Int. J. Advanced Networking and Applications
Volume: 12 Issue: 01 Pages: 4519-4527(2020) ISSN: 0975-0290

Prototype Intelligent Log-based Intrusion Detection System

Gitau Joseph M.
School of Informatics and Innovative Systems, Jaramogi Oginga Odinga University of Science and Technology, Bondo , Kenya
Email: jkibasui@gmail.com
Rodrigues Anthony.J.
School of Informatics and Innovative Systems, Jaramogi Oginga Odinga University of Science and Technology, Bondo , Kenya
Email: tonyr@jooust.ac.ke
Abuonji Paul
School of Informatics and Innovative Systems, Jaramogi Oginga Odinga University of Science and Technology, Bondo , Kenya
Email: pabuonji@jooust.ac.ke

ABSTRACT

The maintenance of web server security is a daunting task today. Threats arise from hardware failures, software flaws, tentative probing and worst of all malicious attacks. Analysing server logs to detect suspicious activities is regarded as a key form of defence, however, their sheer size makes human log analysis challenging. Additionally, traditional intrusion detection systems rely on methods based on pattern-matching techniques which are not sustainable given the high rates at which new attack techniques are launched every day. The aim of this paper is to develop a proto-type intelligent log based intrusion detection system that can detect known and unknown intrusions automatically. Under a data mining framework, the intrusion detection system is trained with unsupervised learning algorithms specifically the *k-means* algorithm and the *One Class SVM* (Support Vector Machine) algorithm. The development of the prototype system is limited to machine generated logs due to lack of real access log files. However, the system's development and implementation proved to be up to 85% accurate in detecting anomalous log patterns within the test logs.

Keywords*:* **prototype, intrusion detection, log-based, data mining.**

Date of Submission: June 05, 2020 Date of Acceptance: July 06, 2020

I. INTRODUCTION

In this advancing and fast developing field, technology has become cheaper, easier to develop, and deploy. On the other hand, this has also made probing and attacking servers cheaper and easier to carryout. It is therefore vital to ensure that web servers are alert and hence secured against any form of attack.

Server logs have been used to confront: failure either hardware or software and record notices, warnings and errors to ensure that system administrators can recover or at least know what caused a system failure event. Log recording has also acted as a form of defense against human attacks where predetermined techniques such as SQL injections can be easily identified.

An average web server receiving traffic of at least 1000 unique visits a day generates a huge log that cannot be analyzed manually. The same web server placed in a large company would receive over 10000 unique visits a day. The sheer size of the log file would be practically impossible to inspect by most system administrators. Most log based intrusion detection systems on the market are pattern-matching technique based, that is, they compare the log entries to a set of predefined patterns that had been manually updated by security experts [1,5, 26]. Though this approach is effective in determining attacks of known patterns, the drawback is that for each new attack the system is defenseless and it takes security experts much time and effort to update the new patterns to the intrusion detection system [14, 15].

From this perspective, current intrusion detection systems are far from intelligent in that they exclusively rely on human intervention to operate effectively and thus, more advanced intrusion detection systems are desirable. These systems should be capable of detecting known and unknown intrusions intelligently and automatically distinguishing normal network activities from those abnormal and possibly malicious ones without or with minimum human intervention.

Some data mining algorithms applied to log based intrusion detection systems came up with an effective anomaly detection based intrusion detection system that relied on nothing more other than the inflowing stream of logs to determine what is normal and what is not (possibly an attack). Those algorithms are based on supervised learning. That is to say they are trained, other than being explicitly programmed, on data sets with labels indicating whether the instances are pre-classified as attacks or not. However, the techniques seemed cumbersome as manually labeling the large volumes of server data mostly over 1GB log files was expensive and difficult. This is what inspired the approach of unsupervised machine learning where no

labels are pre-set hence the system is left to determine what an attack is and what is not, see [11].

With no requirement for class labels, unsupervised learning algorithms seemed to solve this problem. A broad explanation of approach to intrusion detection systems is that when an intrusion detection system becomes "familiar" with the data through the unsupervised learning algorithms, it is likely to detect "abnormal" data when they come in many of which are malicious, see [1, 4].

This paper aims to develop and implement a system that can learn the normal state and nature of a web server where installed and dynamically identify anomalies bringing them to the system administrator's attention as follows:

i). Learning and Detection

Based on the unsupervised learning algorithm used, the system will be able to learn from the current states of the server (or when the server was working optimally). Detect the anomalies in the logs and hence alert the system administrator.

ii). Generality

Based on unsupervised learning, detecting abnormal activities shall be executed automatically without too much human intervention [18, 25, 27].

2.0 BACKGROUND

Designing an intelligent log based intrusion detection system involves the following:

A. 2.1 Intrusion

Threats to web servers come typically from the malfunction of hardware or software, or through malicious behaviour by users of software. Promptly resolving incidents is vital, considering the huge costs of data loss and server down-time. The abundance of computational resources makes lives of computer hackers easier. Without much effort, they can acquire detailed descriptions of system vulnerabilities and exploits to initiate attacks accordingly. According to statistics from [2], the most influential reporting centre for Internet security problems, show that there was a dramatic increase of reported network incidents to CERT/CC.

B. 2.2 Logs

To protect servers from attacks, a common approach is to record server logs to monitor all those prominent activities. Each time a noticeable event happens in the server, an entry will be appended to a log file, in the form of plain text or binary format. Take web log files as an example. Every "hit" to a web site, including requests for HTML pages as well as images, is logged as one line of text in a log file. This records information about who is visiting, where they are from and what they are doing with the web server. Below is a sample of an apache log format,

"%h %u %t \"%r\" %>s %b \"%{Referrer}i\" \"%{User-Agent}i\""

This translates to: -

%h – ip address

%u – Authenticated userID if http authenticated

%t – timestamp [day/month/year: hour: minute: second zone]

%r – request line (method_used requsted_resource protocol)

%>s – status code

%b size of returned obj

\"%{Referrer}i\" – http header referrer

\"%{User-Agent}i\"" – the user agent

Below are examples of apache server access logs

120.254.103.132 - - [14/Jan/2016:12:58:17 +0300] "GET /search?=IntelliIDS HTTP/1.0" 200 5057 "http://black-adkins.com/about/" "Mozilla/5.0 (Windows NT 6.1) AppleWebKit/5321 (KHTML, like Gecko) Chrome/14.0.824.0 Safari/5321"

36.194.62.124 - - [14/Jan/2016:13:24:30 +0300] "GET /productID=3257 HTTP/1.0" 200 4962 "http://www.hart.info/" "Mozilla/5.0 (Macintosh; PPC Mac OS X 10_8_7) AppleWebKit/5360 (KHTML, like Gecko) Chrome/14.0.896.0 Safari/5360"

An experienced system administrator may take a quick glance at web server logs and realize instantly what has happened. However, it is almost impossible for any normal person to check those logs when the log files have accumulated to thousands if not millions of log entries. Naturally, appropriate methods are needed to remove irrelevant information and extract the most salient. What is required, therefore, is an intrusion detection system that is intelligent enough to automatically detect those abnormal activities in the logs without too much human inputs.

C. 2.3 Intrusion Detection Methods

There have been several intrusion detection systems that use log analysis on the market. The intrusion detection methods used are categorized as follows, see [3, 28]:

i). Pattern Matching

This type of system examines the contents of network traffic (in real-time intrusion detection systems) or log file (in log based intrusion detection systems) to look for a sequence of bytes as the pattern to match. The approach is rigid but simple to implement and therefore widely used.

ii). *State-full Pattern Matching*

This performs pattern matching within the context of a whole data stream instead of just looking into current packets.

iii). Protocol Decode-Based Analysis

This makes extensions to the state-full pattern matching method in that it tries to find out the violations against the rules that are defined by the Internet standards.

iv). Heuristic-Based Analysis

Makes decisions based on pre-programmed algorithmic logic. Those algorithms are often the statistical evaluations of the network traffic content.

v). *Anomaly Detection*

This approach tries to find out anomalous actions based on the learning of its previous training experience with patterns assumed as normal.

The first four methods are widely used in industry practices. However, most of these pattern-matching based detectors can only deal with already-known intrusions that

have been recognized by the security experts. Unfortunately, ill-intentioned hackers are aware of those patterns too. When new attack patterns emerge, very likely they could evade the detection by deliberately avoiding those widely publicized matching patterns. The potential damages caused by those attacks are consequentially substantial.

With regard to attacks that become more cunning, more variant, and hence much more dangerous human-maintained, it would be difficult to update pattern-matching intrusion detection systems quickly enough to be effective. Data mining approaches, armed with machine learning algorithms, may provide the solution.

D. 2.4 Data Mining Approaches

Data Mining is defined as the analysis of (often large) observational data sets to find unsuspected relationships and to summarize the data in novel ways that are both understandable and useful to the data owner. During the process of data mining, many machine learning algorithms are available for choosing. Depending on whether the class labels are provided for learning, these machine learning algorithms can be classified as both supervised or unsupervised [10,13].

1) 2.4.1 Supervised learning

Trained with data bearing class labels indicating to which subcategories they belong or what real-valued properties they have, a supervised learning algorithm tries to predict the most likely labels for new test data. There are two major subcategories for supervised learning:

i). *Classification* is to predict the class membership as one of a finite number of discrete labels.

ii). *Regression* is to predict the output value as one of a potentially infinite set of real- valued points.

There are many widely used supervised classification techniques. They include but not limited to Support Vector Machines (SVMs), Decision Trees, Neural Networks, Naive Bayes, Nearest Neighbour and Regression models. For example, based on a Naive Bayes classifier, trained with a data set with virus labels on file headers, an automatic email filter that detects malicious Windows executables coming through the email system has been developed in the past [23].

2) 2.4.2 Unsupervised learning

In unsupervised learning, the data are not labelled, which makes it hard to tell what counts as good. The model generating the output must either be stochastic or must have an unknown and varying input in order to avoid producing the same output every time. From this point of view, the aim of unsupervised learning could be regarded as a generative model that gives a high likelihood to the observed data, [11].

From the perspective of machine learning, the searching for clusters is unsupervised learning. To perform clustering is to try to discover the inner nature of the data structure as a whole, and to divide the data into groups of similarity. From the viewpoint of data mining, clustering is the partitioning of a data set into groups so that the points in the group are similar as possible to each other and as different as possible from points in other groups. There are generally three types of clustering algorithms

i). **Partition-based clustering**

Given a predefined number of clusters, find the optimal partitions for each point. Choose the centres so as to minimize the summed distance. The *k*-means algorithm is a well-known example of this kind of clustering methods.

ii). **Hierarchical clustering**

Hierarchical clustering builds a cluster hierarchy. The hierarchy is a tree of clusters. Every node in the tree contains child clusters while sibling clusters share a common parent node. Depending on how the tree is formed, hierarchical clustering methods fall in two categories, agglomerative and divisive. Agglomerative methods recursively merge points while divisive methods start from a cluster of all data and then gradually split them into smaller clusters.

iii). **Probabilistic based clustering**

This approach assumes that the data comes from a multivariate and finite mixture model with probability as shown below

$$p(x) = \sum_{k-1}^{K} \pi_k f_k(x;\, \theta_k) \qquad eq.(1)$$

where π_k is the class component prior probability, $f_k(x: \theta_k)$ is class conditional density function, and θ_k is its model parameters.

E. 2.5 Novelty Detection

Novelty detection refers to the identification of new or unknown data or signal that a machine learning system is not aware of during training. It is one of the fundamental requirements of a good classification or identification system since some- times the test data contains information about objects that were not known at the time of model training.

Anomaly could be regarded as one kind of novelty. Normally, classifiers are expected to give reliable results when the test data are similar to those used during training. However, the real world is totally different, when abnormal data come in, picking them out is a problem. Compared to conventional 2-class classification problem, an anomaly detection system is trained with only normal patterns and then try to predict those abnormal data based solely on the models built from normal data. There exist a variety of methods of novelty detection that have been shown to perform well on different data sets [17].

1) 2.5.1 Probabilistic/GMM approaches

This category of approaches is based on statistical modelling of data and then estimating whether the test data come from the same distribution that generates the training data. First estimate the density function of the training data. By assuming the training data is normal, the probability that the test data belong to that class can be computed. A threshold can then be set to signal the novelty if the probability calculated is lower than that threshold.

For Gaussian Mixture Modelling (GMM) models, the parameters of the model are chosen by maximizing the log likelihood of the training data with respect to the model. This task could be done using re-estimation techniques such as EM algorithm. However, if the dimensionality of the data is high, a very large number of samples are needed to train the model, which makes the computation even harder [5].

It is simpler to just find the distance of test data from the class mean and set a threshold for the variance. If the test data is far away from the mean plus the variance threshold, then it can be claimed to be novel.

.

2) 2.5.2 Non-parametric approaches

For non-parametric methods, the overall form of the density function is estimated from the data as well as parameters of the model. Therefore, non-parametric methods do not require extensive prior knowledge of the problem and do not have to make assumptions on the form of data distribution, which means that they are more flexible though much more computational demanding.

i). K-nearest neighbour approaches

The *k*-nearest neighbour algorithm is another technique for estimating the density function of data. This technique does not require a smoothing parameter [20]. Instead, the width parameter is set as a result of the position of the data point in relation to other data points by considering the k-nearest data in the training set to the test data.

For novelty detection the distribution of normal vectors is described by a small number of spherical clusters placed by the *k*-nearest neighbour technique. Novelty is assessed by measuring the normalised distance of a test sample from the cluster centres [17].

ii). String matching approaches

String matching approaches is biologically inspired by studying how the immune system works [6].

Treating training data as templates, which are represented by a string (vector of features), they could then compute some measure of dissimilarity between training and test data. The self-data is converted to binary format forming a collection *S*. Then a large number of random strings are generated forming a set R_0. Strings from R_0 are matched against the strings in *S* and those that match are eliminated.

Since perfect matching is extremely rare, the matching criterion is relaxed so as to consider only *r* contiguous matches in the strings. Once R_0 is created, new patterns are converted to binary and matched against R_0. If a match is found, then new pattern belongs too non-self and is rejected. The major limitation appears to be the computational difficulty of generating the initial repertoire. This method has been applied on the detection of computer virus and claimed some good results.

3) 2.5.3 Neural network based approaches

Quite a number of different architectures of neural networks are applied to novelty detection. A neural network can detect novelty by setting a threshold on the output values of the network. Or it can calculate the Euclidean distance between output patterns and target patterns and throw those with highest distance out as the novelty [22].

.

F. 2.6 Feasibility Study

Efficacy of Log Analysis Based IDS => High
Practicality of Log Based IDS => High
Efficacy of AI in Anomaly Detection => High
Efficacy of Unsupervised Learning in Anomaly Detection => High
Top Rated Python Analytical and Statistics Library => SciKit-Learn
Most Favourable Unsupervised Learning Algorithms => *k means* and *One Class SVM*

3.0 Methodology

We now discuss how an intelligent network log analyzer can be built with a more in-depth approach on the theoretical aspects of the algorithms to be incorporated in the system. We introduce how the logs are vectorized, discuss briefly feature extraction, and system development methodology.

3.1 *K* means

The *k means* algorithm is cluster based [12], hence one needs to define the number of clusters, *k,* hence the name. The clusters are the average locations of all the members of a cluster.

If we assume *n data* points then $D = \{x_1, \ldots, x_n\}$

Hence to find K clusters $\{C_1, \ldots C_K\}$ the algorithm is describe below:

initialize $m_1 \ldots m_K$ through random selection as cluster centers
while (no conditions are met, mostly (lack of change in clusters C_K)
for $i = 1,\ldots,n$
calculate $|x_i - m_j|^2$ for each center
assign i to the closest center
end for loop
re-compute each m_j as the mean of the data points assigned to it
end while loop

The overall formula for *k means* is:

$$\sum_{i=0}^{n} \frac{min}{u_j \in C} (||x_j - u_i||.^2) \qquad eq(2)$$

3.2 *One Class SVM*

One class SVM attempts to learn the decision boundaries that achieve the maximum separation between the data points and the origin. The introduction of kernels in *one class SVM* gives it the ability to learn non-linear decision boundaries as well as account for outliers. *One class SVM* utilizes an implicit transformation function $\varphi(x\cdot)$ that is defined by the kernel chosen [16,19]. It then learns the decision boundary which separates most of the data from the origin. Data that lie outside the data points are considered as outliers [7].

By observing that all kernel entries are non-negative (≥0), all the data in the kernel space can be concluded as to belong in the same quadrant.

Assume $g(x\cdot)$ is defined as:

$g(x) = w^T \varphi(x) - \rho$ (1) eq(3)

where w is the perpendicular vector to the decision boundary and ρ is the bias term

Then,

$f(x) = sgn(g(x))$ eq(4)

shows the decision function that *one-class SVM* uses in order to identify normal points. The function returns a positive value for normal points and negative for outliers.

3.3 Text Vectorization and Feature Extraction

Since apache logs are in human readable text form, vectorization is required to convert them into numerals. The feature extraction and text vectorization techniques used in this system are

i). Frequency

A frequency matching function searches through the logs for unique instances of a specific row in each column and assigns each a numerical value.

ii). The Bag of Words representation

Count Vectorization from Scikit-learn: This tokenizes each unique word in the logs and assigns it a unique numeral value (Scikit-learn.org)

4.0 Development

The general purpose of IDS is to quickly identify attacks in the system. IntelliIDS was developed with this in mind hence it is developed to search for known attacks first and then use machine learning to detect unknown attack patterns

4.1 System Design

4.1.1 Experimental Approach IntelliIDS is an experimental framework. Given this, the best algorithm will be determined by the highest accuracy scores. This means that several detection methods will be used and the one with the highest scores will be implemented. For faster development, the system will be developed for console usage other than a GUI based approach. However, the system takes up multiple arguments which enhance its usage and allows for custom analysis. For example, the user has the option to choose the type of analysis, algorithm to use, etc.

4.1.2 Modularity

Module based approach will be used for the entire system hence more features can be added easily.

4.1.3 Rapid Development

Existing source code available [9] and public libraries will be used where necessary to hasten the development process.

4.2 Programming Languages

Since IntelliIDS is experimental and exploration oriented we use some of the best working supervised and unsupervised machine learning algorithms already available in popular programming languages such as python, R, C, C # etc. A thorough search reveals that the best programming language for machine learning is R (since it is data science based) however for the sake of expediency python was chosen for the sake of proficiency with regard to the prototype IntelliIDS.

4.3 Modules

Below is the description for each module in IntelliIDS. For more details see [8].

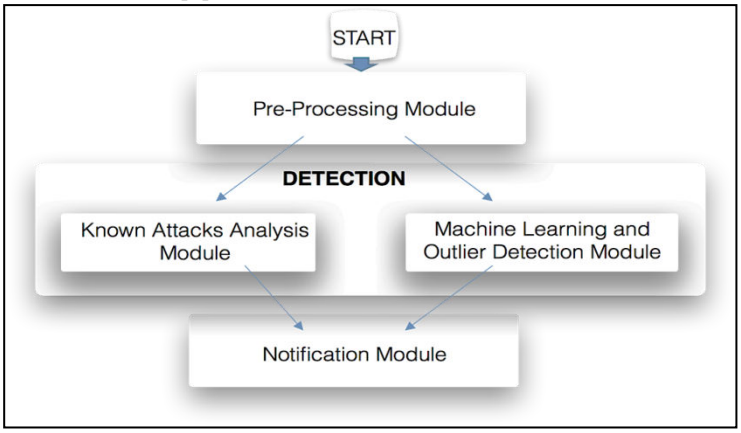

Figure 1- Modules in the system IntelliIDS

4.3.1 Pre-processing module

Input: Raw Apache logs

Output: Vectorised logs in the following format

1. IP
2. Time (Unix time)
3. Request
4. Status

Implementation:

1. Function to parse logs will be written
2. Function to vectorise the logs will be written.

"This will search through the logs for unique words and assigning them a numerical value"

3. Python's pandas library will be used to create Data Frames holding the vectorised logs

4.3.2 Known Attacks Analysis Module

Input: Raw Logs

Output: Report on known attacks detected

Function: Two major functions:

1. Scanning through logs for known attack patterns.
2. Exporting results and calling the notification system when one or more patterns are detected

Implementation: Regular expressions will be used to search through the requested URLs for known attack patterns such as;

XSS : Cross-Site Scripting
SQLI : SQL Injection
CSRF : Cross-Site Request Forgery
DOS : Denial of Service
DT : Directory Traversal
SPAM: Spam

ID : Information Disclosure
RFE : Remote File Execution
LFI : Local File Inclusion

This will be achieved through modified code forked from Scalp (a python library).

4.3.3 Machine Learning and Outlier Detection module

Input: Two Inputs
1. Vectorised logs to be checked
2. Vectorised logs to be trained with (Normal dataset)

Output: Detection Results

Function: Two major functions:
1. Using the Normal Dataset to learn
2. Detect outlier activities based on the Normal Dataset

Implementation: Two unsupervised learning algorithms are tried.
1. *k means*
2. One Class Support Vector Machine (One class SVM)

4.3.4 Notification module

Input: Detection Results

Output: An email to the system administrator containing detected information

Function: sending notifications to system administrators through email

Implementation: Written in Python utilizing the smtplib library.

4.3.5 Real-time Detection Module (yet to be implemented)

Function: Analyse new log lines as they are generated by Apache

4.4 Related packages utilised

To reduce the development and testing time, external libraries in python are used. They include:

i). Numbly (Offering array capabilities for python)

ii). Scikit-learn [24] This Data Science based library includes all machine learning algorithm and test data. It is an open source machine learning library for Python. It features various classification, regression and clustering algorithms both supervised and unsupervised machine learning algorithms, and is built on top of Python numerical and scientific libraries (NumPy and SciPy) for faster interoperation.

iii). Pandas (creating manageable data frames)

5.0 Discussion

For an intelligent log-based intrusion detection system to be termed as successful, it has to identify both known and unknown attack patterns in the given dataset. We now discuss performance measures and experiment results.

5.1 Experiment Design

5.1.1 Performance measures

The performance measures that deemed effective in this case were the true positive rate which will be referred to as the accuracy.

The Detection Rate is the percentage of attacks detected.

$$Detection\ Rate = \frac{Number\ of\ attack\ patterns\ detected}{Number\ of\ attacks\ in\ the\ generated\ Logs}$$

eq. (4)

5.2 Experiment results

A series of experiments were conducted on the two unsupervised leaning algorithms chosen to determine which of the two had a higher performance rate.

5.2.1 Experiment 1 (*k means* Clustering)

i). Test1

Given a training sample of 50000 line of logs that are normal logs and a test sample of 1000 log lines of which 80% were attacks and 20% normal

NB: The Data was labeled for reference purposes.

Table 1 Experiment 1Test 1 Input Logs format

IP (vectored)	TIME (Unix time)	Request (vectored)	Referrer (vectored)	Agent (vectored)
1231122112	1452162152	-1243	3445	23
0100103 4004	14531 64844	52	45	0

Given the parameters in Table 1 , the accuracy of the IDS was 50% which is of no great significance in practice. The parameters were adjusted accordingly by dropping the referrer and the agent paving way for the second test see Table 2.

ii). Test2

With the adjusted parameters, as in Table 2

Table 2: Experiment 1Test 2 Input Logs format adjusted.

IP (vectored)	TIME (Unix time)	Request (Vectored)
1231122112	1452162152	-1243
01001034004	1453164844	52

The accuracy of the system has now increased by 10% giving an accuracy level of 60%. However, the results were still not satisfactory. This demanded another test with more adjustments to the parameters.

iii). Test3

It was noted that the issue was with the time (Unix time) which gave the time in milliseconds hence offering a large dataset that gave the wrong projections to the algorithm. With this in mind, the Unix time was adjusted in the log generator to generate new logs at intervals of 60 second to 300 seconds. This was then rounded up the Unix time to 6 digits working with a smaller number and a better time range to correlate events.

Table 3 Experiment 1 Test 3 Input Logs format adjusted

IP (vectored)	TIME (Unix time)	Request (Vectorized)
1231122112	145216	-1243
01001034004	145316	52

With the new parameters *k means* algorithm delivered an accuracy of 85% which we considered a significant improvement. The *k means* algorithm module gave consistent results ranging from 80 – 85%

5.2.2 Experiment 2 (*One class SVM*)

With previous results from the *k means* algorithm the best working parameters were selected as the initial test.

1. Test1

The same training and testing datasets were used in determining the effectiveness of *One Class SVM*. The format given as input is in Table 4

Table 4: Experiment 2 Test 1 Input Logs format

IP (vectorized)	TIME (Unix time)	Request Vectorized)
1231122112	145216	-1243
01001034004	145316	52

With the success of the *k means* algorithm using the same dataset and parameters, the expectation of this algorithm was high. The novelty detection based algorithm did not fail in detection rates as it gave a success rate ranging from 80% to 85%.

A second test was done to determine whether additional information would be relevant to this algorithm and hence improving the accuracy and detection rate.

2. Test2

A fourth column was added to the datasets the same one that had been removed in 5.2.1.i) Test 1.see Table 5.

Table 5: Experiment 2 Test 2 Input Logs format

IP (vectorized)	TIME (Unix time)	Request (Vectorized)	Referrer (Vectorized)
1231122112	145216	-1243	3445
01001034004	145316	52	45

With these new parameters, the accuracy of the algorithm negatively affected the accuracy which reduced to a range of 70% to 79%. We therefore reverted back to the old parameters to preserve the high accuracy and enhanced detection rate.

6.0 Conclusion

The paper aimed to build an IDS prototype that utilized machine learning to detect known and unknown attack patterns in Apache logs. In this chapter the achievements, problems and limitations of the system are discussed.

Achievements:

1. Modularity of the system. The system has been fully modularized in that additional features can easily be added. Modularization was achieved through classes in python.
2. Machine Learning and Detection: The system detected most of the known attack patterns using the Known-Attack Analysis module and a decent percentage 85% of the same attacks using the Machine Learning and Detection Module.

Challenges: During the implementation, some problems were encountered which include:

1. Lack of Data: When working with unsupervised machine learning algorithms the size of the data matters. This is to say that unsupervised learning works best with large data. This was a problem since it was difficult to obtain access to entire 'access_log' dump. This became a challenge because the only logs we had access to were from our owned websites with minimal traffic hence a year's access_log only contained about 3000 log lines. The solution was to generate logs through a fork of Kiritbasu's Fake-Apache-Log-Generator available [9]. The code was modified.
2. Processing power: With generated logs of up to 1GB (over 15 millions log lines), the processing power required was a bit higher than the machine used to develop the system. Most of the time, a wait of over 40 minutes elapsed before getting the results. In a production setting this would not be realistic since**.**

Limitations:

Since this prototype system was rapidly developed there exist some limitations to its functionality. These include**:**

1. Software development: The system developed has a hard coded log pattern it uses to analyze. The current pattern does not allow for per-logged-in-user analysis which would be a big plus since most attacks are carried out from hijacked accounts and/ or rouge accounts;
2. Definition of the Norm (Normality definition): The success of the system partially depends on the success of the normality definition. For each analysis a normal state of the analyzed system logs had to have been achieved and recorded in order to determine the outliers based on the current analyzed system logs;
3. Real-time detection: Due to time limitations, IntelliIDS operates the same way a batch mode data miner would, this is helpful for post analysis, however, the most appealing way intrusion detection systems ought to work is real-time detection. Plans of implementing this are currently underway and will be released at a later date.

Development, plans are underway to modify the IntelliIDS into a real time scanner. However, many features need to be incorporated in the system to ensure better accuracy and hence higher detection rates. The features that are

missing in this version that would improve the system are:

i. Online Learning: Incorporating the ability to access and analyze remote logs from say a hosted server. This would increase the productivity of the system in that one system can be used from a stationary location to analyze and report on anomalies of remote systems in real-time.
ii. Real time analysis: This is a feature that is vital to any IDS, will ensure that the system analyzes logs as they come in other than a post-analysis based approach that the current version works with.

Though the accuracy of the system needs to be improved, the prototype has been an overall success of an IDS that utilizes unsupervised machine learning algorithms for novel/anomaly detection.

REFERENCES

1. Amoli, P. V., Hamalainen, T., David, G., Zolotukhin, M., & Mirzamohammad, M. (2016). Unsupervised Network Intrusion Detection Systems for Zero-Day Fast-Spreading Attacks and Botnets. *JDCTA (International Journal of Digital Content Technology and its Applications, Volume 10 Issue 2*, 1-13.
2. CERT Coordination Center (CERT/CC). CERT/CC Statistics 1988-2003. http://www.cert.org/stats/cert_stats.html#incidents
3. CISCO Systems Ltd White paper:. The Science of Intrusion Detection System Attack Identification . http://www.cisco.com/en/US/products/sw/secursw/ps2113/products_whitepaper09186a0080092334.shtml. last accessed December 2016 last accessed December 2016
4. Coates, A., Lee, H., & Ng, A. Y. (2010). An analysis of single-layer networks in unsupervised feature learning. *Ann Arbor*, *1001*(48109), 2.
5. Deepa H. Kulkarni Computational Statistics and Predictive Analysis in Machine Learning. (2016). *International Journal Of Science And Research (IJSR)*, *5*(1), 1521-1524. http://dx.doi.org/10.21275/v5i1.nov152818 last accessed February 2017
6. Forrest, S., Perelson, A. S., Allen, L., & Cherukuri, R. (1994, May). Self-nonself discrimination in a computer. In *Research in Security and Privacy, 1994. Proceedings., 1994 IEEE Computer Society Symposium on* (pp. 202-212). IEEE.
7. Gardner, A. B., Krieger, A. M., Vachtsevanos, G., & Litt, B. (2006). One-class novelty detection for seizure analysis from intracranial EEG. *Journal of Machine Learning Research*, *7*(Jun), 1025-1044.
8. Gitau, J. M. (2016) Automated Log Analysis Using AI: Intelligent Intrusion Detection System. Jaramogi Odinga Oginga University of Science andTechnology. http://jooust.ac.ke/projects/siis/2016/JGM-10-2016.pdf last accessed February 2017
9. Github https://github.com/kiritbasu/Fake-Apache-Log-Generator
10. Hand, D. J., Mannila, H., & Smyth, P. (2001). *Principles of data mining*. MIT press.
11. Hinton, G. E., & Sejnowski, T. J. (1999). *Unsupervised learning: foundations of neural computation*. MIT press
12. Kanungo, T, Mount, D. M., Netanyahu, N. S., Piatko, C. D., Silverman, R. and Wu, A. Y. 2002. An efficient *k-means* clustering algorithm: Analysis and implementation. IEEE Transactions on Pattern Analysis and Machine Intelligence, 24(7):881–892.
13. Li, K. L., Huang, H. K., Tian, S. F., & Xu, W. (2003, November). Improving one-class SVM for anomaly detection. *Machine Learning and Cybernetics, 2003 International Conference* Vol. 5, pp. 3077-3081. IEEE.
14. Li, W. (2013). Automatic Log Analysis using Machine Learning: Awesome Automatic Log Analysis version 2.0. http://uu.divaportal.org/smash/get/diva2:667650/FULLTEXT01.pdf last accessed December 2016
15. Ma, P. (2003). Log Analysis-Based Intrusion Detection via Unsupervised Learning. *Master of Science, School of Informatics, University of Edinburgh*.
16. Manevitz, L. M., & Yousef, M. (2001). One-class SVMs for document classification. *Journal of Machine Learning Research*, *2*(Dec), 139-154.
17. Markou, M., & Singh, S. (2003). Novelty detection: a review—part 1: statistical approaches. *Signal processing*, *83*(12), 2481-2497.
18. Matherson, K. (2015). Machine Learning Log File Analysis. http://docplayer.net/10128120-Machine-learning-log-file-analysis.html
19. Muller, K. R., Mika, S., Ratsch, G., Tsuda, K., & Scholkopf, B. (2001). An introduction to kernel-based learning algorithms. *IEEE transactions on neural networks*, *12*(2), 181-201.
20. Parzen, E. (1962). On estimation of a probability density function and mode. *The annals of mathematical statistics*, *33*(3), 1065-1076.
21. Patil , A. S and Patil, D. R. Post-Attack Intrusion Detection using Log Files Analysis. *International Journal of Computer Applications* 127(18):19-21, October 2015. Foundation of Computer Science (FCS), NY, USA.. http://dx.doi.org/10.5120/ijca2015906731 last accessed December 2016
22. Ryan, J., Lin, M. J., & Miikkulainen, R. (1998). Intrusion detection with neural networks. *Advances in neural information processing systems*, 943-949.

23. Schultz, M. G., Eskin, E., Zadok, E., Bhattacharyya, M., & Stolfo, S. J. (2001, June). MEF: Malicious Email Filter-A UNIX Mail Filter That Detects Malicious Windows Executables. In *USENIX Annual Technical Conference, FREENIX Track* (pp. 245-252).
24. *Scikit-learn: machine learning in Python — scikit-learn 0.18.1 documentation*. (2016). *Scikit-learn.org*. Retrieved 2 December 2016, from http://scikit-learn.org/stable/
25. Svensson, C. (2015). Automatic Log Analysis System Integration: Message Bus Integration in a Machine Learning Environment. http://www.divaportal.org/smash/get/diva2:818538/FULLTEXT01.pdf last accessed February 2017
26. Yen, T. F., Oprea, A., Onarlioglu, K., Leetham, T., Robertson, W., Juels, A., & Kirda, E. (2013, December). Beehive: Large-scale log analysis for detecting suspicious activity in enterprise networks. In *Proceedings of the 29th Annual Computer Security Applications Conference* (pp. 199-208). ACM.
27. Zwietasch, T. (2014). Detecting anomalies in system log files using machine learning techniques . http://dx.doi.org/10.18419/opus-3454 last accessed February 2017
28. Rai, K., Davi, M. S. & Guleria, A. (2016), Decision Tree Based Algorithm for Intrusion Detection: *Int. J. Advanced Networking and Applications, Volume: 07 Issue: 04 Pages: 2828-2834 (2016) ISSN: 0975-0290.*