. For a good probability distribution to be obtained, average values of the 10 runs were used.

The algorithm shows a constant predictive capability for different iterations; indicating that it is stable and well defined. Error rates are almost constant as well as the degree of precision of the algorithm. Therefore, these indicators are proof of a workable and robust algorithm as used in this approach.

**Table 6. Error Rates and Entropy Gain of the Model**

| Run | Kappa statistic | Mean absolute error | Root mean square error | Relative absolute error | Root relative squared error | Entropy gain |
|---|---|---|---|---|---|---|
| 1 | 0.993 | 0.002 | 0.042 | 0.854 | 11.322 | -4350.690 |
| 2 | 0.996 | 0.002 | 0.034 | 0.633 | 9.187 | 998.915 |
| 3 | 0.995 | 0.002 | 0.036 | 0.669 | 9.693 | -1146.690 |
| 4 | 0.993 | 0.002 | 0.041 | 0.818 | 10.961 | 987.743 |
| 5 | 0.995 | 0.002 | 0.037 | 0.676 | 9.915 | -77.236 |
| 6 | 0.997 | 0.001 | 0.030 | 0.543 | 8.198 | 1003.152 |
| 7 | 0.996 | 0.002 | 0.031 | 0.598 | 8.355 | 3149.522 |
| 8 | 0.994 | 0.002 | 0.039 | 0.819 | 10.648 | 2051.642 |
| 9 | 0.994 | 0.002 | 0.039 | 0.774 | 10.548 | 990.220 |
| 10 | 0.994 | 0.002 | 0.038 | 0.795 | 10.394 | 997.680 |

Error rates are widely used to calculate the accuracy of a system given certain inputs with envisaged output. The kappa statistic of the 10 different folds for the ten iterations gives a value between 0.99 and 1. This is a confirmation of the percentage of correctly classified instances as against the near negligible value of the misclassified data. Kappa statistic is usually used to assess how the classified instances agree with true values in terms of assigning data to categories given certain inputs [27]. If kappa-statistic is greater than zero (that is kappa-statistic > 0), then there is the likelihood of having a good classifier and as such the proposed model has good performance measure.

The mean absolute error (mae) measures how close values predicted by a model are to the actual outcome. With the errors for the 10 iterations being less than 1, it implies that the model has more than 99% of accuracy. This is clearly indicated by the root mean square error (rmse), which computes the deviations of the predictions from actual data. A value less than one for the root mean square error (that is, rmse < 1), shows a stable and good model. Evaluating the performance of a model can be affected by the size of the input data. However, this model performs best with a larger sample space and the relative absolute error (rae) is used to show how the model performs for different folds of the dataset. With 1 > rae (relative absolute error), there is enough evidence to claim that the model's performance is not deteriorated by the input size for all iterations of the 10-folds. Therefore, the relation such that mae < rmse < rae <1 is a function of a good performing model with a high degree of accuracy [27].

The root relative squared error (rrse) measures the accuracy of predictions of a model based on a fitness function in order to avoid over-fitting. Over-fitting in a model occurs when the complexity of a model does not allow it to make accurate predictions. In this case, the model may tend to describe the noise in the dataset rather than the underlying relationship between attributes. With a 10-fold cross validation, it can be seen from the values of the root relative squared error that there is no over-fitting problem in the model hence its accuracy.

The comparison between the kappa statistic and error rates of the model is depicted in Figure 5.

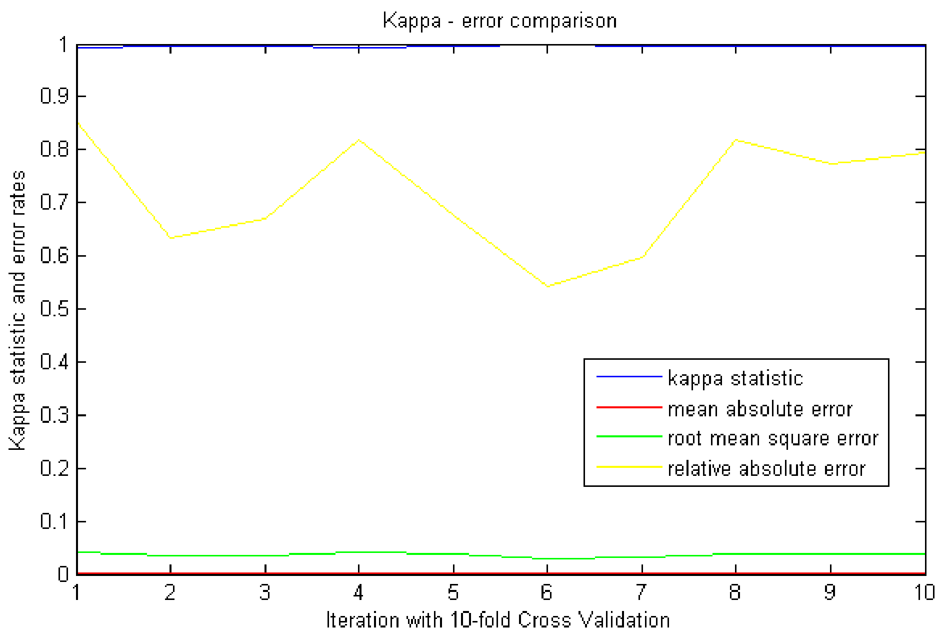

**Figure 5. Kappa – Error Comparison**

The entropy gain, which describes the degree of randomness of a dataset as shown in Table 6 and Figure 6 respectively, provides information for understanding the probability density of the distribution. With very low and high entropy values, it can be said that the dataset has characteristics of an even distribution and therefore can be predicted with a high degree of accuracy.

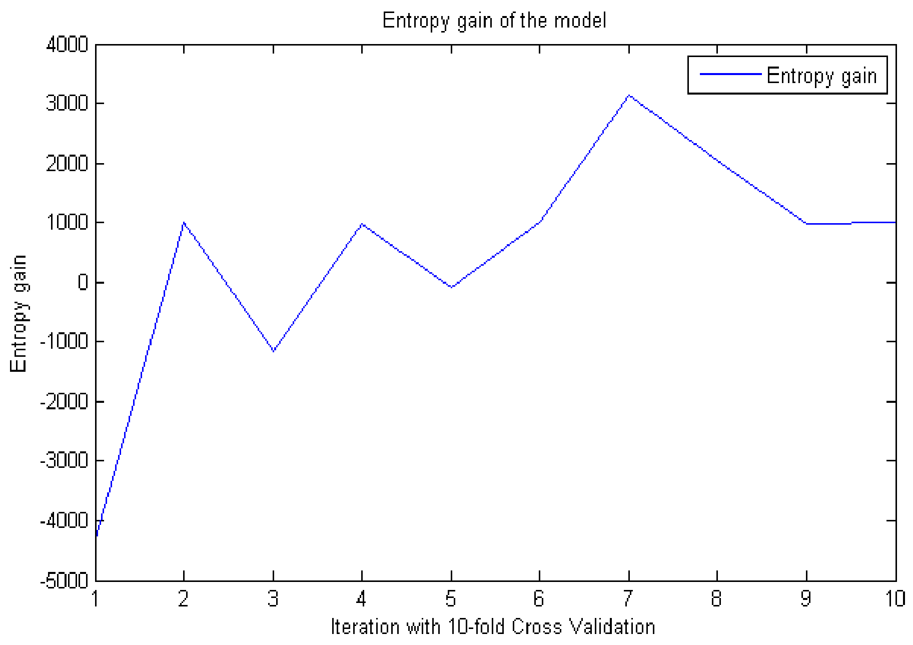

**Figure 6. Entropy Gain of the Model**

A cost and benefit analysis of the model is presented here using Receiver Operating Characteristic (ROC) analysis [2, 26, 28]. The false positive rate (FPR) is plotted on the x-axis against the True Positive Rate (TPR) on the y-axis. The area above the curve shows a high performance by the model showing that it can be very effective in detecting intrusions. The ROC graph is shown in Figure 7. The curve is plotted using summary information in Table 7.

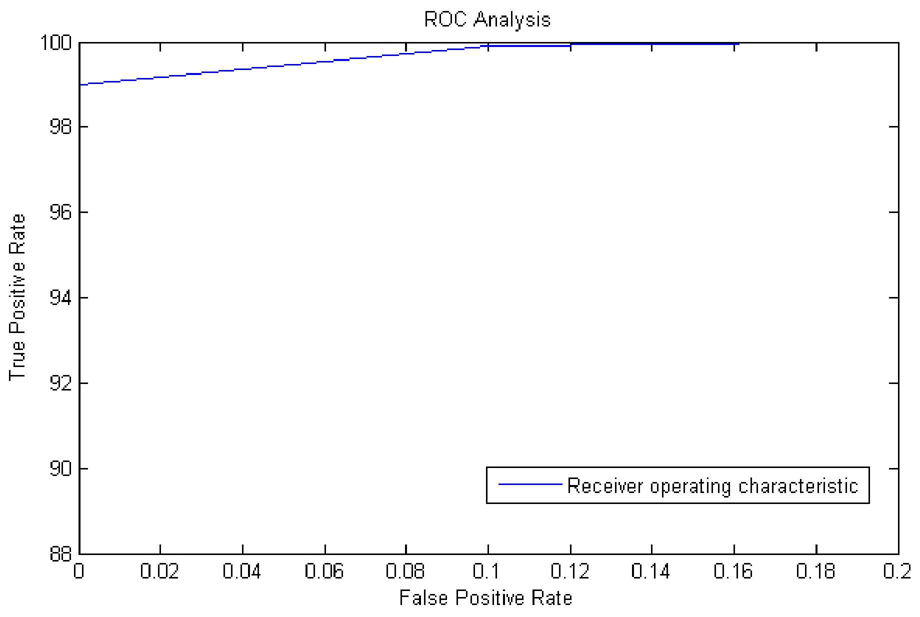

**Figure 7. Receiver Operating Characteristic (ROC) Analysis**

**Table 7. Summary of Model Performance for Four Types of Attacks**

| | DOS | PROBE | R2L | U2R |
|---|---|---|---|---|
| True Positive Rate | 100% | 99.9% | 99% | 88.9% |
| False Positive Rate | 0.2% | 0.1% | 0 | 0 |
| Precision Rate | 99.7% | 99.7% | 99.6% | 94,1% |
| Recall Rate | 100% | 99.9% | 99% | 88.9% |
| F-Measure | 99.8% | 99.8% | 99.3% | 91,4% |
| Detection Rate | 99.9% | 99.8% | 98.9% | 88.8% |

Table 7 gives details of the performance of the algorithm on four types of attacks and normal data.

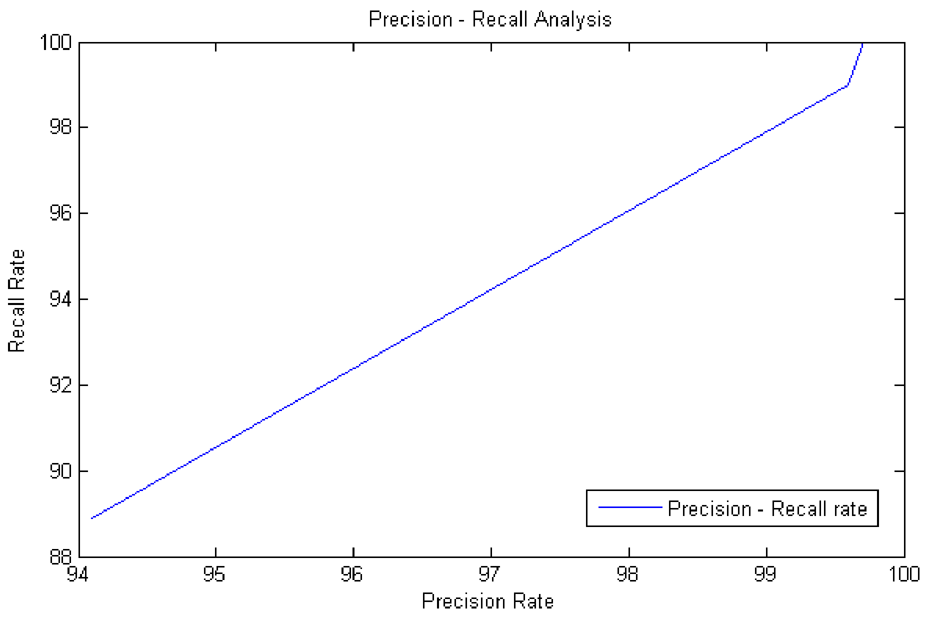

**Figure 8. Precision Recall Analysis**

Using the data in Table 7, the precision recall graph is plotted. This graph is used as a means of ascertaining the stability of the model in making predictions [2, 28]. As shown in Figure 8, the model shows evidence of good performance.

## 4.5. Comparison of Results

The results gotten from the various experiments conducted for the proposed algorithm will be compared to the algorithms proposed by others in terms of detection rate, false positive rate and accuracy. It is important to note that every system has its own limitations and no system can be 100% efficient. While some of the proposed systems may be efficient, they may increase the computational complexity of the target machine, which normally takes its toll on the machine's resources such as CPU and memory usage. The performance of the proposed technique was compared with the works of [29, 30, 31, 32, 33] respectively. The comparison results are shown in Table 8.

**Table 8. Performance Comparison of the Proposed System with Different Algorithms**

| Algorithm | Detection Rate | False Positive Rate |
|---|---|---|
| Proposed System – HYBRITQ-4 | 99.7% | 0.001% |
| Zhang, Zulkernine & Haque (2008) | 94.7% | 3% |
| Mingqiang, Hui & Qian, (2012) | 95.3% with k =8 | 2.08 |
| Yu & Dasgupta (2011) | 99.58 | 0.47 |
| Gaffer, Yahia & Ragab (2012) | - | 0.45% |
| Zhu & Zheng (2008) | 93.01% | 2.43% |

The comparison in Table 8 shows that HYBRITQ-4 can perform well in terms of detection rate and false positive rate in the detection of known and novel intrusions on a network.

## 5. Conclusion

This research's direction was geared towards enhancing the efficiency of intrusion detection without increasing the computational complexity of the system. This was achieved using the combined features of three algorithms (J48, Boyer Moore and K-NN) to propose a hybrid technique using both misuse and anomaly detection approaches. The HYBRITQ-4 algorithm performed well against four different attacks (Table 7) with high accuracy of detection and low false positive rate. We were able to demonstrate that the algorithm has the capability to detect known and novel intrusive patterns in network traffic. The findings from the results of experimentation have shown that on different iterations, the algorithm was stable and can provide significant benefits to intrusion detection through the initial preprocessing of packets based on protocol information. The preprocessing stage quickens the rule match process, thereby increasing the speed of detection as well as enhancing the overall performance of the system. Further research may be directed towards enhancing the training time of the algorithm when the feature space of connection increases during the building of the model. This will result in a very robust solution with an enhanced performance during the detection of intrusions in a network.

## References

[1] N. B. Idris and B. Shanmugam, "Artificial intelligence techniques applied to intrusion detection", INDICON, Annual IEEE, **(2005)**, pp. 52-55.

[2] M. Panda and M. R. Patra, "A comparative study of data mining algorithms for network intrusion detection", Emerging Trends in Engineering and Technology, ICETET, First International Conference, **(2008)**.

[3] T. S. Chou, J. Fan, S. Fan and K. Makki, "Ensemble of machine learning algorithms for intrusion detection", Systems, Man and Cybernetics, SMC, IEEE International Conference, **(2009)**.

[4] A. EshghiShargh, "Using artificial immune system on implementation of intrusion detection systems, in Computer Modeling and Simulation, EMS, Third UKSim European Symposium, **(2009)**.

[5] C. M. Ou, Y. T. Wang and C. R. Ou, "Intrusion detection systems adapted from agent-based artificial immune systems", Fuzzy Systems (FUZZ), IEEE International Conference, **(2011)**.

[6] J. Shun and H. A. Malki, "Network intrusion detection system using neural networks", Natural Computation, ICNC, Fourth International Conference, **(2008)**.

[7] L. X. Mei and Q. Zhi, "The application of hybrid neural network algorithms in intrusion detection system", E -Business and E -Government (ICEE), International Conference, **(2011)**.

[8] F. Li, "Hybrid neural network intrusion detection system using genetic algorithm", Multimedia Technology (ICMT), International Conference, **(2010)**.

[9] M. Xue and C. J. Zhu, "Applied research on data mining algorithm in network intrusion detection", Artificial Intelligence, JCAI, International Joint Conference, **(2009)**.

[10] V. K. Pachghare and P. Kulkarni, "Pattern based network security using decision trees and support vector machine", Electronics Computer Technology (ICECT), 3rd International Conference, **(2011)**.
[11] M. Gudadhe, P. Prasad and K. Wankhade, "A new data mining based network intrusion detection model", Computer and Communication Technology (ICCCT), International Conference, **(2010)**.
[12] S. Anithakumari and D. Chithraprasad, "An efficient pattern matching algorithm for intrusion detection systems", Advance Computing Conference, IACC, IEEE International, **(2009)**.
[13] R. K. Lenka and P. Ranjan, "A comparative study on DFA-based pattern matching for deep packet inspection", Computer and Communication Technology (ICCCT), Third International Conference, **(2012)**.
[14] X. Yong, H. F. Bao and Z. Y. Lai, "A descending suffix tree-based pattern matching algorithm for intrusion detection", Consumer Electronics, Communications and Networks (CECNet), 2nd International Conference, **(2012)**.
[15] A. Ojugo, A. Eboka, O. Okonta and F. Aghware, "Genetic Algorithm Rule-Based Intrusion Detection System (GAIDS)", Journal of Emerging Trends in Computing and Information Sciences, vol. 3, **(2012)**.
[16] P. A. D. Gomez and D. F. Hougen, "A Genetic Algorithm Approach for Doing Misuse Detection in Audit Trail Files", Computing, CIC, 15th International Conference, **(2006)**.
[17] S. M. Hussein, F. H. M. Ali and Z. Kasiran, "Evaluation effectiveness of hybrid IDS using snort with naïve bayes to detect attacks", Digital Information and Communication Technology and it's Applications (DICTAP), Second International Conference, **(2012)**.
[18] W. Y. Zhang, Q. B. Yang and Y. S. Geng, "A survey of anomaly detection methods in networks", Computer Network and Multimedia Technology, CNMT, International Symposium, **(2009)**.
[19] S. Kotsiantis, D. Kanellopoulos and P. Pintelas, "Data preprocessing for supervised leaning", International Journal of Computer Science, vol. 1, **(2006)**, pp. 111-117.
[20] R. Padmashani, S. Sathyadevan and D. Dath, "BSnort IPS better snort intrusion detection / prevention system", Intelligent Systems Design and Applications (ISDA), 12th International Conference, **(2012)**.
[21] V. Igure and R. Williams, "Taxonomies of attacks and vulnerabilities in computer systems", Communications Surveys & Tutorials, IEEE, vol. 10, **(2008)**, pp. 6-19.
[22] J. Wei, "Survey of network and computer attack taxonomy", Robotics and Applications (ISRA), IEEE Symposium, **(2012)**.
[23] M. Tavallaee, E. Bagheri, W. Lu and A. A. Ghorbani, "A detailed analysis of the KDD CUP 99 data set, in Computational Intelligence for Security and Defense Applications, CISDA, IEEE Symposium, **(2009)**.
[24] K. C. Khor, C. Y. Ting and S. Amnuaisuk, "A probabilistic approach for network intrusion detection", Modeling & Simulation, AICMS, Second Asia International Conference, **(2008)**.
[25] K. Q. Yan, S. C. Wang, S. S. Wang and C. W. Liu, "Hybrid intrusion detection system for enhancing the security of a cluster-based wireless sensor network", Computer Science and Information Technology (ICCSIT), 3rd IEEE International Conference, **(2010)**.
[26] A. Cufoglu, M. Lohi and K. Madani, "A comparative study of selected classifiers with classification accuracy in user profiling", Computer Science and Information Engineering, WRI World Congress, **(2009)**.
[27] N. S. Chandolikar and V. D. Nandavadekar, "Efficient algorithm for intrusion attack classification by analyzing KDD cup 99", Wireless and Optical Communications Networks (WOCN), Ninth International Conference, **(2012)**.
[28] T. Fawcett, "ROC graphs: Notes and practical considerations for researchers", Mach. Learning, vol. 31, **(2004)**, pp. 1-38.
[29] J. Zhang, M. Zulkernine and A. Haque, "Random-Forests-Based Network Intrusion Detection Systems", Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions, vol. 38, **(2008)**, pp. 649-659..
[30] Z. M. Qiang, H. Hui and W. Qian, "A graph-based clustering algorithm for anomaly intrusion detection", Computer Science & Education (ICCSE), 7th International Conference, **(2012)**.
[31] S. H. Yu and D. Dasgupta, "An effective network-based intrusion detection using conserved self pattern recognition algorithm augmented with near-deterministic detector generation", Computational Intelligence in Cyber Security (CICS), IEEE Symposium, **(2011)**.
[32] Y. C. Zhu and Y. Zheng, "Research on intrusion detection system based on pattern recognition", Networked Computing and Advanced Information Management, NCM, Fourth International Conference, **(2008)**.
[33] S. M. Gaffer, M. E. Yahia and K. Ragab, "Genetic fuzzy system for intrusion detection: Analysis of improving of multiclass classification accuracy using KDDCup-99 imbalance dataset", in Hybrid Intelligent Systems (HIS), 12th International Conference, **(2012)**.

## Authors

**Ayei Ibor**, he is currently a Lecturer in Computer Science at Cross River University of Technology, Nigeria. He has been involved in training and awareness programmes in computer and information security for a couple of years now. He has a Master of Science (M. Sc) degree in Computer Security and Forensics from the University of Bedfordshire, United Kingdom. With an experience of over eight years in computing ranging from programming to systems hardening, he is highly skilled in various areas of computing including systems administration, network security, programming/scripting, penetration testing, vulnerability management and ethical hacking. At the completion of his Master of Science degree, he won the Dean's Prize for the best overall performance in the Faculty of Creative Arts, Technologies and Science.

**Gregory Epiphaniou,** he has worked as a full time University Lecturer at the University of Bedfordshire, United Kingdom for four years. He is currently a Cyber Security Technical Consultant at QA Ltd, London, United Kingdom. He is involved in Cyber Security training in a variety of industrial and bespoke courses (CISMP, CISSP, CEH, Linux), for both public and private sector clients. He also teaches at the QA executive M.Sc in Cyber Security as an Associate Lecturer at Northumbria University He undertakes course development, public seminars / talks, taster days and a leading developer for the GCHQ summer school in Cyber Security.