Distributed Network Intrusion Detection System

Himanshu Agarwal, Abhiraj Bishnoi, Santy Blesson Pushparaj, Indrayudh Roy, Parth Gilitwala

Department of Computer Science and Engineering, Symbiosis Institute of Technology, Pune

himanshu.agarwal@sitpune.edu.in
abhiraj.bishnoi@sitpune.edu.in
p.blesson@sitpune.edu.in
indrayudh.roy@sitpune.edu.in
parth.gilitwala@sitpune.edu.in

***Abstract*—Information Security is an issue of very serious global concern in modern times. Now more than ever, there is a pressing need for better Network Security in the form of intelligent Network Intrusion Detection and Prevention Systems that augment traditional lines of defence like Firewalls, chain of trust measures like certificates, two-factor authentication and encryption. With the proliferation of devices connected to the Internet and the unforeseen increase in its use cases, everything from utility companies, stock markets, corporate databases, business Services and research facilities are prone to network based attacks that can result in physical damage, intellectual property theft and catastrophic losses in terms of life and money. No longer are Fire-Sales and the complete takeover of a country's infrastructure in the realm of fiction (think Die-Hard 4). There is a constant war being waged between crackers and the security industry and as time goes on, increasingly sophisticated security mechanisms are needed to keep malicious entities from penetrating the network. In this paper, we examine the design and implementation of an intelligent statistical anomaly based Distributed Network Intrusion Detection System (NIDS), using the principles of supervised machine learning.**

***Keywords*—Distributed Network Intrusion Detection, Supervised Learning, ISCX IDS dataset, Apache Storm**

I. Introduction

A. Problem Statement

The problem of designing a network-based intrusion detection system is as much in the availability of good data to validate the system, as much as in its implementation. In 1999, the KDD Cup Challenge was held, and the eponymous KDD '99 data set became academia's standard for conducting research on intrusion detection. Over time, the KDD '99 dataset has become antiquated, and can no longer serve as an accurate representation of real world network traffic[]. In this paper, we propose a solution that works around this drawback, in addition to addressing two major issues in traditional Network Intrusion Detection Systems:

- The ability of the system to accurately classify malicious packets that conform to slight modifications of traditional attack patterns, i.e., its ability to 'generalise' to new forms of attack patterns that is has been trained to recognize.
- The problem of performance, i.e., the ability of the system to handle increasingly large volumes of data without incurring significant losses in throughput or responsiveness.

B. Motivation

As the internet continues to grow in terms of scale, complexity, openness and accessibility; there is a tremendous proliferation of devices connected to the Internet. With the advent of the 'Internet of Things', the use cases for Internet connected devices continue to grow at a staggering rate and with this increase comes a commensurate increase in the likelihood of network based attacks. The widespread availability of technology and information in the modern age has helped attackers resort to extremely sophisticated measures to break into networks. Now more than ever, there is a pressing need for intelligent and scalable network protection systems and network security as a science is set a pivotal role in the society of the future. In this paper, we study the current state of the art, research on existing intrusion detection mechanisms and apply our learning towards the design and implementation of a distributed statistical anomaly based network intrusion detection system.

We advocate the use of machine learning for classification of incoming packets due to its promise in 'generalising' to new forms of attacks that signature based intrusion detection systems may have problems with. In particular, we opt for a 'Supervised' approach due to the availability of a world class labelled intrusion detection dataset (UNB ISCX IDS 2012). The dataset has been generated using sophisticated modelling techniques and serves as an accurate representation of network traffic present in the real world, with a good mix of benign and attack packets.

The overall traffic of the Internet has increased dramatically, and is predicted to increase at an exponential rate in the near future. As a consequence, there is a commensurate increase in the amount of traffic that is to be analysed by an intrusion detection system, a design constraint which if not taken care of, can lead to a serious bottleneck. This motivated us to employ a cluster based design, which affords the use of commodity hardware over specialized hardware, offering the best return on investment. It offers the possibilities of dynamic load balancing, elasticity, scalability and fault-tolerance; in addition to providing scope for parallelization of computationally heavy operations; which can result in a direct increase in system throughput, responsiveness and performance.

II. BACKGROUND

A. Network Intrusion Detection Systems

Intrusions in the context of network security refer to network attacks against vulnerable services. This may be in the form of data-driven attacks on applications; host-based attacks like privilege escalation, unauthorized logins and access to sensitive files; or malware like viruses, worms and Trojan horses. They attempt to compromise the golden trio of integrity, confidentiality and/or availability of a resource. Intrusions can result in services being denied, systems failing to respond, loss of data, financial loss, espionage; and in extreme cases, loss of life and property. Network intrusion detection is the art and science of detecting these intrusions in a network. A Network Intrusion Detection System(NIDS) serves as a secondary level of protection that analyses all packets passing through the network to detect intruders based on anomalies and/or known signatures, complementing the first layer – the firewall – which is the original point of contact between the intranet and the internet. While the purpose of an NIDS is to notify the network administrator of a potential security breach, a Network Intrusion Detection and Prevention system (NIDPS) has the capability to act on intrusions detected in the network in order to reduce or prevent further damage. This is usually in the form of changes in Firewall policies, dynamically readjusting Firewall rules to block connections from a potentially malicious host.
‘

B. Machine Learning

II.B.1 INTRODUCTION AND BACKGROUND

Machine learning is a field of study that deals with teaching computers to learn something without explicitly programming them to do so. It draws inspiration from many diverse sources ranging from probability and statistics to computer science and biology. The basic premise of machine learning is to provide the machine learning algorithm with a corpus of data and a few characteristic features of the data, and given this information, ‘train’ the machine to infer knowledge or ‘learn’ from the data. Given a few key features of the data at hand, the machine is able to come up with its own mathematical representation of the data, as a function of the chosen features.

In more concrete terms, consider a sample of 1, 00,000 bank transactions, with information like the ages of the parties making a transaction, the amount transferred, the time of the transfer, their account numbers of both parties making the transaction and so forth. If we chose to define the transactions in terms of the account numbers involved, the amount transferred and the time of the transaction, we can have the machine ‘learn’ a pattern that fits this data. Once this pattern has been learnt, it can be used in a number of ways including but not limited to fraud detection and transaction prediction. The biggest challenge is getting the model to generalise to data it hasn’t had exposure to. For example, if we train the algorithm to recognize a pattern in bank X’s customers, can we use the same model on bank Y’s customers? The answer to this question depends on the quality of the dataset, the quality of features chosen and the sheer volume and variety of data available to us. It also depends on the algorithm we choose and the answer to an age old philosophical question, that of Occam’s razor.

II.B.2 SUPERVISED AND UNSUPERVISED MACHINE LEARNING

Supervised learning is a branch of machine learning that deals with the task of inferring a function from labelled training data. The training data consists of a set of training examples. In supervised learning, each example is a pair of an input object (typically a vector) and a desired output value (also called the supervisory signal). A supervised algorithm analyses the training data and produces an inferred function, which can be used for mapping new examples. An optimal scenario will allow for the algorithm to correctly determine an object’s label for unseen circumstances. The parallel in human and animal psychology is often referred to as concept learning, or ‘learning by example’.

By contrast, unsupervised learning is a class of machine learning algorithms that are used to draw inferences from datasets consisting of input data without labelled responses. The most common unsupervised learning method is cluster analysis, which is used for exploratory data analysis to find hidden patterns or grouping in data. The clusters are modelled using a measure of similarity which is defined upon metrics such as Euclidean or probabilistic distance. Since the examples given to the learner are unlabeled, there is no evaluation of the accuracy of the structure that is output by the relevant algorithm, which is one way of distinguishing unsupervised leaning from supervised and reinforcement learning.

II.B.3 USING MACHINE LEARNING FOR CLASSIFICATION

A popular application of many machine learning algorithms is to classify a sample space of objects into one or many categories. This is done by training the algorithm with a dataset and teaching it to classify objects on the basis of relevant features. For example: Given enough labelled instances of photographs of a baby and a fully grown man, the machine can be taught to recognize co-relations between the features selected and the photograph of a man. Likewise, it can infer co-relations between the photograph of a baby and the relevant features selected to come up with a mathematical abstraction of a baby in terms of the features selected. When we feed the algorithm with the photograph of a baby that was not part of the training set; it extracts relevant features from the photograph and plugs it into the model it has developed. If the result corroborates with what it expects, it classifies the photograph as being that of a baby. If it doesn’t, repeats the process iteratively with other models it has ‘learned’.

III. RESEARCH METHODOLOGY AND IMPLEMENTATION

A. *Statistical Anomaly Based IDS*

We opt for a statistical anomaly based approach to classifying packets flowing through the network. This is to work around the limitations of signature based approaches in dealing with modified forms of traditional attack patterns. In a nutshell, the system works by identifying 'patterns' in the packets flowing through a network and classifying a packet as a potential threat based on its position relative to a baseline that is calculated using statistical techniques. The system computes a numerical value for each incoming packet using a mathematical abstraction (feature extraction and reduction), and compares this value against the learned Baseline (which is the result of a mathematical abstraction of well formed packets and anomalous packets) to test for potential threats. If the value is found to be above the threshold (baseline), this indicates anomalous behaviour, and the packet is flagged as being potentially dangerous.

B. *Dataset Selection*

In network intrusion detection (IDS), anomaly-based approaches in particular suffer from accurate evaluation, comparison, and deployment which originate from the scarcity of adequate datasets. Many such datasets are internal and cannot be shared due to privacy issues, others are heavily anonymized and do not reflect current trends, or they lack certain statistical characteristics. Despite the significant contributions of the revered DARPA and KDD '99 datasets in the intrusion detection domain, their accuracy and their ability to reflect real world conditions have been criticized in McHugh (2000) and Brown et al.(2009). The DARPA datasets were constructed for network security purposes, to simulate traffic seen in a medium sized US Air Force base. Upon careful examination of DARPA '98 and '99, many issues have been raised including their failure to resemble real world traffic and numerous irregularities due to synthetic nature of their approach to data generation and insertion into the dataset. Since KDD '99 draws heavily from DARPA '98, these issues apply to KDD as well and made it unsuitable for our purposes. As network behaviours and patterns change and intrusions evolve, it has become necessary to move away from static and one-time datasets towards more dynamically generated datasets which not only reflect the traffic compositions and intrusions at that time, but are also modifiable, extensible and reproducible. The ISCX IDS dataset from the Canadian Institute of Cyber security at the University of New Brunswick dataset is systematically generated to meet these needs []. In additional to being available publicly upon request, it has the following features

- ❖ Realistic network traffic
- ❖ Labelled dataset
- ❖ Total interaction capture
- ❖ Complete capture
- ❖ Diverse intrusion scenarios
- ❖ Completely identified
- ❖ Large Size(~190 GB)

An overview of the composition of network traffic that is part of the dataset is outlined in table I.

TABLE II
ISCX IDS DATASET TRAFFIC COMPOSITION

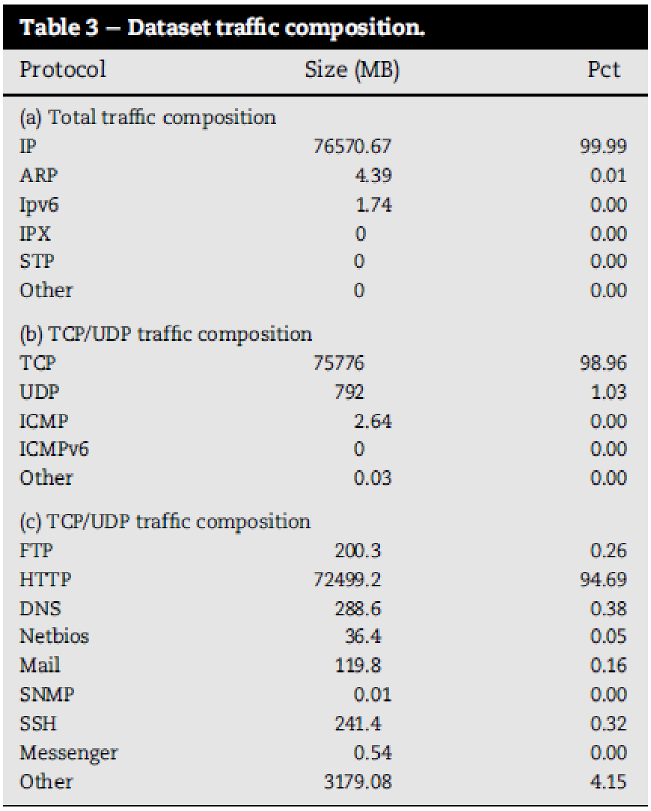


Table 3 – Dataset traffic composition.

Protocol	Size (MB)	Pct
(a) Total traffic composition		
IP	76570.67	99.99
ARP	4.39	0.01
Ipv6	1.74	0.00
IPX	0	0.00
STP	0	0.00
Other	0	0.00
(b) TCP/UDP traffic composition		
TCP	75776	98.96
UDP	792	1.03
ICMP	2.64	0.00
ICMPv6	0	0.00
Other	0.03	0.00
(c) TCP/UDP traffic composition		
FTP	200.3	0.26
HTTP	72499.2	94.69
DNS	288.6	0.38
Netbios	36.4	0.05
Mail	119.8	0.16
SNMP	0.01	0.00
SSH	241.4	0.32
Messenger	0.54	0.00
Other	3179.08	4.15

The dataset has instances of attack packets and normal packets in an approximate ratio of 2:5, with the attack packets belonging to four high level classes of attacks:

- ❖ Denial of Service(DoS)
- ❖ Distributed Denial of Service(DDoS)
- ❖ Brute Force SSH
- ❖ Port Scanning

The following data has been taken verbatim from the UNB ISCX IDS 2012 website:

"The UNB ISCX 2012 intrusion detection evaluation dataset consists of the following 7 days of network activity (normal and malicious):

Day, Date, Description, Size (GB)

- Friday, 11/6/2010, Normal Activity. No malicious activity, 16.1
- Saturday, 12/6/2010, Normal Activity. No malicious activity, 4.22

- Sunday, 13/6/2010, Infiltrating the network from inside + Normal Activity, 3.95
- Monday, 14/6/2010, HTTP Denial of Service + Normal Activity, 6.85
- Tuesday, 15/6/2010, Distributed Denial of Service using an IRC Botnet, 23.4
- Wednesday, 16/6/2010, Normal Activity. No malicious activity, 17.6
- Thursday, 17/6/2010, Brute Force SSH + Normal Activity, 12.3”

The comprehensive nature of this dataset, both in size and diversity, ease of availability and the lack of viable alternatives prompted us to choose the UNB ISCX IDS 2012 dataset as the dataset of choice for training our model.

C. *Feature Selection*

Feature Selection or attribute selection is the process of searching for and selecting the best subset of attributes that serve as an accurate representation of the data present in the dataset. The notion of “best” is relative to the problem at hand, but typically means highest accuracy. The ISCX dataset is a fully labelled dataset that contains network capture for a span of 7 days. The network capture is available in the form of full packet capture (PCAP) files which contain four different attack scenarios along with normal traffic. Out of the 19 features available in the labelled dataset we select 8 features that are important for our research work. Relevant features are selected using the State Space Search feature extraction method built in the WEKA Data Mining Software. In addition to the 8 features we add a new feature called ‘duration’ obtained from combining the features ‘startDateTime’ and ‘stopDateTime’, in order to account for the duration of connections and reduce the number of total features. State space search is a process in which successive configurations or states of an instance are considered, with the goal of finding a goal state with a desired property. Problems are often modelled as a state space, a set of states that a problem can be in. The set of states forms a graph where two states are connected if there is an operation that can be performed to transform the first state into the second. The three key benefits of performing feature selection on datasets are as follows:

- ❖ Reduces Over-fitting: Less redundant data means less opportunity to make decisions based on noise.
- ❖ Improves Accuracy: Less misleading data means modelling accuracy improves.
- ❖ Reduces Training Time: Less data means that algorithms train faster.

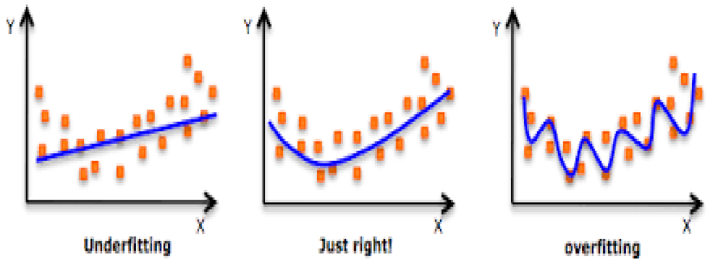
The list of nine features selected to train the model are the application name, the total number of destination packets, the total number of source packets, the direction of the connection (source to destination or vice versa), the source address, the destination address, the name of the protocol and the duration of the connection, calculated by subtracting the ‘startDateTime’ from the ‘stopDateTime’.

D. *Supervised Machine Learning*

We opt for a supervised approach to train our model due to the availability of a high quality labelled data set. The system uses supervised learning techniques to infer a model of regular network traffic from the training data, proceeding to use this model to classify network packets on a real time basis. Prior to training, the dataset id partitioned into two halves; the ‘training set’ and the ‘testing set’ in a 70-30 split. The training set is used to ‘train’ the model, serving as input data to the supervised learning algorithm during the learning process, wherein the algorithm makes a best effort attempt to learn the model/pattern underlying the training dataset. The testing set is used to validate the performance of the classifier generated by the machine learning algorithm,, as its forced to make predictions on data that has no prior exposure to, which simulates a real world situation wherein the model has to make accurate predictions on the basis of unseen data.

The biggest challenge is to tune the parameters of the algorithm in such a manner that the generated model best fits the data without over-fitting or under-fitting to the data, i.e., the model generalises well to unknown instances. Under-fitting refers to creating a highly simplified model that misclassifies a large part of the training data set in addition to the test dataset; naturally leading to poor accuracy. Over-fitting occurs when a model is excessively complex, such as having too many parameters relative to the number of observations. A model that has been over-fit has poor predictive performance, as it overreacts to minor fluctuations in the training data. Over-fitting occurs when a model begins to "memorize" training data rather than "learning" to generalize from a trend. As an extreme example, if the number of parameters is the same as or greater than the number of observations, a simple model or learning process can perfectly predict the training data simply by memorizing the training data in its entirety, but such a model will typically fail drastically when making predictions about new or unseen data, since the simple model has not learned to generalize at all. The challenge is to find the right balance between the two approaches resulting in an optimal system. The underlying philosophy is to choose the simplest hypothesis that best fits the data and is often known as the principle of Occam’s razor. An illustration of these concepts is shown in Figure I.

FIGURE I
REPRESENTATION OF DIFFERENT FITS IN CLASSIFIERS



E. Classifier Selection

Selecting a suitable classifier is one of the most important aspects of a supervised machine learning problem. Several different types of classifiers were tested, with different permutations of parameters within each classifier type.

Often the hardest part of solving a machine learning problem, choosing the right estimator makes use of the following information for the given problem at hand:

Data sets size

Labelled or Unlabelled dataset

Type of prediction or target classes

Type of data (text, multimedia etc)

From the point of view of our implementation we considered the following classifiers:

Support Vector Machines (SVM)

High accuracy, nice theoretical guarantees regarding overfitting, and with an appropriate kernel they can work well even if the data isn't linearly separable in the base feature space. Especially popular in text classification problems where very high-dimensional spaces are the norm. SVMs are known for being memory-intensive, hard to interpret, and fairly complex to run and tune.

When we trained our model using Linear SVC, there were minimalistic changes over the results obtained when using the ensemble decision trees.

Decision Trees

By far the biggest advantage of using decision trees is that they are easy to interpret and explain. Decision trees handle feature interactions and they are non-parametric, For example, decision trees easily take care of cases where you have class A at the low end of some feature x, class B in the mid-range of feature x, and A again at the high end.

A natural problem that tends to associate with decision trees is overfitting. We avoided overfitting by using an ensemble method of trees for classification. Using this method we can make the system as a whole, scalable and fast.

F. Pre-processing raw data to usable form

Since the essence of the solution is using Machine Learning to classify packets, it is essential to build an abstract mathematical model of the incoming data suitable for classification purposes. We do this by extracting features relevant to the classification task from each incoming packet. The whole process results in the conversion of incoming packets in Packet Capture (PCAP) format into a Comma separated values (CSV) file, with each row consisting of a list of numerical values corresponding to the values of the relevant features selected during the feature selection process. The pre-processing process undertaken during the training phase of the model is very similar to the one taken during real-time packet analysis with the only difference being the presence of a Packet Sniffer as an additional component, and a script to convert the output of the packet sniffer (PCAP file) into a CSV file as outlined above. The PCAP files corresponding to the dataset are ~190 GB in size and it is prohibitively expensive and time consuming to load this data into memory all at once in order to train the model. To work around this limitation, the creators of the dataset have provided XML versions of these PCAP files, and during training, we further reduce these XML files to contain just the features we require. The workflow of the pre-processing stage which generates the CSV file(s) is as follows:

Pre-processing - Creation of CSV files:

- ❖ Create a list of all the XML files to be parsed. These files contain tags which correspond to the features of the packet. Each XML file contains a day's worth of traffic on average, with some days needing 3 XML files to store the day's traffic.
- ❖ Create a set of common dictionaries mapping non numeric features to numeric values. When the value of the feature is extracted from between the tags, it's fed into a function which attempts to locate it in the corresponding dictionary. If the value is found, the numerical equivalent of the value is returned. If not, the value is added to the dictionary as a new element and its equivalent numerical value is generated.
- ❖ For each packet, a list of numerical values is extracted, are the values of the features we have selected and these values are written in the form of a comma separated record in the CSV file.
- ❖ This process is repeated till all 12 XML files have been converted into CSV files.

Pre-processing - Creating Test and Train files for each attack:

- ❖ There are 4 kinds of attacks in the dataset - DOS, DDOS, Brute force, and Network Infiltration.

- ❖ A test and training set was required and each file was supposed to contain attacks and normal packets in the ratio of 30:70.
- These packets were randomly selected from the generated csv files.

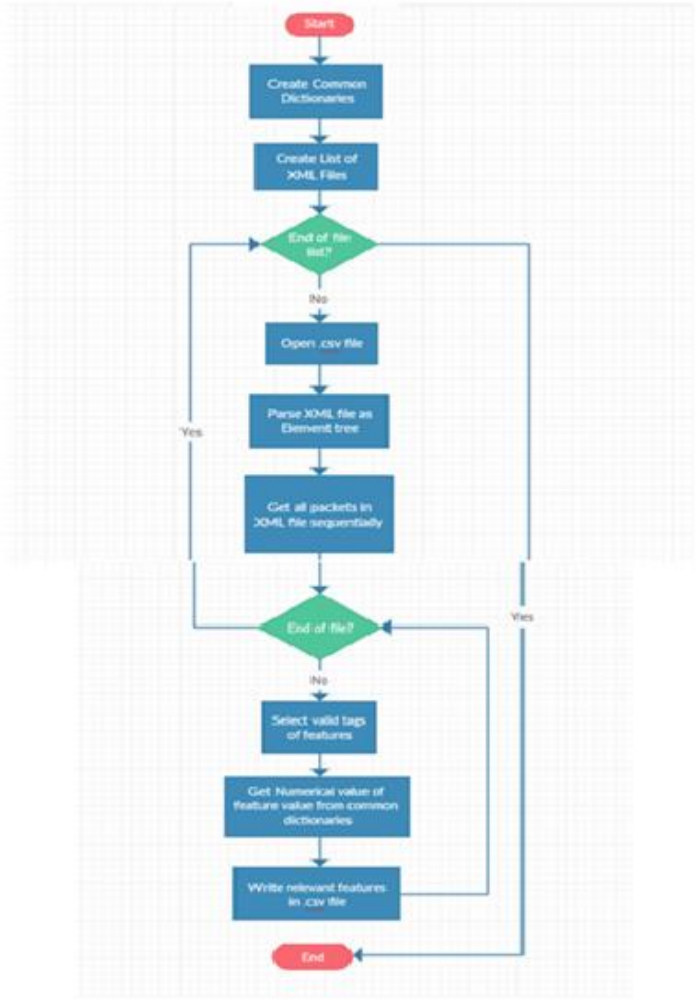

Start
Create Common Dictionaries
Create List of XML Files
End of file list?
No
Open .csv file
Parse XML file as Element tree
Get all packets in XML file sequentially
Yes
End of file?
No
Select valid tags of features
Get Numerical value of feature value from common dictionaries
Write relevant features in .csv file
Yes
End

FIGURE I
FLOWCHART DESCRIBING PRE-PROCESSING OF PACKETS

In a nutshell, our system works by identifying ‘patterns’ in the packets flowing through a network and classifying a packet as a potential threat based on its position relative to a baseline that is calculated using statistical techniques. The system 'learns' from labelled training packets in the ISCX ’12 dataset to come up with an underlying mathematical representation of well formed packets that don’t pose a threat to the network and anomalous packets that are potentially dangerous using supervised classification algorithms.

The system computes a numerical value for each incoming packet using a mathematical abstraction (Feature Extraction and reduction), and compares this value against the learned Baseline (which is the result of a mathematical abstraction of well formed packets and anomalous packets) to test for potential threats. If the value is found to be above the threshold (baseline), this indicates anomalous behavior, and the packet is flagged as being potentially dangerous.

easy way to comply with the conference paper formatting requirements is to use this document as a template and simply type your text into it.

C. Page Layout

r must use a page size corresponding to A4 which is 210mm (8.27") wide and 297mm (11.69") long. The margins must be set as follows:

- Top = 19mm (0.75")
- Bottom = 43mm (1.69")
- Left = Right = 14.32mm (0.56")

Your paper must be in two column format with a space of 4.22mm (0.17") between columns.

IV. Page Style

All paragraphs must be indented. All paragraphs must be justified, i.e. both left-justified and right-justified.

V. Text Font of Entire Document

The entire document should be in Times New Roman or Times font. Type 3 fonts must not be used. Other font types may be used if needed for special purposes.

Recommended font sizes are shown in Table 1.

VI. Title and Author Details

Title must be in 24 pt Regular font. Author name must be in 11 pt Regular font. Author affiliation must be in 10 pt Italic. Email address must be in 9 pt Courier Regular font.

TABLE III
Font Sizes for Papers

Font Size	Appearance (in Time New Roman or Times)		
	Regular	**Bold**	**Italic**
8	table caption (in Small Caps), figure caption, reference item		reference item (partial)
9	author email address (in Courier), cell in a table	abstract body	abstract heading (also in Bold)
10	level-1 heading (in Small Caps), paragraph		level-2 heading, level-3 heading, author affiliation
11	author name		
24	Title		

All title and author details must be in single-column format and must be centered.

Every word in a title must be capitalized except for short minor words such as “a”, “an”, “and”, “as”, “at”, “by”, “for”, “from”, “if”, “in”, “into”, “on”, “or”, “of”, “the”, “to”, “with”.

Author details must not show any professional title (e.g. Managing Director), any academic title (e.g. Dr.) or any membership of any professional organization (e.g. Senior Member IEEE).

To avoid confusion, the family name must be written as the last part of each author name (e.g. John A.K. Smith).

Each affiliation must include, at the very least, the name of the company and the name of the country where the author is based (e.g. Causal Productions Pty Ltd, Australia).

Email address is compulsory for the corresponding author.

VII. Section Headings

No more than 3 levels of headings should be used. All headings must be in 10pt font. Every word in a heading must be capitalized except for short minor words as listed in Section III-B.

1) Level-1 Heading: A level-1 heading must be in Small Caps, centered and numbered using uppercase Roman numerals. For example, see heading “III. Page Style” of this document. The two level-1 headings which must not be numbered are “Acknowledgment” and “References”.

2) Level-2 Heading: A level-2 heading must be in Italic, left-justified and numbered using an uppercase alphabetic letter followed by a period. For example, see heading “C. Section Headings” above.

3) Level-3 Heading: A level-3 heading must be indented, in Italic and numbered with an Arabic numeral followed by a right parenthesis. The level-3 heading must end with a colon. The body of the level-3 section immediately follows the level-3 heading in the same paragraph. For example, this paragraph begins with a level-3 heading.

D. Figures and Tables

Figures and tables must be centered in the column. Large figures and tables may span across both columns. Any table or figure that takes up more than 1 column width must be positioned either at the top or at the bottom of the page.

Graphics may be full color. All colors will be retained on the CDROM. Graphics must not use stipple fill patterns because they may not be reproduced properly. Please use only *SOLID FILL* colors which contrast well both on screen and on a black-and-white hardcopy, as shown in Fig. 1.

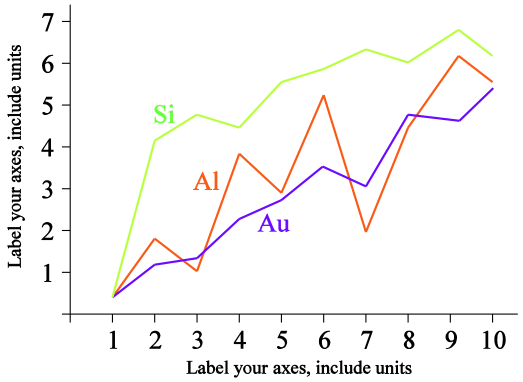


Fig. 1 A sample line graph using colors which contrast well both on screen and on a black-and-white hardcopy

Fig. 2 shows an example of a low-resolution image which would not be acceptable, whereas Fig. 3 shows an example of an image with adequate resolution. Check that the resolution is adequate to reveal the important detail in the figure.

Please check all figures in your paper both on screen and on a black-and-white hardcopy. When you check your paper on a black-and-white hardcopy, please ensure that:

- the colors used in each figure contrast well,
- the image used in each figure is clear,

- all text labels in each figure are legible.

E. *Figure Captions*

Figures must be numbered using Arabic numerals. Figure captions must be in 8 pt Regular font. Captions of a single line (e.g. Fig. 2) must be centered whereas multi-line captions must be justified (e.g. Fig. 1). Captions with figure numbers must be placed after their associated figures, as shown in Fig. 1.

Fig. 2 Example of an unacceptable low-resolution image

Fig. 3 Example of an image with acceptable resolution

F. *Table Captions*

Tables must be numbered using uppercase Roman numerals. Table captions must be centred and in 8 pt Regular font with Small Caps. Every word in a table caption must be capitalized except for short minor words as listed in Section III-B. Captions with table numbers must be placed before their associated tables, as shown in Table 1.

G. *Page Numbers, Headers and Footers*

Page numbers, headers and footers must not be used.

H. *Links and Bookmarks*

All hypertext links and section bookmarks will be removed from papers during the processing of papers for publication. If you need to refer to an Internet email address or URL in your paper, you must type out the address or URL fully in Regular font.

I. *References*

The heading of the References section must not be numbered. All reference items must be in 8 pt font. Please use Regular and Italic styles to distinguish different fields as shown in the References section. Number the reference items consecutively in square brackets (e.g. [1]).

When referring to a reference item, please simply use the reference number, as in [2]. Do not use “Ref. [3]” or “Reference [3]” except at the beginning of a sentence, e.g. “Reference [3] shows …”. Multiple references are each numbered with separate brackets (e.g. [2], [3], [4]–[6]).

Examples of reference items of different categories shown in the References section include:

- example of a book in [1]
- example of a book in a series in [2]
- example of a journal article in [3]
- example of a conference paper in [4]
- example of a patent in [5]
- example of a website in [6]
- example of a web page in [7]
- example of a databook as a manual in [8]
- example of a datasheet in [9]
- example of a master’s thesis in [10]
- example of a technical report in [11]
- example of a standard in [12]

VIII. Conclusions

The version of this template is V2. Most of the formatting instructions in this document have been compiled by Causal Productions from the IEEE LaTeX style files. Causal Productions offers both A4 templates and US Letter templates for LaTeX and Microsoft Word. The LaTeX templates depend on the official IEEEtran.cls and IEEEtran.bst files, whereas the Microsoft Word templates are self-contained. Causal Productions has used its best efforts to ensure that the templates have the same appearance.

Acknowledgment

The heading of the Acknowledgment section and the References section must not be numbered.

Causal Productions wishes to acknowledge Michael Shell and other contributors for developing and maintaining the IEEE LaTeX style files which have been used in the preparation of this template. To see the list of contributors, please refer to the top of file IEEETran.cls in the IEEE LaTeX distribution.

References

[1] S. M. Metev and V. P. Veiko, *Laser Assisted Microtechnology*, 2nd ed., R. M. Osgood, Jr., Ed. Berlin, Germany: Springer-Verlag, 1998.

[2] J. Breckling, Ed., *The Analysis of Directional Time Series: Applications to Wind Speed and Direction*, ser. Lecture Notes in Statistics. Berlin, Germany: Springer, 1989, vol. 61.

[3] S. Zhang, C. Zhu, J. K. O. Sin, and P. K. T. Mok, “A novel ultrathin elevated channel low-temperature poly-Si TFT,” *IEEE Electron Device Lett.*, vol. 20, pp. 569–571, Nov. 1999.

[4] M. Wegmuller, J. P. von der Weid, P. Oberson, and N. Gisin, “High resolution fiber distributed measurements with coherent OFDR,” in *Proc. ECOC’00*, 2000, paper 11.3.4, p. 109.

[5] R. E. Sorace, V. S. Reinhardt, and S. A. Vaughn, “High-speed digital-to-RF converter,” U.S. Patent 5 668 842, Sept. 16, 1997.

[6] (2002) The IEEE website. [Online]. Available: http://www.ieee.org/

[7] M. Shell. (2002) IEEEtran homepage on CTAN. [Online]. Available: http://www.ctan.org/tex-archive/macros/latex/contrib/supported/IEEEtran/

[8] *FLEXChip Signal Processor (MC68175/D)*, Motorola, 1996.

[9] “PDCA12-70 data sheet,” Opto Speed SA, Mezzovico, Switzerland.

[10] A. Karnik, “Performance of TCP congestion control with rate feedback: TCP/ABR and rate adaptive TCP/IP,” M. Eng. thesis, Indian Institute of Science, Bangalore, India, Jan. 1999.

[11] J. Padhye, V. Firoiu, and D. Towsley, “A stochastic model of TCP Reno congestion avoidance and control,” Univ. of Massachusetts, Amherst, MA, CMPSCI Tech. Rep. 99-02, 1999.

[12] *Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specification*, IEEE Std. 802.11, 1997.