Purging/Antirobustness as a Panacea for Attack Subversion and Autoimmunity on the Network

Olubadeji Bukola
Department of computer science
Federal University of Technology, Akure

Adetunmbi A. O.
Department of computer science
Federal University of Technology, Akure

ABSTRACT
Security of information is of utmost importance to any organization or individual, which depend on computer system or internet for business transaction or source of information or research. Many viruses are able to recognize certain anti-virus software, and respond differently to such software than to programs designed for other purposes. Some viruses go after the databases stored by anti-virus products. Some viruses simply go after anti-virus products, trying to erase them. Immune systems also face this daunting control challenge. On the one hand, there is need to minimize damage from pathogens, without wasting energy and resources, but on the other must avoid initiating or perpetuating autoimmune responses.

Several preventive measures including identification and authentication, logic access control, audit trails, digital signature and firewalls have been developed for the purpose of information security on system. As a result of inadequacies of these measures intrusion detection was introduced to complement these techniques and hence guarantee full protection of computing resources. Detection system is the process of identifying and detecting unauthorized access or abnormal incursions, actions and events in the system, which provides information for timely counter measures.

This paper presents a systematic approach to intrusion detection using machine learning techniques to purging in order to avoid autoimmunity on network. Machine learning is an automatic process of extracting hidden or interesting knowledge from data in order to generate its own rule based on the given set of data. In this paper rough set theory will be used as a mathematical tool to deal with imprecise and insufficient knowledge, finding hidden patterns in data and reduce dataset [12]. Appraisal of the shortcomings of the current intrusion detection systems (IDS) will be pointed out and the international knowledge discovery and data mining tools (KDD99) are used for benchmarking intrusion detection used, with the designing of rough-set model.

General Terms
Security, Signature pattern, Attack types

Keywords
Autoimmunity, Purging, Intrusion detection, Rough Set Theory

1. INTRODUCTION
Purging is one of the approaches reviewed by [11] that can facilitate strategic robustness. It can be define as a systematically and permanently removal of old and unneeded data, purging is stronger than delete. It is often possible to regain deleted objects by undeleting them, but purged are gone forever, Purging is only effective when individual replication rates are sufficiently large to tolerate the effects of removal of defective components. Thus apoptosis (programmed cell death) is a common strategy for eliminating cells upon damage to their genomes or upon infection, provided these cell types are capable of regeneration. Nerve cells and germ cells produce factors which strongly inhibit apoptosis and removal, in these cases has deleterious consequences. In severe infection, it can make sense to purge nerve cells. Cytotoxic T cells of the vertebrate adaptive immune system use an anti-robust strategy to deal with pathogens. Responding to signs of trouble by purging the damaged or infected cells, rather than by trying to stabilize these cells to help them live with the problem. This can be an effective way of dealing with a threat that otherwise can propagate, and anti-robustness at the cellular level confers robustness at the (multicellular) organismal level [12].

Physiologists consider the purpose, function or goal of a biological structure when trying to understand how that structure works. Immunologists do the same thing. The goal of any immune system is to protect against pathogens and these systems have therefore evolved to increase the fitness of the organism by reducing the damage caused by such organisms [13], ideally without wasting energy and resources [15]. To use this functional approach successfully, one must account for the tradeoffs and constraints that organisms face. Here, focus is on autoimmunity that has been instrumental in immune evolution: Autoimmunity: immune systems need to minimize the risk of autoimmunity. A single autoimmune mistake is potentially lethal, if directed against essential components of the body, they need to work in ways that rapidly evolving pathogens cannot exploit, subvert, or sabotage. As dependency on internet is on the increase on daily basis either for business transactions, source of information, research, and so on, so also is attack by intruders who exploit flaws in internet architecture and protocols, operating system to carry out their nefarious activities.

The possible actions of Autoimmune computer virus negative actions could be severed as it could lead to total grounding of the network, rendering the network inoperable for a period of time, gaining access to sensitive and confidential information, thereby causing disruption of business transactions and research projects, Deletion of non-viral files from the file-system, adding their fingerprint in the database file file e.g. an antivirus treating as infected files beginning with string 'MZ' will delete all .EXE files. Allowing viruses to spread, removing their fingerprint from database file: prevents detection of viruses that otherwise would be detected. Enabling a perfect virus time-bomb, the virus silently floods the net, undetected, activating itself at a given time, defacing web pages and stealing and vandalism of system resources.

The development of computer networking has changed the stand-alone pattern of computing, but it has also increased the risk and opportunity of network intrusion. The design of secure measures to prevent unauthorized accesses to resources and data of systems becomes a very important issue in the network security domain. Network security and intrusion

detection systems are one of the key research areas in the networking era as the most difficult problem today is how to deal with and rely on the huge volume of information that flows across the network while many network attacks are being reported every day. At present, it is impossible to completely eliminate the occurrences of security events, and what security faculty can do is to try their best to discover intrusions and intrusion attempts so as to take effective measures to patch the vulnerabilities and restore systems.

This brought about intrusion detection (ID) and intrusion detection system (IDS). Intrusion is defined as any set of action that attempt to compromise the integrity, confidentiality or availability of system resources [1]. Intrusion detection is defined to be the problem of identifying individuals who are using a computer system without authorization (i.e., crackers) and those who have legitimate access to the system but are exceeding their privileges (i.e., the insider threat). Intrusion detection systems (IDSs) are deployed to protect the computer infrastructures. The classical IDSs fall into two classes – anomaly based, and misuse based. An anomaly based IDS specify the normal behaviour of users or applications and consider any pattern falling outside the defined behaviour as an attack. A misuse based IDS specifies the signatures of attacks and parses audit files to detect any matches. The metrics for evaluating IDS are false alarms (false positives) and missed alarms (false negatives). Individual IDSs are often found to be unsatisfactory with respect to either or both of the metrics. For instance, anomaly based detection can generate many false positives since deviation from the specified normal behaviour is not necessarily an attack. Also, if the definition of normal behaviour is updated at runtime, an expert intruder can slowly change her behaviour to finally include it in the definition. This may give rise to a false negative. Misuse based detection can generate many missed alarms since for most practical open systems it is very difficult to define an exhaustive attack data base. IDSs are also classified as network-based or host-based in terms of source of data. The former collect raw network packets as the data source from the network and analyze for signs of intrusions Host-based IDS operates on information collected from within an individual computer system such as operating system audit trails, C2 audit logs, and System logs [18]; [4].

In recent years, the problem of network intrusion detection has attracted a lot of attention in the field of network security. Network intrusions carried out in various forms, such as worms, virus, spamming, Trojan horse, and many others, pose two major threats and damage on the victims. First, the intruders probe, gather, and deduce sensitive information about target hosts in an effort to gain unauthorized access to them and their networks. Second, the intruders inject unwanted packets into the target networks, aiming to disrupt the normal communications and services provided by the target networks. It is, therefore, critically important to implement effective network intrusion detection systems (NIDSs) to monitor the network and detect the intrusions in a timely manner.

One of the more interesting challenges for intrusion detection in a networked environment is to track users and objects (e.g., files) as they move across the network. For example, an intruder may use several different accounts on different machines during the course of an attack. Correlating data from several independent sources, including the network itself, can aid in recognizing this type of behavior and tracking an intruder to their source. In a networked environment, an intruder may often choose to employ the interconnectivity of the computers to hide his true identity and location.

Discovering intrusion in a network environment is costly, time consuming, and high risk activity. Over the years, researchers and designers have used many techniques to build different intrusion detection systems. Despite this, there have been one or more problems with present intrusion detection systems (IDS). Some major ones include [2]: high number of false positives and false negatives, and lack of efficiency.

The most important source of information for system administrator or site security officer (SSO) is the output of IDS, which automatically identify potential attacks and produce descriptive alerts. Due to the complicated nature of detecting actual intrusion, most current IDS place the burden of distinguishing an actual attack from a large set of false alerts on the system administrator or site security officer. Attempts made to secure the evolving computerized world led to developments of several prevention techniques to check intruder's activities. Some of these measures include identification and authentication, logical access control and digital signature and firewalls. However these measures are still the ultimate but a mirage for now coupled with the fact that there are flaws in internet protocols [16]. Building a completely secure system is difficult; even a secure system is vulnerable to abuse by insider who abuses their privileges.

The possibilities and opportunities on the internet are limitless; unfortunately, so too are the risks and chances of malicious intrusions. Two cases of security breaches were reported on the yahoo page 20th January, 2007; firstly, a university of Utah student was sentenced to four months imprisonment on the 18th January, 2007 for hacking into a university computer system to change his grade in December 2004. He used a software program to decrypt the password on the mathematics department's computer system and then found the professor's password. [6] Rightly said the interconnection of computers opens a new front for the attacker who can exploit the connection either to get the data being transferred or to penetrate one or more of the connected systems.

Hence, none of these preventive measures alone is sufficient to guarantee full protection of resources in computing environment and computer systems are most likely to remain unsecured for a while to come. Therefore intrusion detection is required to complement these preventive security measures to provide additional layer of protection. Intrusion detection is meant to identify and detect unauthorized access or abnormal phenomena, actions and events in the system, which provides important information for timely countermeasures. An intrusion detection system (IDS) is a computer program that attempt to perform intrusion detection by either misuse or anomaly detection or combination of techniques [5]. Intrusion detection is based on assumption that the behaviour of the intruder differs from that of legitimate user in ways that can be quantified. There is no exact distinction between an attack by an intruder and normal use of resources by an authorized user. There are some overlap behaviours, as illustrated in fig. 1 which gives an insight into the nature of the task confronting the designer of an intrusion detection system. Thus, a loose interpretation of intruder behaviour, which will catch more intruders, will lead to a number of false positives, or authorized users identifies as intruders. Effective intrusion detection must prevent damage from attack while avoiding subversion by attack and autoimmune mistakes.

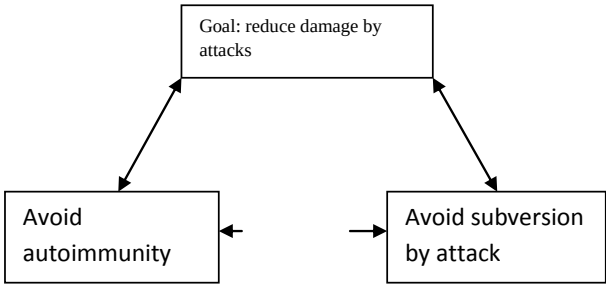

Fig. 1: Indicate that an effective intrusion detection system must prevent damage from attacks while avoiding subversion by attack and autoimmune mistakes. As indicated by the dashed arrow, efforts to avoid autoimmunity might compromise efforts to avoid subversion, and vice versa.

IDS should preferably perform its task in real time. Most intrusion detection in use today are either rule based or expert system based which strengths depend on the ability of the expert that codes it. Improved intrusion detection models based on machine learning were proposed, developed and implemented.

2. RELATED WORKS

The first generation of Intrusion Detection Systems captured network information/data into textual files. As these textual files grow, they become quite enormous and complex; making manual analysis cumbersome and unfeasible, usually resulting to undetected attacks and false alarms. Majority of the IDS used today are either rule-based or expert-system based. Their strengths depend largely on the ability of the security personnel that develops them. The former can only detect known attack types and the latter is prone to generation of false positive alarms [1]. Several researchers have addressed the problem of false alarms and missed alarms with traditional IDSs which are classified as anomaly-based and misuse-based. Also, traditional IDSs often generate a very large number of alerts for practical attack scenarios. The alarms correspond to elementary goals of the attack being realized. This large volume of alarms makes it difficult for a system administrator or even an automated intrusion response system to take appropriate actions. To counteract this problem, several researchers have developed alert correlation methods to construct attack scenarios. One class of techniques combines alerts based on similarity of certain alert attributes [3]. For example, in [3], source and destination IP addresses and ports are used for determining similarity and graphs are drawn with links between related alerts. However, this class misses out on correlating a large set of related alerts. A second class of techniques by [7] and [8] uses training set data to determine relations between alerts.

A computer attack is any malicious activity directed at a computer system or the services it provides. Computer attacks includes virus, Trojan horse, worm unauthorized access to system probing of system to gather information or a physical attack against computer hardware among others. Today more systems are attacked by intruders due to increased connectivity, especially on the internet, and vast spectrum of financial possibilities that are opening up these subversions attempt trying to exploit flaws in the operating system, internet protocols as well as in application programs [18].

[10] Describes taxonomy of attacks, grouping them into four major categories.

Probe: this scans a potential target (network) to gather information or find known vulnerabilities. These are usually harmless and common unless vulnerability is discovered and later exploited.

Denial of service (DOS): this type of attack which prevent normal operation, such as causing the target host or server to crash, or slow down networks operations after being congested by frivolous packets.

Remote to root local (R2L): this type of attack in which an authorized user is able to by pass normal authentication and execute commands on the target.

User To Root(U2R): this is an attack in which a user with login access is able to by pass normal authentication to gain the privileges of another user or exploit vulnerability to gain root access to the system.

Table below shows the different attack types for both training (known) and the additional attack types included for testing (novel) for the four categories.

Table 1: Known and novel attack types

DOS	Probe	R2L	U2R
Known			
Back, land, Neptune, Pod, smurf, teardrop	ipsweep, satan, nmap, portsweep	ftp_write, guess_passwd, warezmaster, warezclient, imap, phf, spy, multihop	rootkit, load module, buffer_ overflow, perl
Novel			
apache2, udpstorm, processtable, mailbomb	Saint, mscan	named, xlock, sendmail, xsnoop, worm, snmpgetattack, snmpguess	xterm, p.s., sqlattack, httptunnel

Monitoring for and responding to security incidents in large-scale, complex enterprise networks require a new approach to security incident management. Security reports indicating a policy violation come from a heterogeneous collection of components, such as intrusion detection sensors, firewall access policy violations or anomalous network traffic loads. Protecting against attacks currently in progress or eliminating a new vulnerability involves the reconfiguration of several different types of devices, such as firewalls, border gateways, software updates, and even host-based wrappers.

The challenge is to collect all information from the numerous data sources and to decide on appropriate actions for each reactive component. Simply forwarding all reports to a central location will not scale to large networks. Local decision making, however, may lack the global view necessary to thwart large-scale attacks. Defending against worms, particularly day-zero worms, is perhaps the most pressing challenge for a large enterprise. Such a worm can have a devastating impact as it automatically propagates itself to all vulnerable machines on a network. Defending against worm attacks, for which no pre-existing attack signature is available, requires the automation of tasks that current system administrators must perform manually. These include: automatic aggregation and correlation of security reports to detect activity at a local site, automated short-term defensive actions to stop local worm infections, cooperative alert sharing across administrative boundaries to protect sites not

yet infected, and automated back-off when a worm is contained or in the event of a false alarm.

3. DESIGN METHODOLOGY

Rough set theory (RST) approach will be used for detecting intrusions in this paper, RST is a useful mathematical tool to deal with imprecise and insufficient knowledge, find hidden pattern data, and reduce dataset size [14]. Rough set is an extension of conventional set theory (fundamental for all mathematics) which supports approximations in decision making. The RST concept is based on pair of conventional sets called lower and upper approximation. The lower approximation is a description of objects which are known in certainty to belong to the subject of interest, while upper approximation is a description of objects which possibly belongs to the subset.

Table 2: Example of intrusion data

S/N	Protocol	Service	Flag	Category
S1	Tcp	http	SF	normal
S2	Tcp	http	REJ	intrusion
S3	Tcp	telnet	REJ	intrusion
S4	Udp	ftp	REJ	normal
S5	Udp	telnet	SF	normal
S6	Tcp	ftp	REJ	normal

Here, the table 2 consists of three conditional features (protocol, service and flag), one decision feature (class) and six objects. Each row within the information table represents a se or an object. Every column represents an attribute that can easured for each object. The task of feature selection here is to choose the smallest subset of these conditional features so that the resulting reduced dataset remains consistent with respect to the decision feature. A dataset is said to be consistent if for every set of objects whose attribute valves are the same, the corresponding decision attributes are identical.

3.1 A Predictive Model for Detecting Attack Intrusions

Machine learning techniques, known for their significant contributions in classification model, are suitable for intrusion detection because IDSs simply model the normal behaviour of a system or store signatures of a well-known attacks patterns in order to classify network traffic or user behaviour on a computer or network either as normal or intrusive. An IDS simply acts as a security cameras and burglar alarm that identify and inform the security officer or the network administrator of any security breaches on the system.

The processes of building a predictive model capable of distinguishing between normal and abnormal signals (called intrusions or attacks) for intrusion detection using data mining techniques are divided into four major phases as illustrated in fig.2 these phases are capturing of network packets, processing of network data into suitable input format (like an information system), data normalization and extraction, and rules generation for intrusion detection which constitutes the training phase of our approach.

An IDS is being considered as a decision table, S = {U, A, V, F}, let $S = \langle U, A, V, f \rangle$ where U is a universe containing a finite set of N objects $\{x_1, x_2 \dots \dots x_N\}$. A is a non-empty finite set of attributes used description of objects. V describes valves of all attributes, which is $V = \bigcup_{a \in A} V_a$ where V_a forms a set of values of the a-th attribute. $f: \bigcup xA \rightarrow V$ is the total decision function such that $f(x,a) \in V_a \; for \; every \; a \in A \; and \; x \in \bigcup$. Information system is referred to as decision table (DT) if attribute in S is divided into disjoint sets called condition (C) and decision attributes (D) where A = C ∪ D and C ∩ D = ∅ , with A = C∪D, C ∩ D = ∅ and D = {normal, attack | attack = DoS, probe, R2I and U2r}. V consists of valves of connection features, U is the finite set of connections in the table and A is the finite set of connection features. Denoting normal and attack classes as D_{normal} and $D_{attack,}$ which is a partition of U by the decision attribute D. Based on rough set concept, the lower approximation of both normal and attack are given as $\underline{A}D_{normal} \; and \; \underline{A}D_{attack.}$

Hence, the positive region given as: $POS_C(D) = \underline{A}D_{normal} \cup \underline{A}D_{attack}$ contains only those connections that belong to either of the classes but not. Degree of dependency of any given features $\square \in C$ based on information on relation IND(D) is given as:

$$\gamma_B(D) = \frac{|POS_B(D)|}{|U|} \qquad 1$$

The TreeReduct rough algorithm will be used to induce detection rules for both normal and attack types. Rules generated using this algorithm is then used to classify and evaluate the performance of the TreeReduct algorithm on known and novel attacks.

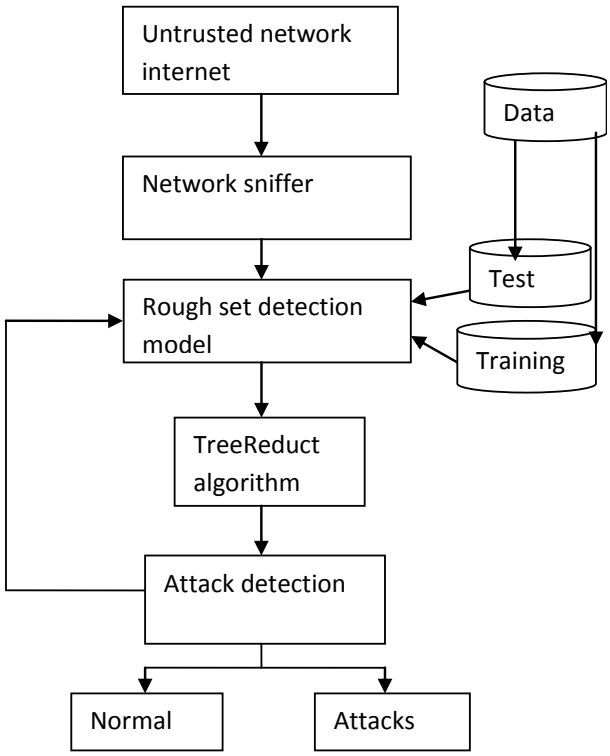

Fig. 2: System Architecture of intrusion Detection Based on Rough Set Theory

The system model comprises of two main phases: training and testing. In the training phase system builds a model using the training data to give maximum generalization accuracy (accuracy on unseen data). To find out the intrusion in testing phase, the constructed model detects the test data. The propose algorithm operates thus:

The training phase is carried out in the following order: given a dataset in relational form is normalized based on Shanon entropy described in section as pre-processing, duplicated records are removed from the dataset and induced rules for classification for both normal and attack. For testing phase, the rules induced by treeReduct algorithm to measure its

performance against known (variations from different network used for training) and novel attack for each attack type. Also, two different intrusion detection classification models are presented; the first is a single classification model for all the four attacks and normal while the second is three different models for classification of attacks types into DOS, Probe and R2L, U2R using LEM2 and TreeReduct algorithm.

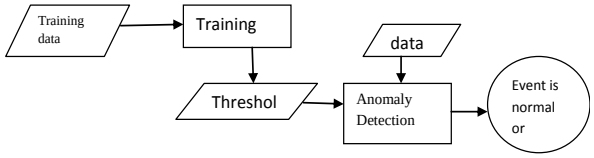

Fig.3: Data flow of the Anomaly Detection System

The anomaly detection system uses a training process to derive thresholds from the training data, and detects an event as normal or abnormal (as shown in figure 3).

3.2 Pre-processing

Rough set has the ability of building a good classifier even on small training dataset but work only on discretized data. Since, real life data is made of both or either continuous and discrete attribute valves, then the need for discretization before training commence. Discretization can be defined as set of cuts over domains of attributes, representing an important pre-processing task for numeric data analysis. For instance, KDD'99 dataset the numerical (continuous) attributes in dataset are discretized based on Entropy, a supervised splitting technique exploring class distribution information in its calculation and determination of split-point. Entropy discretization technique leads to reduction of data size and makes use of class information, which may assist in improving classification accuracy. In discretizing a numerical attribute A, the value of A with the minimum entropy value is selected as split-point, and the resulting intervals are recursively partitions to arrive at a hierarchical discretization computer as follows [9].

1. Given D consisting of data tuples defined by a set attributes and a class label attribute

A split-point for A can partition the tuples in D into two subsets satisfying the conditions

A ≤ split point and A > split point respectively, thereby creating a binary discretization.

The expected information requirement for classifying a tuple in D based on partitioning by A is given by

$$Info_A(D) = \frac{|D_1|}{|D|} Entropy(D_1) + \frac{|D_2|}{|D|} (D_2) \quad 2$$

Where D_1 and D_2 correspond to the tuples in D satisfying the conditions A ≤ split point and A > split point respectively. |D| is the number of tuples in D.

The entropy function for a given set is computed based on the class distribution of the tuples in the set. For example, given n classes, $C_1, C_2, ...C_n$, the entropy of D_1 is

$$Entropy(D_1) = \sum_{i=1}^{n} P_1 log_2(p_1) \quad 3$$

Where P_1 is the probability of C_1 D_1, determined by dividing the number of tuples of C_1 in D_1 by $|D_1|$, the total number of \tuples in D_1. Hence, in selecting a split-point for attribute A, the chosen attribute is the one with attribute value that gives the minimum expected information required (i.e. min ($Info^A(D)$). The process of determining a split-point is recursively applied to each partition obtained until the information requirement is less than a small threshold $\varepsilon(0)$.

4. EXPERIMENTAL SETUP

The training set employed for this analysis is the "10% KDD" (kddcup_data_gz.file) dataset, the continuous features are discretized based on entropy, a supervised discretization technique discussed in fig. 2.

For this experiment a total of 310,782 data were collected, used for both normal and attack type. Thus attack amount to 250,193 which is (60%) and normal is 60,593 which is (40%); out of the data collected 186,472 were used for training while the remaining 124,315 were used for testing.

Prior to the discretization, redundant records from the dataset were removed which is 4264 since rough set theory does not require duplicate instances to classify and identify discriminating features.

Hence for our experiment we confined our search space to only four important fields, we selected these field based on heuristic analysis of training data to identify potential fields that seemed unique to a particular attack type. These fields provide information on destination and source address, type of service, flag, duration of connection, number of byte sent, number of access control files and number of outbound commands in an ftp control session features of the connection. From our analysis of the training data, these fields appear to be amongst the decisive fields that can help identify an attack from a normal connection. From the above experiment, we were to create a rule that could successfully classify smurf, ping of death, mail bomb, synflood, back, and land type of all from the DOS class of attacks in the connection. Both training and testing data, there is little false positives which is what we are trying to minimize as much as possible.

4.1 Result Discussions

The feasibility of this approach was demonstrated on the KDD cup intrusion detection benchmark dataset earlier discussed. A total of 310,782 records were used for the experiment out of which 186,472 records randomly selected form the training dataset constituting 60.06% of the entire records used for experimental purpose. While 124,312 (39.94%) records were used mainly for testing. The training records were randomly selected from the 10% KDD dataset for training made of 22 attack types while the records used for testing are randomly selected from the test data set consisting of 39 attack types.

All the data types earlier mentioned are simply grouped as attack and the second category is simply named normal. Preprocessing is grouped into three steps. In the first step, categorical features like protocol_type (3 different symbols tcp, udp,icmp), Service (66 different symbols), and flag (11 different symbols) were mapped to integer values ranging from 1 to N where N is the total number of symbol variation in each feature. In the second step, continuous-value attributes like duration, src_bytes, dst_bytes are standardized based on entropy earlier discussed. Appendix 1 shows the cutoff points of entropy on continuous attributes and the mapping obtained on the discretized dataset.

After preprocessing, instances of duplicated records were removed from the training dataset. A total of 4264 records set

made up of 3188 normal and 1076 attack were actually used for training and in obtaining the signature patterns of attack.

Attack signature patterns are obtained based on comparison of features in each network connection with the class label normal with that of attacks. Results are presented in terms of variation(s) per attribute that achieved good levels of discrimination of normal from attack. This clearly distinguished a particular class label in the training data set. This can easily be achieved by generating the frequency of each variation per attribute against each class – normal and attack. The variation per attributes, Table 5 shows the signature pattern of attacks obtained from the training dataset and a total of 12 attributes out of 41 presented for training are chosen. Figures 4 and 5 show the dependency on the variations of attributes 4 and 5 to buttress or ratified the content of table 5. The pattern in figure 4 reveals attacks at variation 8 and 9 of attribute 4. The frequencies of normal at those variations are 0 while attacks are 276 and 92 respectively.

Table 3: Signature pattern of attacks obtained from the training set

s/n	Column selected	Variations (x)	Variation(x) Translation
1	1: duration	4; and 6	$585 < x \leq 712$ $717.5 < x \leq 899.5$
2	4: flag	8 9	RSTO
3	5: src_bytes	16	$19.5 < x \leq 27.5$
		23 25 6	$1031 < x \leq 1033.5$ $49080 < x \leq 132704$ $x > 882177$
4	6: dst_bytes	7	$142.5 < x \leq 144$
5	11: hot	1	$X > 2.5$
6	18: num_shells	1	$X > 0.5$
7	21: is_host_login	2	$X > 1$
8.	24: srv_count	25	$X > 419$
9	26: srv_serror_rate	1	$0.00499916 < x \leq 0.0149975$
10	30: diff_srv_rate	2	$0.0349961 < x \leq 0.104988$
11	37: dst_host_srv_diff_host_rate	2	$0.504944 < x \leq 0.634949$
12	41: dst_host_srv_rerror_rate	11	$0.824952 < x \leq 0.939942$

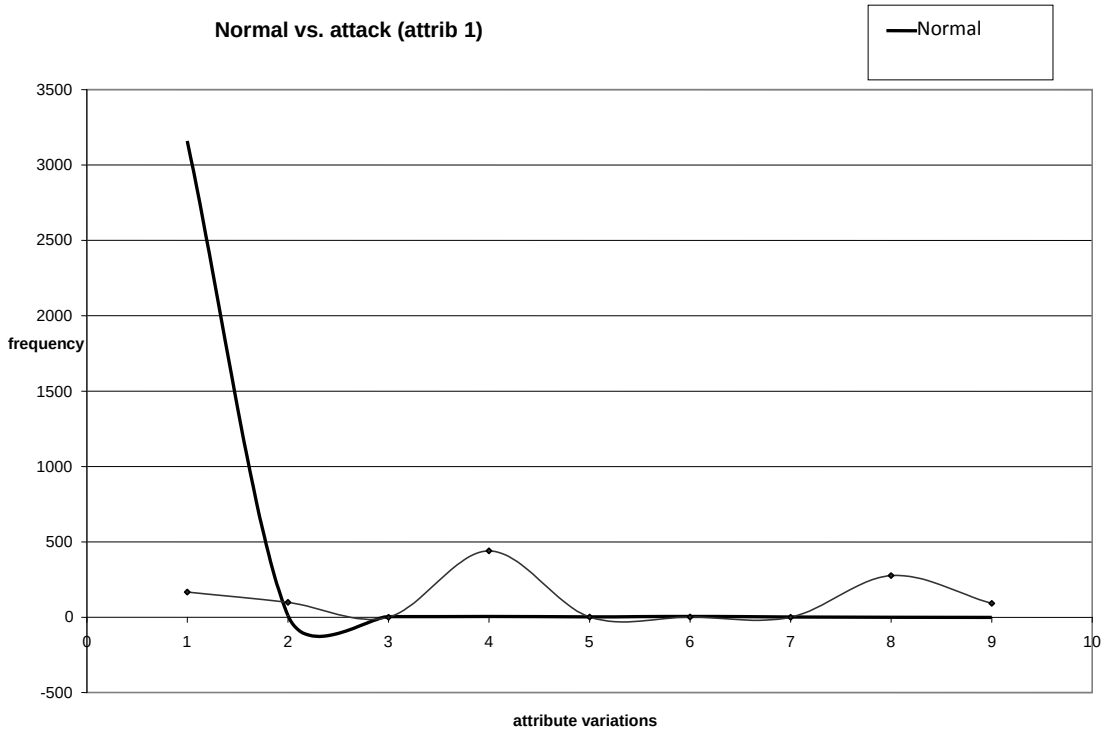

Fig. 4: Attribute 1 variation dependency of normal and attacks

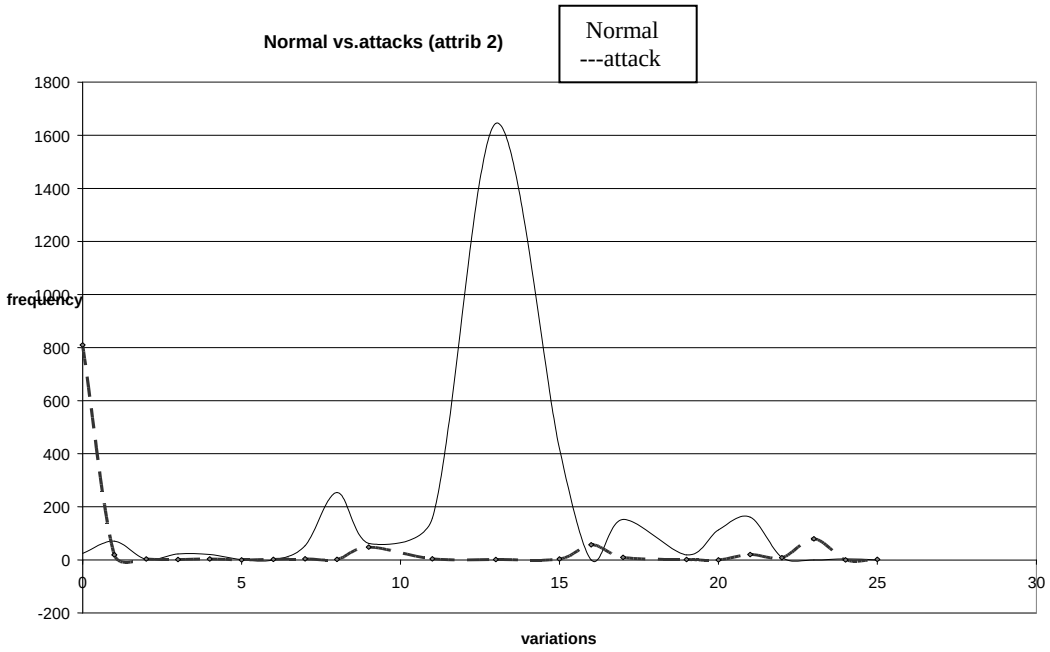

Fig. 5: Attribute 2 Variation Dependency of Normal and Attacks

The performance measures or accuracy is computed thus:

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN}, where$$

a. True Positives (TP), the number of normal is correctly classified.
b. True Negatives (TN), the number of attack is correctly classified.
c. False positive (FP), the number of normal classified as attack.
d. False Negative (FN), the number of attack falsely classified as normal

Table 4: Classification obtained from Training dataset

Predicted as Actual	Normal	Attack
Normal(38354)	36354(100.00%)	0(0.00%)
Attack(150116)	41828(3.65%)	144642(96.35%)

$$Accuracy = \frac{36354 + 144642}{36354 + 144642 + 0 + 4}$$

$$= \frac{180996}{186470} = 0.9706 = 97.06\%$$

Table 5: Classification obtained from Test dataset

Predicted as Actual	Normal	Attack
Normal(24235)	24198(99.85%)	37(0.15%)
Attacks(100077)	41117(33.17%)	83196(66.93%)

$$Accuracy = \frac{24198 + 83196}{24198 + 41117 + 0 + 37} = \frac{107394}{124312} = 86.39\%$$

Tables 4and 5 show the confusion matrix obtained using the obtained signature of attack on training and test datasets. The computed accuracy obtained on the training and testing are satisfactory and thus shows that it is a promising approach.

5. CONCLUSION

The need for effective and efficient security on our system cannot be over-emphasized. This position is strengthened by the degree of human dependency on computer systems and electronic superhighway (internet) which grows in size and complexity on daily basis for business transaction, source of information or research. In this paper a framework that facilitates and systematic construction of adaptable and extensible intrusion detection system was presented. TreeReduct, a predictive rough set based algorithm for mining unknown patterns for automated models for intrusion detection was used. Comparisons of the performances of the classifiers using kdd'99 intrusion detection evaluation dataset were drawn based on the group the classifier belongs (supervised or unsupervised). The method of improving intrusion detection system based on machine learning techniques were described and implemented using C++ programming language.

From the experiment, the performances of the rough set based on treereduct algorithm on the training and testing sets are adjudged good as the accuracy stood at 97.06% and 86.39% respectively. Also, the treereduct algorithm seems easier in obtaining effective signature patterns for classifying network traffic. This method could as well be employed in obtaining virus signatures and in other classifying problems. The results of the developed tools are satisfactory though it can be improved upon as it presently detects only known signatures. These tools will go a long way in alleviating the problems of security of data on computing systems.

5.1 Future scope of the research

Integrating IDSs with Network Management Systems Intrusion detection should be integrated with a network management system. A lot of network anomalies can be filtered by a network management system first because this is part of its function. On the other hand, when detecting an intrusion, the IDS can communicate the network management system to take appropriate actions, e.g., re-route some services from compromised host.

Anomaly detection: More study is required on anomaly detection by mimicking human immune system, to know how the immune system is able to control false positive, attack subversion and autoimmunity in the body. The human immune system in detection does a lot of validation on suspected attack before deeming them confirmed as an attack. Given the wide range of normal behavior of the services and hosts in a network, it is very likely that a detected anomaly is actually normal (i.e. legitimate).

6. REFERENCES

[1] Adetunmbi A. O., Falaki S. O., Adewale O. S. and Alese B. K. (2008). Network Intrusion Detection Based on Rough Set and K-Nearest Neighbour. International Journal of Computing and ICT Research, 2(1).

[2] Anderson, J.P.(1980). Computer Security threat monitoring and surveillance, Technical report, James P. Anderson co. Box 42, Fort Washington.

[3] Apache (2002). "Apache Chunk Buffer Overflow Attack".

[4] Byunghae-Cha, K.P. And Jaittyun, S. (2005). Neural Networks Techniques for Host anomaly Intrusion Detection using Fixed Pattern Transformation. ICCSA 2005, LNCS 3481, 254-263

[5] Balasubramanuyan, J.S, Garcia-Fernadez, J.O., Isaacoff, D., Spafford, E., and Amboni,D.(1998) An architecture for intrusion detection using autonomous agent 14th Annual computer security conference(ACSAC''98), pages 13-24' IEEE. Computer society.

[6] Cobb, S. (1996) The NASA GUIDE TO PC AND LAN security, McGraw-Hill Book Company

[7] Cuppens F. and Ortalo R. (2000). "LAMBDA: A Language to Model a Database for Detection of Attacks", In Proceedings of the Third International Workshop on the Recent Advances in Intrusion Detection (RAID'2000),Toulouse, France.

[8] Dacier M, Deswarte Y and Kaâniche M. (1996). "Models and Tools for Quantitative Assessment of Operational Security", in 12th International Information Security Conference (IFIP/SEC'96), (S.K. Katsikas and D. Gritzalis,Eds.), 177-186, Chapman & Hall, Samos (Greece).

[9] Jiawei and Micheline, K.(2006) Data Mining: Concepts and techniques, second edition, Elsevier inc.

[10] Kendall, K. (1999) A database of computer attacks for the evaluation of intrusion detection systems, Masters thesis, Massachusetts institute of technology, USA.

[11] Krakauer, D.C. and Plotkin, J.B. (2005) Principles and parameters of molecular robustness. In Robust Design: A Repertoire of Biological, Ecological, and Engineering Case Studies (Jen, E., ed.), pp. 71–103, Oxford University Press

[12] Krakauer, D.C. (2005) Robustness in biological systems: a provisional taxonomy. In Complex Systems Science in Biomedicine (Deisboeck, T.S. and Kresh, Y., eds), pp. 185–207, Plenum

[13] Matzinger, P. (1998) An innate sense of danger. Semin. Immunol. 10, 399–415

[14] Pawlak, Z. (1982) Rough sets: international journal of computer and information science, vol. 11. No.5, pp. 341 - 356. 1982

[15] Schmidt-Hempel, P. (2005) The evolutionary ecology of insect immune defense. Annu. Rev. Immunol. 50, 529–551

[16] Slagel, M. (2001) the design and implementation of MAIDS (Mobile Agent for Intrusion Detection System) Master's creative Component paper, Iowa state university, Ames Iowa.

[17] Sundaram, A. (1996). An Introduction to Intrusion detection.

[18] Spafford, E. (1989): internet worm program: an analysis ACM Communication Review, 19(1) 17-57.