

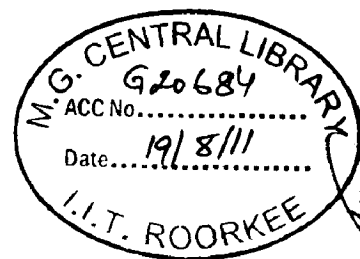
SNORT BASED HYBRID INTRUSION DETECTION SYSTEM WITH AUTOMATIC SIGNATURE GENERATION

A DISSERTATION

*Submitted in partial fulfillment of the
requirements for the award of the degree
of*
MASTER OF TECHNOLOGY
in
COMPUTER SCIENCE AND ENGINEERING

By

SHIV KUMAR



**DEPARTMENT OF ELECTRONICS AND COMPUTER ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY ROORKEE
ROORKEE -247 667 (INDIA)
JUNE, 2011**

CANDIDATE'S DECLARATION

I hereby declare that the work, which is being presented in the dissertation entitled "SNORT BASED HYBRID INTRUSION DETECTION SYSTEM WITH AUTOMATIC SIGNATURE GENERATION" towards the partial fulfillment of the requirement for the award of the degree of **Master of Technology in Computer Science and Engineering** submitted in the Department of Electronics and Computer Engineering, Indian Institute of Technology Roorkee, Roorkee, Uttarakhand (India) is an authentic record of my own work carried out during the period from July 2010 to June 2011, under the guidance of **Dr. R. C. Joshi, Professor**, Department of Electronics and Computer Engineering, IIT Roorkee.

The matter presented in this dissertation has not been submitted by me for the award of any other degree of this or any other Institute.

Date: 30/5/11

Place: Roorkee




(SHIV KUMAR)

CERTIFICATE

This is to certify that the above statement made by the candidate is correct to the best of my knowledge and belief.

Date: 30/5/11

Place: Roorkee



(Dr. R. C. JOSHI)

Professor

Department of Electronics and Computer Engineering

IIT Roorkee.

ACKNOWLEDGEMENT

First and foremost, I would like to extend my heartfelt gratitude to my guide and mentor **Dr. R. C. Joshi**, Professor, Department of Electronics and Computer Engineering, Indian Institute of Technology Roorkee, for his invaluable advices, guidance, encouragement and for sharing his broad knowledge. His wisdom, knowledge and commitment to the highest standards inspired and motivated me. He has been very generous in providing the necessary resources to carry out my research. He is an inspiring teacher, a great advisor, and most importantly a nice person.

I also wish to thank Emmanuel S. Pilli for his valuable suggestions and timely help regarding the domain knowledge and datasets. I am greatly indebted to all my friends, who have graciously applied themselves to the task of helping me with ample moral supports and valuable suggestions.

On a personal note, I owe everything to the Almighty and my parents. The support which I enjoyed from my father, mother and other family members provided me the mental support I needed.


SHIV KUMAR

ABSTRACT

With the tremendous growth of network-based services and information on Internet, the number of the network hosts has sharply increased. But the network-based computers are often vulnerable; due to this reason we need systems to detect these vulnerabilities. Intrusion detection is the process of identifying suspicious activities on a target system or network.

Intrusion Detection System (IDS) used today suffer from several shortcomings in the presence of complex and unknown attacks. Hence in this dissertation Snort based hybrid Intrusion Detection System with automatic signature generation is investigated. The problem of unknown attacks with IDS is solved using anomaly detection. Entropy is one of the well known detection technique used in intrusion detection. In this work, a system is designed with the help of Entropy based technique and integrated with real time system Snort (Signature based technique) so that it can have advantages of both techniques. A feature extraction system is designed which can be used for calculating the important features for which entropy can be calculated for anomaly detection. Another issue of IDS, hectic amount of alert data, has also been addressed by developing alert unification system which comprises of alert ranking and reduction system. Alert reduction system is used to efficiently unify alerts generated by hybrid IDS whereas alert ranking system is used to give ranks to those alerts according to their importance.

Also signature database of IDS is very limited and it is very hectic to manually update it. For automating this task various signature generation systems were proposed. In this thesis, an automatic signature system based on honeypot is proposed with Real Time Rule Accession (RTRA) capability. Honeypot is used to collect attack data on the network which is used by association rule generation algorithm for generating rules. These rules are added in Snort. An open source signature generation system, Honeycomb is compared with our system. The experiment results show the dominance of our system over honeycomb in respect to quantity, completeness and non-redundancy of rules.

TABLE OF CONTENTS

	Page No.
Candidate's Declaration & Certificate.....	i
Acknowledgements.....	ii
Abstract.....	iii
Table of Contents.....	iv
List of Figures.....	vii
List of Tables.....	viii
1. Introduction and Statement of the Problem.....	1
1.1 Introduction and motivation.....	1
1.2 Statement of the Problem.....	3
1.3 Organization of the Report.....	3
2. Background and Related Work.....	5
2.1 Intrusion Detection Model.....	5
2.2 Classification of Intrusion Detection System (IDS).....	7
2.2.1 Classification on the basis of techniques of Intrusion Detection....	7
2.2.2 Classification on the basis of source of input for Intrusion Detection	7
2.3 Network based IDS (NIDS).....	8
2.3.1 Approaches of NIDS.....	8
2.3.2 Limitation of NIDS.....	9
2.4 Anomaly detection approaches.....	9
2.4.1 Data mining approaches to Intrusion Detection	10
2.4.2 Statistical approaches to Intrusion Detection.....	11
2.4.3 Other techniques of Intrusion Detection.....	12
2.5 Snort – A Network based IDS.....	13
2.5.1 Introduction to Snort.....	13

2.5.2	Components of Snort.....	14
2.5.3	Snort modes.....	17
2.6	Automatic signature generation systems.....	17
2.6.1	Honeycomb.....	17
2.6.2	Other systems.....	18
2.7	Alert correlation.....	19
2.8	Research gaps.....	19
3.	Proposed Framework for Intrusion Detection System.....	21
3.1	System framework.....	21
3.2	Automatic Snort compatible signature generation system (ASSG).....	21
3.3	Hybrid IDS based on Snort and Entropy technique.....	24
3.4	Feature extraction system.....	27
4.	Detailed Design and Implementation.....	30
4.1	Signature generation module.....	30
4.1.1	Attack traffic capturing module.....	31
4.1.2	Apriori association rule mining module.....	31
4.1.3	RTRA in Snort module.....	31
4.2	Feature extraction module.....	32
4.2.1	Preprocessing.....	32
4.2.2	Discretization algorithm.....	32
4.2.3	Feature analysis algorithm.....	33
4.2.4	Result analysis system.....	34
4.3	Information extraction module.....	35
4.4	Entropy based detection module with Suspect Index.....	36
4.4.1	Learning algorithm.....	37
4.4.2	Long term statistics based.....	38
4.4.3	Short term statistics based.....	38
4.5	Signature based IDS and alert generation module.....	38
4.6	Alert unification module.....	39

5. Experimental Results and Discussions.....	40
5.1 Signature generation system.....	40
5.1.1 Data logging and system parameters.....	40
5.1.2 RTRA without optimization.....	41
5.1.3 RTRA with optimization.....	43
5.1.4 Comparison with Honeycomb.....	44
5.2 Feature extraction with KDD.....	46
5.3 Hybrid IDS.....	47
5.3.1 Data collection.....	47
5.3.2 Entropy calculation.....	48
5.3.3 Suspect Index of hosts.....	50
5.3.4 Alert reduction and ranking system.....	51
6. Conclusions and Scope for Future Work.....	52
6.1 Conclusions.....	52
6.2 Scope for Future work.....	53
REFERENCES.....	54
LIST OF PUBLICATIONS.....	58

LIST OF FIGURES

Figure No.		Page No.
Figure 2.1	Intrusion Detection Model.....	6
Figure 2.2	Components of Snort.....	14
Figure 2.3	Rule Representation of Snort.....	16
Figure 3.1	Level 0 System design.....	22
Figure 4.1	System design for signature generation system.....	30
Figure 4.2	System design for feature extraction system.....	32
Figure 4.3	System design for Hybrid IDS.....	35
Figure 5.1	Rules generated by Weka and accession in Snort against successive stream of attack records (number of attributes considered in frequent item set is 6).....	42
Figure 5.2	Rules generated by Weka and accession in Snort against successive stream of attack records (number of attributes considered is 7)	42
Figure 5.3	Rules generated by Weka and accession in Snort (after removing redundancy) when number of attributes considered in frequent item set is 7	44
Figure 5.4	Information gain of various features of Smurf attack.....	46
Figure 5.5	Graphs representing relation between Entropy and Time stamp for Source IP.....	48
Figure 5.6	Graphs representing relation between Entropy and Time stamp for feature Source Port.....	49

LIST OF TABLES

Table No.		Page No.
Table 2.1	Rules generated by Honeycomb.....	18
Table 5.1	Various parameters used for system analysis.....	40
Table 5.2	Snort rules generated for RTRA.....	41
Table 5.3	Alerts generated by our IDS.....	44
Table 5.4	No. of rules generated.....	45
Table 5.5	KDD Statistics.....	47
Table 5.6	Relationship between threshold and no. of anomalous point.....	50
Table 5.7	Feature values and their Suspect Index.....	50
Table 5.8	Comparison between Snort and Proposed System.....	51

Chapter 1

Introduction and Statement of the Problem

1.1 Introduction and motivation

Due to rapid growth of Internet and network based services; security becomes the primary concern for organizations. There are several ways in which an attacker can attack the network of an organization. These can be accessing information for which he is not authorized, bringing down the whole network, etc.

The security of a computer is compromised when an intrusion takes place. An intrusion can be defined as any set of actions that attempt to compromise the integrity, confidentiality or availability of a resource.

The Internet is being used by increasing number of users day by day. A survey [1, 2] show that the number of hosts connected to the Internet has increased to almost 550 million and more than 1.5 billion users are currently using the Internet. The recent survey of Mini Watts Marketing Group [2] estimated that the total number of Internet users was 1,802,330,457 on December 31st 2009. In 2010, the Kaspersky system logged 1,311,156,130 network attacks. That number was just 220 million in 2009.

Here's a review [3] of the major attacks seen in recent years:

- **In 2009:** Sites in the Gawker Media network, which includes some of the most popular blogs, were offline for extended period of time due to a denial of service attack. Dedicated server provider SoftLayer Technologies and domain registrar Dotster are each hit with a large denial of service attacks targeting their domain name servers. The attack on SoftLayer caused availability problems for TechMeme and TwitPic, while thousands of web sites hosted at Dotster were down.

In April, Customers of the Planet are hit by web site outages as a result of a DDoS aimed at the huge hosting company. The Planet said on its Twitter stream, "Given the volume of the attack, our network operations team rerouted all name server

traffic through our DDoS mitigation capabilities.” The Planet hosts more than 48,000 servers.

In the same month, Domain registrar *Register.com* is hit with a DDoS that causes several days of disruptions for its customers. Register.com is the eighth-largest registrar, managing 2.7 million domains.

- **In 2010:** The year 2010 saw a change in first place from the previous year. Based on the number of attacks, NETAPI.buffer-overflow.exploit took the number one slot. This exploit targets the MS08-067 vulnerability and was also used in the Kido family of worms.

The other attacks like DoS SYN flood, WORM, SCAN were the usual suspects, with malicious packets from Helkern (Slammer) worms and exploits targeting the MS03-026 vulnerability, which was used, for example, in the Lovesan worm in 2003. Also of interest was the CVE-2010-2729 a attack in 19th place which was used by Stuxnet to exploit the MS10-61 vulnerability.

Review shows that with the increasing number of Internet users, the cyber crimes also have been increased worldwide to a great value. Fortunately, some intrusion prevention techniques as a first line of defense, such as user authentication (e.g. using passwords and biometrics), avoiding programming errors, and information protection (e.g. encryption) have been applied to protect computer systems. Intrusion prevention alone is not sufficient because as systems are becoming even more complex. As the technology advances, more security is added to the systems. But none of the system is completely secure. There remain some loop holes which make system vulnerable. For example, after it was first reported many years ago, exploitable “buffer overflow” still exist in some recent system software due to programming errors. Hence Intrusion detection system comes into picture to protect the system with these holes. Intrusion detection methods with machine intelligence started appearing in the last few years. Using intrusion detection methods, you can collect and use information from known types of attacks and find out if someone is trying to attack your network or particular hosts. The information collected this way can be used to harden your network security, as well as for legal purposes.

1.2 Statement of the Problem

The aim of this dissertation is to design and implement Snort based hybrid intrusion detection system with automatic signature generation.

With the help of this framework we will be able to overcome the problem of unknown attacks, large amount of alerts for analysis and limited attack signature database using following steps

- a) Automatic signature generation system to deal with the problem of limited attack signature database.
- b) IDS which uses both techniques for Intrusion detection (Signature based and anomaly based) to deal with the problem of unknown attacks.
- c) Alert ranking and reduction system for efficient unification of alerts generated by both techniques. It also focuses on ranking of alerts according to their importance which leads to better and timely analysis of alerts.
- d) Feature extraction system for finding out the important attributes for different network attacks. The features obtained from this can make anomaly detection based intrusion detection more efficient.

1.3 Organization of the Report

This dissertation report comprises of six chapters including this chapter that introduces the topic and states the problem. The rest of the report is organized as follows.

Chapter 2 gives the background of Intrusion Detection System (IDS), description of techniques of intrusion detection, description of Snort, related work in the field of anomaly detection and research gaps.

Chapter 3 describes the framework designed for IDS which comprises of three independent systems automatic signature generation system, hybrid IDS with alert ranking system and feature extraction system.

Chapter 4 gives the detailed design and implementation of all modules of the proposed framework.

Chapter 5 discusses the system parameters, data logging and experimental results of all the three systems. It also includes comparison with existing systems.

Chapter 6 concludes the dissertation work and gives suggestions for future work.

Chapter 2

Background and Related Work

2.1 Intrusion Detection Model

Figure 2.1 shows the first intrusion detection model was given by Dorothy E. Denning in 1988 [4]. This was the base model of all the intrusion detection systems. The model was independent of any particular system, application environment, system vulnerability, or type of intrusion, thereby providing a framework for a general purpose intrusion detection expert system. The model has six main components:

(a) Subjects:

Initiators of activity on a target system, generally normal users

(b) Objects:

Resources managed by the system – files, commands, devices etc.

(c) Audit Records:

Audit records are generated by the target system in response to actions performed or attempted by subjects on objects – user login, file access etc. These are the 6-tuples actions.

<Subject, Action, Object, Exception-Condition, Resource-Usage, Time-Stamp>

(d) Profiles:

Profiles are the structures that characterize the behavior of subjects with respect to objects in terms of statistical metrics and model of observed activity. Observed behavior is characterized in terms of statistical metrics and models. Metrics for example event counter, interval timer, resource measures etc.

Models can be operational model, mean and standard deviation model, multivariate model, time series model etc. Structure of a profile record can be in this format

<Variable-Name, Action-Pattern, Exception-Pattern, Resource-Usage, Period, Variable-Type, Threshold, Subject-Pattern, Object-Pattern, Value>

(e) *Anomaly Records:*

Anomaly records are generated when abnormal behavior is detected.

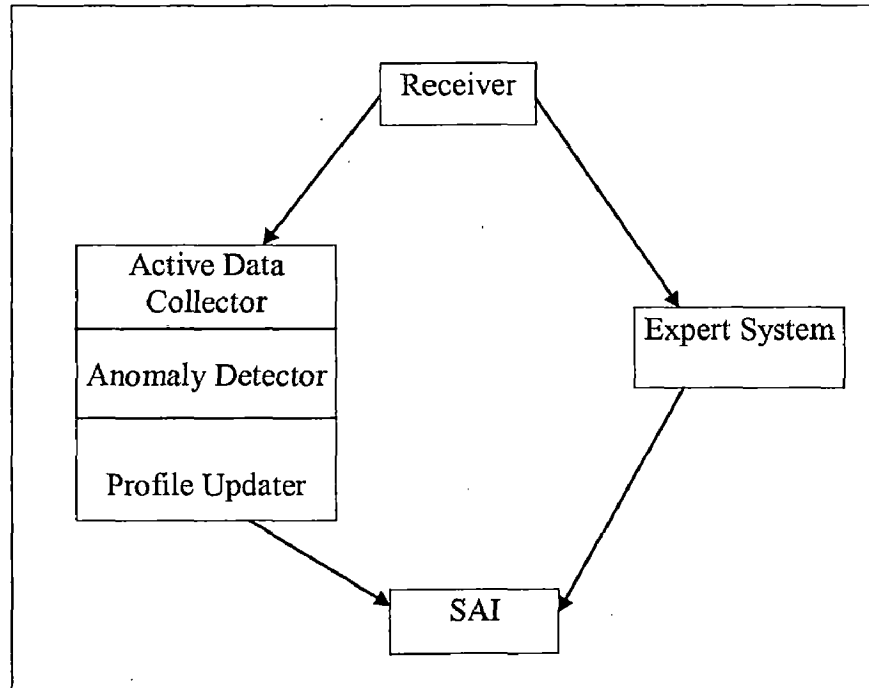


Figure 2.1 Intrusion Detection Model

(f) *Activity Rules:*

Activity rules are actions taken when some condition is satisfied, which update profiles, detect abnormal behavior, relate anomalies to suspected intrusions, and produce reports. Generally four types of rules are defined: Audit-record rule, Periodic-activity-update, Anomaly-record and Periodic-anomaly-analysis rule.

The model can be regarded as the rule based pattern matching system. When an audit record is generated, it is matched against the profiles. Type information in the matching profile then determines what rules to apply to update the profiles, check for abnormal behavior, and report anomalies detected. Figure shows the Intrusion detection system prototype.

2.2 Classification of Intrusion Detection System (IDS)

IDS can be classified into various categories on two bases

- 1) Techniques used for intrusion detection
- 2) Source of input for intrusion detection.

2.2.1 Classification on the basis of techniques of Intrusion detection

There are two techniques of intrusion detection: misuse detection and anomaly detection

(a) Misuse Detection

The system based on misuse detection we called Misuse-based systems which can detect known attacks with high success rate and low time cost, but when faces the unknown attacks it becomes very powerless [5]. This is also called signature based detection.

(b) Anomaly Detection

An anomaly detection technique identifies the observed activities that deviate significantly from the normal usage as intrusions [6] Thus anomaly detection can detect unknown intrusions, which cannot be addressed by misuse detection. But, nearly all the anomaly-based detection techniques have a fatal disadvantage due to mechanism of machine learning that if the results of machine learning cannot cover all the normal or abnormal data then we may get some false alarms.

2.2.2 Classification on the basis of source of input for Intrusion Detection

Generally IDS are classified into two types on the basis of level of intrusion detection

(a) Host based IDS: Host-based intrusion detection systems or HIDS are installed as agents on a host. These intrusion detection systems can look into system and application log files to detect any intruder activity [7]. Some of these systems are reactive. It means that they inform you only when something has happened. Some HIDS are proactive; they can sniff the network traffic coming to a particular host on which the HIDS is present and

generate alerts. IDES is the example of HIDS [8].

(b) *Network based IDS*: NIDS are intrusion detection systems that capture data packets transmitted on the network and match them against rule base of signatures. Matching with any of the signature triggers an alert. Then there will be corresponding action like logging of the packet. SNORT is the example of NIDS.

2.3 Network based IDS (NIDS)

NIDS operate as a stand-alone device that monitors traffic on the network to detect attacks [9]. Its attack recognition module uses four common techniques to recognize an attack signature:

- Frequency or threshold crossing
- Correlation of lesser events
- Pattern, expression or bytecode matching
- Statistical anomaly detection

NIDS comes in two general forms; signature based NIDS and heuristic based NIDS. These two types of NIDS provide a varying degree of security based on several objectives.

2.3.1 Approaches of NIDS

(a) *Signature based NIDS*: Pfleeger and Pfleeger describe signature-based systems as pattern matching systems that detect threats based on the signature of the attack matching a known pattern [10].

(b) *Heuristic based NIDS*: These systems are synonymous with anomaly-based systems, which detect attacks through deviation from a model of normal behavior. To form a model of a normal behavior there are various machine learning approaches such as SVM based, Fuzzy based, Genetic algorithms based, Expert Voting, Game theoretic based, neural network based, Data mining based classification, immunity based from which we will discuss briefly about some of the important approaches in consequent section.

2.3.2 Limitation of NIDS

Now we will discuss about the limitation of both types of NIDS.

(a) Limitation of Signature based NIDS: The attack is known and is detected or the attack is unknown to the system and allows the attack to proceed undetected. Through this definition two inherent limitations can be developed.

- 1) The dependency on rule base of IDS.
- 2) This limitation leads directly to the second limitation whereby signatures that are not placed in its database go undetected.

(b) Limitation of Heuristic based NIDS: These do not suffer same inherent limitations as signature based NIDS. But they do have limitations.

- 1) They need abnormality by intruders for detection. It assumes that intrusions would provide unusual activity that would allow for detection.
- 2) This will create another limitation if the behavioral patterns that are not malicious.
- 3) These systems are limited by the information analyzed and how well the analysis fits into its current system [9].

2.4 Anomaly detection approaches

Traditionally, intrusion detection relies on extensive knowledge of security experts, in particular, on their familiarity with the computer system to be protected. As we have studied known attacks can be handled well with the help of misuse detection but for unknown attacks anomaly detection methodology is used which works by calculation of deviation of user behavior with their profile. So for this we have to design or develop statistical models of user profiles which are made with the help of machine learning techniques. A lot of machine learning techniques have been devised which will be discussed briefly in this section.

2.4.1 Data mining approaches to Intrusion Detection

There is often the need to update an installed IDS due to new attack methods or upgraded computing environments. Since many IDSs are constructed by putting rule base of attack signatures. Hence updating these rule bases of IDSs is expensive and slow. Hence we use data mining framework for generating attack signatures. The central idea is to utilize auditing programs to extract an extensive set of features that describe each network connection or host session, and apply data mining programs to learn rules that accurately capture the behavior of intrusions and normal activities [11]. These rules can then be used for misuse detection.

Data mining generally refers to the process of extracting descriptive models from large stores of data. Recent rapid development in data mining has made available a wide variety of algorithms particularly useful in mining audit data. [11] gives the application of different data mining techniques for building data mining models which are following

(a) Classification:

Intrusion detection can be thought of as a classification problem: classify each audit record or connection or packet into one of a discrete set of possible categories, normal or intrusion.

(b) Association Rules:

There is empirical evidence that program execution and user activities exhibit frequent correlations among system features [11]. The main aim of association rules are to find out frequent relation between attributes. Given a set of records, where each record is a set of items, $\text{support}(X)$ is defined as the percentage of records that contain item set X .

There is often need to study the frequent sequential patterns of network events in order to understand the nature of many attacks.

(c) Feature Construction:

The mined frequent episodes from network connection records are used as guidelines to construct temporal statistical features for building classification models.

A number of research works have been proposed in this area. MADAM ID [12] is a project of Columbia University which showed how data mining techniques can be used to construct IDS in a systematic and automated manner. ADAM [13], IDDM [14] and eBayes [15] uses anomaly detection techniques for intrusion detection. ADAM is a network based IDS which learns the behavior of network through normal attack free traffic and represents them in the form of association rules. The traffic for last delta seconds is considered and is mined for new association rules. Anomaly based techniques usually requires training data to learn normal behavior which is difficult to obtain. Clustering based approaches are also used in this field of research. The above training data is not required here [16]. MINDS [17] project at University of Minnesota uses data mining techniques to automatically detect attacks. A score is assigned to each connection to determine how anomalous it is compared to normal traffic. They are successful in detecting numerous novel intrusions that could not be identified by widely used tools like Snort.

Even Support vector machine (SVM) is also used in this area. After that outlier analysis, another data mining technique is used for detecting intrusion as a outlier. SPOT is used a outlier algorithm. Some approaches based on fuzzy and association rule mining is also given. Wang wunwu [18] gives the fuzzy expert system based approach in which automated learning of fuzzy rules is done with the help of genetic algorithm. Here, genetic algorithm is also used to optimize the membership functions corresponding to different linguistic variables. It also takes into account network features important to intrusion detection. For this it has a feature extraction module. All these data mining techniques mainly depend on the training data i.e. they build model around feature values which can be changed easily during attack. Hence we must take into account overall traffic pattern and have to use statistical techniques.

2.4.2 Statistical approaches to Intrusion Detection

If we talk about the statistical techniques used for building models to detect intrusions, the first model is proposed by Denning [4]. Here Denning told the general model based on anomalies found in the user profiles developed by the statistical algorithms over a period of time. A lot of methods are borrowed from statistical signal theory and pattern

recognition theory for detecting anomalies like Principal Component Analysis (PCA), covariance methods, Auto Regressive Moving Average (ARMA) etc. Entropy is another well-known measure for quantifying the information of network traffic and has been extensively studied for anomaly detection and prevention. G. Nychis et al. [19], uses the entropy based technique for anomaly detection in network traffic. It uses the standard deviation and mean for finding out anomalies I network traffic. But all these statistical techniques use long term network statistics for formulation of user profiles which is not a trivial task. M. Celenk et al. [20], proposes a model for using entropy based anomaly detection which does not use long term statistics. But the problem with this model is that it finds anomalies with respect to current data. So if there are more anomalies than normal data then it will give huge number of false alerts. Also it is not tested in real time.

2.4.3 Other Techniques of Intrusion Detection

If we talk about the Fuzzy logic, it addresses the formal principles of approximate reasoning. It provides a sound foundation to handle imprecision and vagueness as well as mature inference mechanisms by varying degrees of truth. As boundaries are not always clearly defined, fuzzy logic can be used to identify complex patterns or behavior variations. And it can be accomplished by building an intrusion detection system that combines fuzzy logic rules with an expert system in charge of evaluating rule truthfulness [18]. Genetic algorithms are used to tune the fuzzy membership functions to improve the performance and select the set of features available from audit data that provide the most information to fuzzy expert system. Hence main objective of Genetic algorithm based Fuzzy IDS is that we need less fuzzy rules to achieve a certain high rate of recognition of intrusions. Also, generation of fuzzy rules does not need any luminous knowledge.

The main disadvantage of anomaly detection approach is that it may get some false alarms if the results of machine learning cannot cover all the normal or abnormal data. Contiguous voting approach solves this problem [5]. Contiguous voting comes into picture when knowledge does not exactly match with any other Experts. Then all will vote. And if the number of expert which its power value below the arbitrage value is more than that the number of expert which is power value beyond the arbitrage value, we can judge the data abnormal.

If we see different optimization techniques, swarm optimization and ant optimization have been used in intrusion detection as anomaly detection approaches. Game theory have been also used in which IDS decides its strategy by seeing the strategy of attacker and similarly attacker also performs this thing also. At last both attain an equilibrium called as Nash Equilibrium.

2.5 Snort – A Network based IDS

Snort fills an important “ecological niche” in the realm of network security [10]. It is a cross-platform; lightweight network intrusion detection expert systems that can be deployed to monitor small TCP/IP networks and detect a wide variety of suspicious network traffic as well as outright attacks. It can also be deployed rapidly to fill potential holes in a network's security coverage.

2.5.1 Introduction to Snort

Snort is a libpcap-based packet sniffer and logger that can be used as a lightweight network intrusion detection system. It features rules based logging to perform content pattern matching and detect a variety of attacks and probes, such as CGI attacks, SMB probes, and much more. Snort has real-time alerting capability, with alerts being sent to separate alert file. The detection engine is programmed using a simple language that describes per packet tests and actions. Ease of use simplifies and expedites the development of new exploit detection rules.

Snort is cosmetically similar to tcpdump but is more focused on the security applications of packet sniffing. The major feature that Snort has which tcpdump does not is packet payload inspection. Snort decodes the application layer of a packet and can be given rules to collect traffic that has specific data contained within its application layer. This allows Snort to detect many types of hostile activity. Another advantage is that its decoded output display is somewhat more user friendly than tcpdump's output. One powerful feature that Snort and tcpdump share, is the capability to filter traffic with Berkeley Packet Filter(BPF) commands. This allows traffic to be collected based upon a variety of specific packet fields.

2.5.2 Components of Snort:

Snort is logically divided into multiple components [21]. These components work together to detect particular attacks and to generate output in a required format from the detection system.

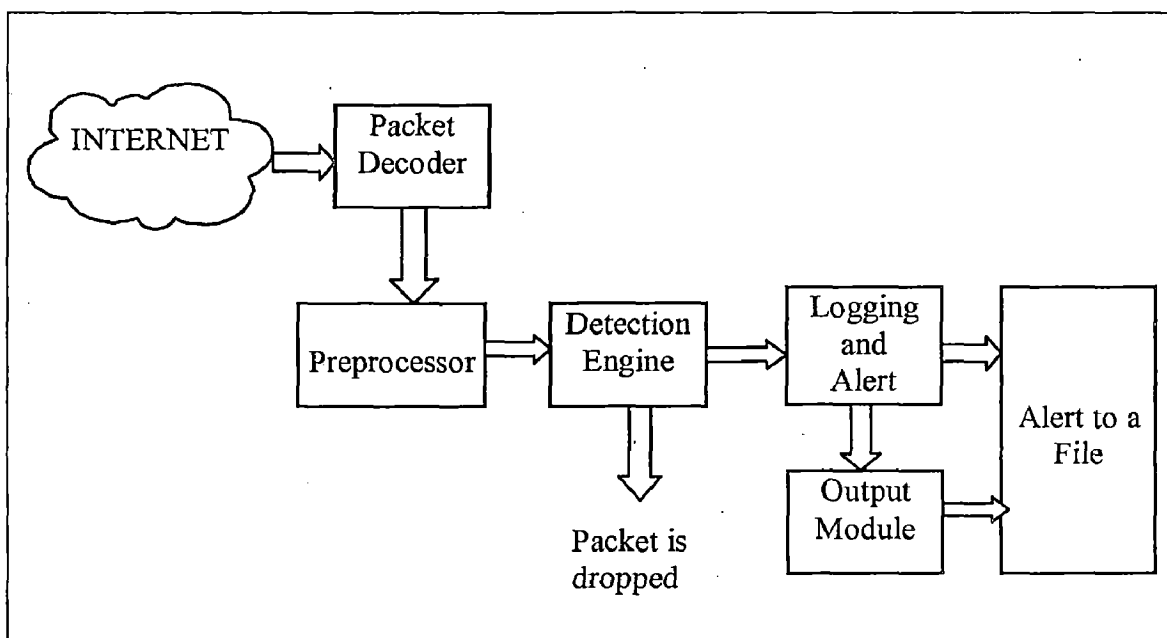


Figure 2.2 Components of Snort

A snort based IDS consists of the following major components:

- Packet Decoder
- Preprocessors
- Detection Engine
- Logging and Alerting System
- Output Modules

Figure 2.2 shows how these components are arranged. Any data packet coming from Internet enters the packet decoder. On its way towards the output modules, it is either dropped, logged or an alert is generated.

A brief introduction to these components is presented below.

(a) Packet Decoder:

The main work of packet decoder is to take packets from different interfaces. There are various interfaces available on a system like ethernet (eth0), wireless LAN (wlan0), local loop (ll0). After capturing packets it transmits to preprocessor or detection engine.

(b) Preprocessors:

Preprocessing can be said as the plug-ins that are used to arrange or modify the packets received from packet decoder before the operations performed by detection engine to detect intrusions. Some of them generate alerts by inspecting the packet headers without passing them to detection engine. They are very important for efficient functioning of IDS as various intruders insert special characters to avoid the signature detection. The work of preprocessors is to modify the packets against these modifications.

Also there are preprocessors for detecting port scanning attacks. They simply check for the inter arrival time between different packets from the same source. They are also used in defragmentation. As you can not apply rules when packet is fragmented and when you transmit a large amount of data, it must be transferred in fragmented packets.

(c) The Detection Engine:

This component is the main component of Snort. As the name suggest, it is responsible for detecting any intrusion. As IDS is one of the kinds of expert system. Like all expert system, it also has a rule base. Now all the packets pass through this component. They are checked against the signatures stored in the rule base of IDS. If any of the signatures generates alert, it will not be checked against other signatures. But the main problem here is the efficient data structure employed for those signatures.

As there are a lot of signatures and IDS have to check all packets against these signatures hence this is a time critical part of IDS. Depending on the no. of rules and data structure, IDS may take variable time for responding to different packets. So finally the main design issues for these components are

- Number of rules

- Properties of the machine on which IDS is installed
- Network traffic
- Pattern matching algorithm used

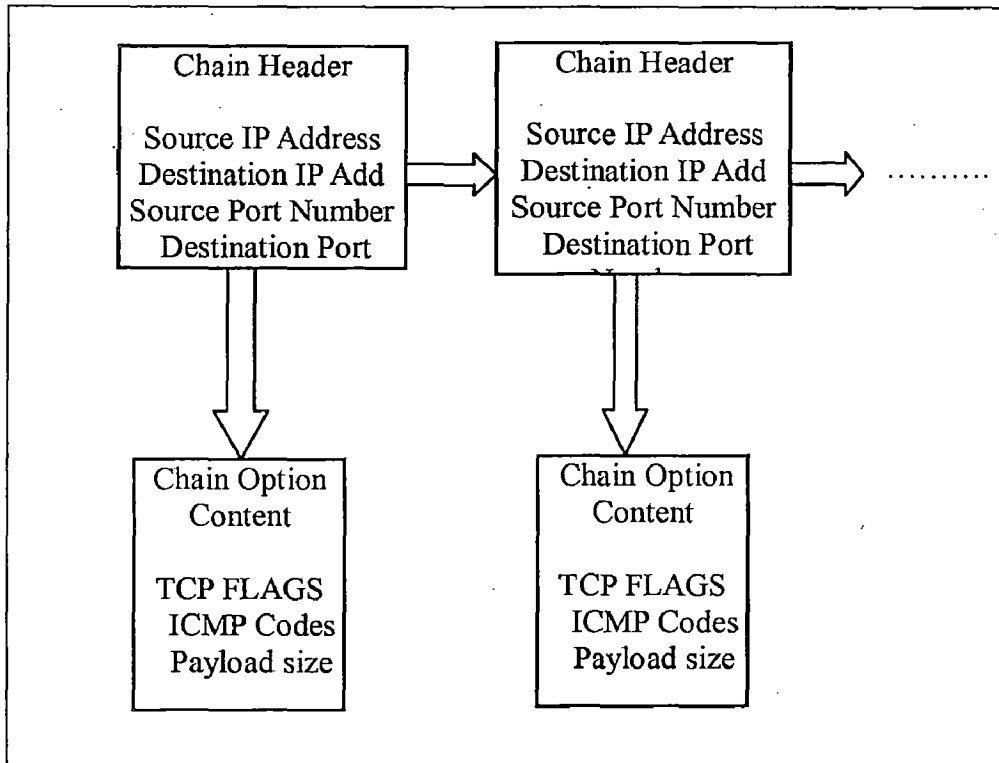


Figure 2.3 Rule Representation of Snort

(d) Logging and Alerting System:

For efficient functioning of Snort, robust logging and alerting system is required. As there are two possible cases after detection engine finishes its processing on the packet. Either the packet is to be logged or dropped. All the logged packets are stored in files either in text format or tcp-dump format.

(e) Output Modules:

These modules are responsible for managing outputs or alerts generated by logging and alerting system. We can directly redirect all the alerts into database rather than in text files so that they can be efficiently analyzed with the help of better user interfaces.

2.5.3 Snort Modes

Snort can be configured to run in four modes [22].

(a) *Sniffer mode*: In this snort simply reads the packets off the network and displays them in a continuous stream on the console.

(b) *Packet Logger Mode*: In this snort simply logs the packets to disk.

(c) *Network Intrusion Detection System*: The most complex and configurable configuration, which allows Snort to analyze network traffic for matches against a user-defined rule set and performs several actions based upon what it sees.

(d) *Inline mode*: In this mode snort obtains packets from iptables instead of from libpcap and then causes iptables to drop or pass packets based on Snort rules that use inline-specific rule types.

2.6 Automatic Signature Generation Systems

The Intrusion Detection System (IDS) used today suffer from several shortcomings in the presence of complex and unknown attacks. It is very hectic to manually update the signature database of IDS. For automating this task various signature generation systems were proposed.

2.6.1 Honeycomb

Honeycomb [24], [25] was a system for NIDS which automatically generated attack signatures. It used longest common subsequence (a pattern matching technique) to find similarities in network traffic which was previously seen on the honeypot logged data. This process is quite hectic and also not efficient. The honeycomb system used LCS implementation. Suffix trees were used to find common substrings in linear time. This system used the Ukkonen's algorithm. It totally depends on LCS between connections. Due to this it generates a large number of signatures. It is also biased towards the content of packet payloads (as in long packet payloads in worm attacks). Hence it is very

effective in generating signatures in worm attacks. However, with shorter byte sequences in payload its performance degrades.

Honeycomb generates the rules based on content matching; hence it considers the packets individually. This results in a large number of rules.

Table 2.1 Rules generated by Honeycomb

ALERT tcp any 0 -> any 0,19977,20233,20489,20745, 21001, 22025,22793,23049,24841,25609, 26121 (msg: "Honeycomb Fri Feb 11 13h00m56 2011 "; ip_proto: "1 flow: stateless;)
ALERT tcp any 0 -> any 0 (msg: "Honeycomb Fri Feb 11 13h00m56 2011 "; ip_proto: "182"; flags: FSR+; ack: 2427870985; flow: stateless;)

2.6.2 Other systems

Pouget and Dacier [23] used the traffic collected by honeypot for network forensics. It used clustering algorithm for finding frequent patterns in the traffic. "Phrase distance" was used to further enhance the clusters. Another system which was proposed by Kim and Karp [26] named Autograph. Like honeycomb it is also mainly used for generating signatures for worm attacks. It used Rabin Carp algorithm instead of LCS. Unlike honeycomb it takes its input from DMZ traces. Earlybird [27] is similar to Autograph in terms of attack class i.e. worm attack and input source but differs in the algorithm used from previous systems as it measures packet content prevalence. Nemean [28] was yet another system proposed by V. Yegneswaran et. al. used the Clustering and automata learning. It was designed for general attack class. But it has a disadvantage that it doesnot have on-line capabilities. The main issues which we came across are the number of attack signatures generated and attack class targeted. Also none of the above systems has been integrated with any existing NIDS. A model of IDS using honeypots was proposed in [29]. Here clustering technique and genetic algorithm are applied on honeypot logged data to find abnormal activities. This approach was not realized on any real time system.

2.7 Alert correlation

Another design issue in building IDS is the proper representation of alerts to security analyst. As IDS generates huge amount of alerts, it is very difficult to analyze them. Various alert correlation techniques are proposed. T. Zang et al. [30], provides a summary of all alert fusion techniques. These techniques mainly use aggregation, verification and correlation. Aggregation combines alerts having same attributes like Source IP, Destination IP etc. within a time window. Then correlation is performed on those alerts to find out the attack patterns. Hence these techniques only help in network forensics and aggregation only focus on grouping of alerts for correlation. No focus is upon the representing alerts according to their importance so that we can stop them timely.

2.8 Research gaps

(a) Many anomaly detection approaches are proposed for detecting anomalies in network traffic but these are not efficient to implement in real time. Hence techniques should be devised to be effective in real time.

(b) Most of the current IDS use one of the two techniques. It is already discussed that there are some problems with each methods. Hence by combining both techniques, efficient IDS can be built. So we augment the existing signature based IDS with some anomaly detection techniques which is the main focus of this dissertation.

(c) Slowing down of system performance when maximizing security is applied. Hence need to trade off between security enforcement levels and performance and usability of an enterprise information system.

(d) While designing IDSES, other security options like firewalls are ignored. Better systems can be built with the collaboration of other security options.

(e) Amount of log data generated by IDS is very large. Hence some type of metrics can be defined so that by keeping all information we can reduce the amount of log data.

(f) Another design issue in building IDS is the proper representation of alerts to security analyst. As IDS generates huge amount of alerts, it is very difficult to analyze them.

(g) In all approaches, the main emphasis on the implementation of particular approach in intrusion detection. But none have taken account the attack properties and behavior in machine intelligence approaches. Better IDS can be built if we take those attack properties into account.

(h) With respect to signature generation systems, the main issues which we came across are the number of attack signatures generated and attack class targeted. Also none of the above systems has been integrated with any existing NIDS.

To fill the research gaps discussed above we are designing a IDS by combining various approaches i.e. signature based IDS (Snort), entropy based anomaly detection method with feature extraction and alert reduction and ranking system and a Real Time Rule Accession (RTRA) technique with the help of association rule mining and honeypot to address the issues of unknown attacks, alert reduction and automatic signature generation.

Chapter 3

Proposed Framework for Intrusion Detection System

3.1 System framework

To address all the issues discussed in research gaps, here we are proposing a framework of IDS. Framework comprises of broadly three independent systems

(a) Automatic signature generation system

(b) Snort based Hybrid IDS

(c) Feature Extraction System

Automatic signature system addresses the issue of manually updating the signature database of signature based IDS.

Snort based hybrid IDS deals with the problem of IDSs encountered with unknown attacks. Also it addresses the issue of better alert management along with better alert and logging system for Snort.

Feature extraction system deals with the issue of ignoring the attack properties while designing IDS. But here we are using this mainly for determining features for Hybrid IDS.

Figure 3.1 shows the overall level 0 design of the proposed framework. In this we are only showing the main systems for this framework of IDS whereas internal details are left for the next section. All systems are autonomous. It means they are designed in a manner that they do need each other for functioning. One system only increases the efficiency and robustness of the other system when integrated with that. Now we will see the functionality of each system.

3.2 Automatic Snort compatible signature generation system (ASSG)

As already pointed out that the main problem with signature based IDS is that their

limited attack signature database. Also regularly updating the signature database manually is very hectic. To deal with this problem, this system is put along with other systems. As shown in figure, this system takes input from the honeypot.

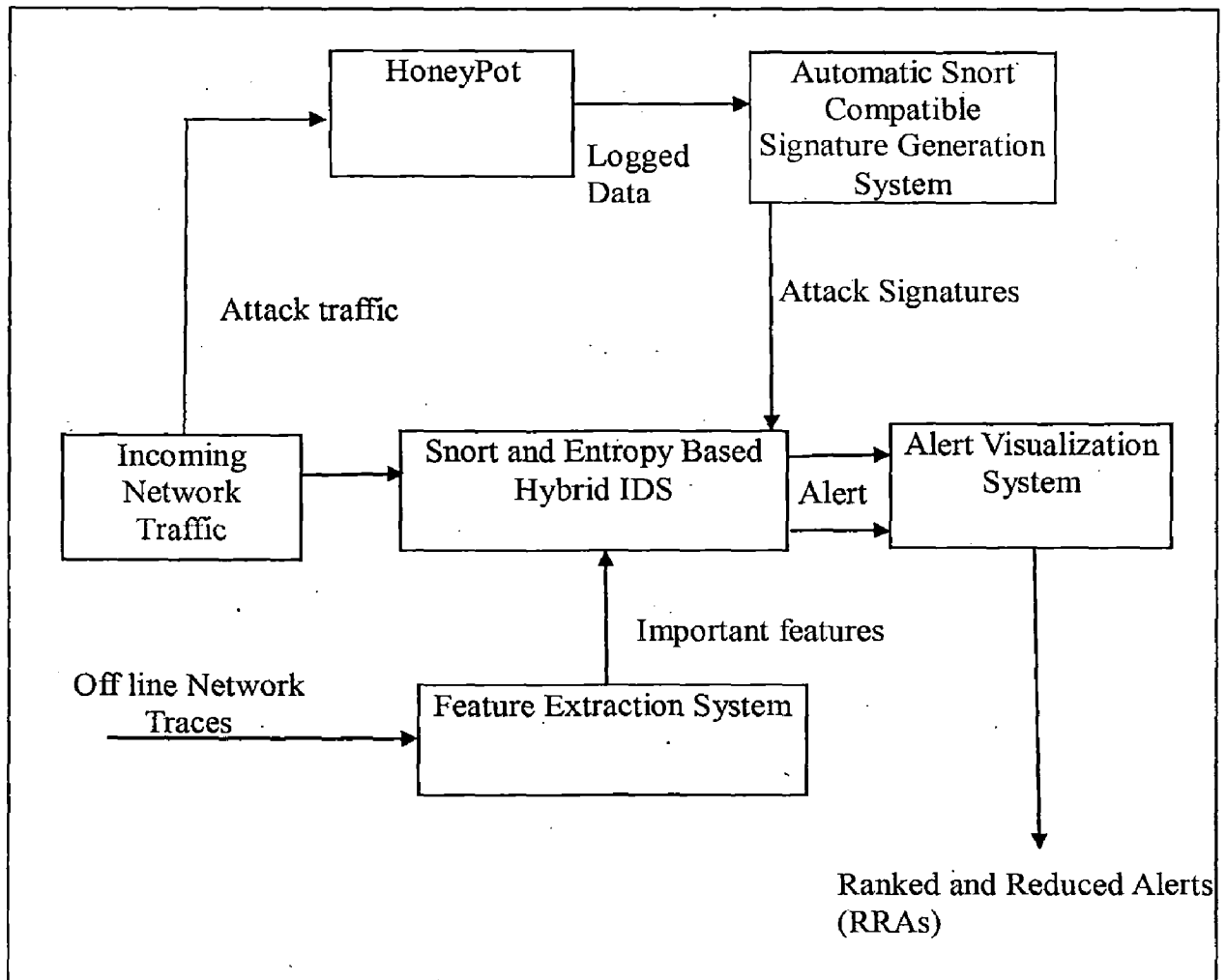


Figure 3.1 Level 0 System design

A honeypot is an information system resource whose value lies in unauthorized or illicit use of that resource [31]. They are computer systems placed on a network to attract attackers, allowing administrators to capture and analyze current attack methodologies and use that information to harden their systems and networks. They are closely monitored network decoys serving several purposes: they can distract adversaries from more valuable machines on a network, they can provide early warning about new attack

and exploitation trends and they allow in-depth examination of adversaries during and after exploitation of a honeypot. They are a resource that has no authorized activity. Theoretically, a honeypot should see no traffic because it has no legitimate activity. This means any interaction with a honeypot is most likely unauthorized or malicious activity. Hence on line traffic which goes into the honeypot is considered to be attack traffic.

For signature generation we need data to generate model for attacks. Actually we cannot generate the signatures until we are sure that these patterns are attack patterns. Hence we have to use some tools which can assure that of attack. And we have already described online traffic which goes into honeypot is considered to be attack traffic. Now all the network traffic which passes through honeypot is logged. Now we have to use that data to generate signatures. As attack data is very large. We cannot analyze them without having efficient algorithms. Data mining comes into picture at this step of this system.

Data mining is used to extract useful information from a huge amount of data. We have already investigated various techniques of data mining for intrusion detection. But they are somewhat different. It means we do not need data mining approaches for intrusion detection here. Instead we have to find interesting patterns by which we can generate signatures of attack. Hence we can use various techniques like association rule mining, clustering for generating those patterns. As there are various pros and cons for every technique, hence we have to decide which algorithm should be used to generate those patterns.

As we need signatures of attack and signatures are of the form “if and then”, hence association rule generation algorithms are in best interest of us. There are various association rule generation algorithms are available like apriori, FP-Growth, eclat and pincer search. They all are different in terms of their style of working. As mainly difference comes into their performance with respect to time complexity. Time complexity matters when data is very large. But later in implementation details, we will see data on which these algorithms have to apply is not large instead it is very small. Hence time complexity hardly matters. So we can use any of the association rule generation algorithm. Any rule generation algorithm works in three steps:

- 1) Candidate item set generation
- 2) Pruning
- 3) Large item set generation

Once we have all rules generated by the algorithm. Our task is which rules we have to take and which have not. This thing depends upon various parameters of algorithm used. As in association rule mining algorithm, there is a factor of support. "Support factor" mainly used in pruning step. As all data in honey pot log is attack, we cannot ignore even a single packet. Hence we have to put factor very small.

After finalizing all rules, we have to insert them in IDS dynamically. This is the most important part of this design. As we have clearly mentioned the main design issue of IDS is off line rule addition. If we do not do this thing, whole this process is useless. For this we are using Real time rule accession (RTRA). In this technique, not all honey pot data is taken at once. Data is taken in parts and rules are generated from them and added in IDS and restart the IDS. Hence all the rules are added in IDS dynamically with very small time lag.

3.3 Hybrid IDS based on Snort and Entropy technique

Another design issue of IDS is unknown attacks. Unknown attack means the attacks of which signatures are not available. As we have already pointed out what an attack signature is. If we change the attack traffic slightly, so it is possible that we are not able to find out that attack due to absence of signature. So it is totally possible that existing attacks have different signatures. Hence we cannot totally rely upon signature database of IDS. Even a slight change in attack pattern will cause IDS to miss the attack.

To deal with this problem, we have to use some technique so that we can detect these types of attacks. For this anomaly detection techniques are used. Actually anomaly detection algorithms formulate a model around a user profile and find out the significant deviation from stored user profile. Various types of anomaly detection techniques we have already investigated in section 2. We are using one of those techniques here. The main problem here is to decide which technique we should use. As in data mining

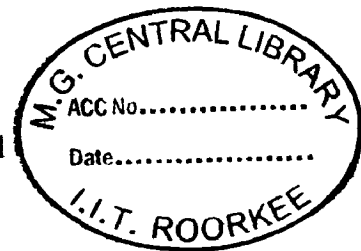
algorithms, we need some kind of labeled data (attack and normal) to use. This type of data is rarely available (KDD). But even if it is available, we need real time data from time to time for training. Each time we have to label it, it is not possible. So we have to use statistical techniques. Hence here we are using Entropy based anomaly detection approach. Actually we are only focused on attack which mainly depend on the traffic statistics like port scanning, DDoS, worm etc. We are not considering attacks like guess password etc. Also attacks like DDoS are more severe than later type of attacks. Entropy is a very good tool for formulating traffic statistics. This system mainly comprises of two sub modules

- 1) Entropy calculation
- 2) Anomaly detection

These both sub modules have their own design issues.

The main design issues with Entropy calculation system are

- 1) Features for which entropy is to be calculated
- 2) Features must be categorical



As entropy of whole network data is of no use. Hence we have to tell the network features for which entropy is to be calculated. This task also is not trivial to decide which network features are important. There are mainly two types of network features.

- 1) Absolute features
- 2) Apparent features

Absolute network features are those which are directly present in the network traffic like Source IP etc whereas apparent features are the features which we have to calculated from network traffic like out degree and in degree of hosts. For these things we have put feature extraction system. Feature extraction system gives input to the Entropy based anomaly detection system (EBAD). As it is necessary to tell EBAD about the features around which statistical model is to be prepared for anomaly detection.

Once the features are finalized around which entropy is calculated. The problem is that the features must be categorical. As entropy is not calculated for continuous attributes, so that we must have some kind of discretization algorithm for this.

Once we have calculated entropy, we have to use some algorithm to find anomalies in data. For finding anomalies in entropy data, mainly two techniques are there

- 1) Wavelet based
- 2) Heuristic based

First technique is based on signal theory. It uses the concepts of wavelets for anomaly detection. Here time series data is decomposed into high, low and mid level frequency components. Low frequency components show the expected value where as mid to high frequency components shows the anomalies and deviations. As an alternative to wavelet based, we have heuristic based, in which we calculate a standard value for mean and standard deviation and find the score of each observation and look for the anomalies. There are also some issues in this method which will be discussed and handled in next section.

After we have calculated anomalies, we have to generate alerts to point the security analyst (SA) along with some information like timestamp, source ip, destination port so that SA can take corresponding action.

We cannot totally rely upon anomaly based module for finding attacks, as they generate lot of false alerts. And also for detecting attacks which are not based on traffic patterns like guess password attack. Hence we must have signature based system.

Hence by keeping all these points in mind, we have designed a system which is the main part of our design which is mainly responsible for detection of all intrusion activities on network level. As shown in fig. 3.1, all the network traffic passes through this system. This system mainly takes inputs from feature extraction system and signature system. This system comprises mainly of two sub systems Snort (signature based IDS) and EBAD.

All the network traffic goes to both systems in parallel. Signature based IDS is responsible for generating alerts for known intrusions where as EBAD is responsible for alert generation for unknown attacks. Signatures generated from ASSG directly go into Signature IDS rule base. Both systems generate alert independently. Hence there is need of some system to integrate them.

So the final output of this system is alerts generated by both systems. We cannot show these both alerts to SA as due to following problems

- 1) Alert redundancy will be there.
- 2) Formats of both types of alerts are different.
- 3) Snort generates huge number of alerts.
- 4) EBAD can give false anomaly points.

To address all these issues, alert unification system is put to integrate them efficiently. Efficiency here denotes the quality and quantity of alerts. Hence this alert unification system is mainly composed of two sub systems Alert reduction system and Alert ranking system. Alert reduction system is mainly responsible for integration of alerts without redundancy whereas alert ranking system is used to give the ranks to alerts. Rank of the alerts means which alerts are most important to SA. So that preventive action can be taken timely. Finally the output of this system is ranked and reduced alerts. The internal details of all these systems will be discussed in next section.

3.4 Feature extraction system:

Feature extraction yet another independent system of this framework. We have already pointed out in research gap that IDSEs are not designed in keeping in mind the properties of different network attack. To address that issue, we have designed this system. The main work of this system is to study the different attack traces and find out which network features are important for a particular attack. Network attacks can be classified into mainly four categories

- *DoS*: Attacker tries to prevent legitimate users from using a service.

- *Remote to Local (R2L)*: Attacker does not have an account on the victim machine, hence tries to gain access.
- *User to Root (U2R)*: Attacker has local access to the victim machine and tries to gain super user privileges.
- *Probe*: Attacker tries to gain information about target host.

Also network features are mainly classified into following categories:

- *Basic Features*: Basic features can be derived from packet headers without inspecting the payload. For example, duration, protocol_type, service, flag, source bytes and destination bytes.
- *Content Features*: Domain knowledge is used to assess the payload of original TCP packets. This includes features such as number of failed login attempts.
- *Time-based Traffic Features*: These features are designed to capture properties that mature over a temporal window e.g. 2 seconds. One example of such a feature would be the number of connections to the same host over the 2 second time interval.
- *Host-based Features*: These features utilize a historical window estimated over the number of connections (100). Hence these are designed to assess attacks, which span intervals longer than 2 seconds.

The main design issues with this module are

- 1) Preprocessing
- 2) Feature discriminating algorithm.

Before proceeding to the main feature discriminating algorithm which is mainly responsible for generating final important features, we have to use some type of preprocessing on that data.

As various discriminating algorithms are available, here we are using entropy based. As it is simple and also as previously discussed that it can only be applied on categorical attributes, we have to use discretization algorithm in preprocessing step.

For discretization algorithm, various algorithms are available like equal frequency based, entropy based. We are using equal interval based as entropy based discretization is very time consuming and also there is no point of use as we are using it in discriminating algorithm as in both area, nearly same technique is used.

Hence this system can be used to study any off line network traffic with slight modifications which can provide important information to build better IDSs.

Chapter 4

Detailed Design and Implementation

4.1 Signature generation module:

Design of this module includes major three modules. First is data collection by honeypot, second is mining the data to generate attack rules and last is the RTRA (real time rule accession) in Snort. But all these modules are incompatible. E.g. honeypot log file cannot be directly given as input to the mining algorithm. So to eliminate these problems, interfaces are designed in between these modules. Similar interface is designed for rules generated by Apriori and rules added in Snort. A log file is very large so in order to efficiently process these log files DBMS is used.

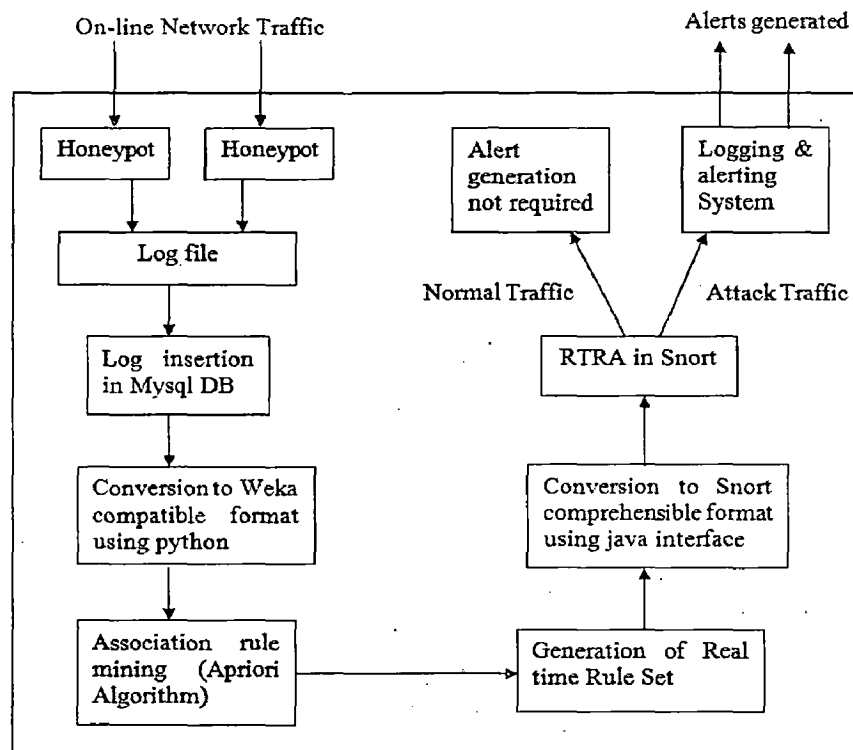


Figure 4.1 System design for signature generation system

4.1.1 Attack traffic capturing module

For applying association rule generation algorithm we need attack data. Honeypots are used to collect that data. These are implemented using honeyd tool. A virtual topology is created to deceive the attacker using honeyd tool. Whenever attacker tries to attack this network, the data is logged. This logged data need preprocessing as described in previous section. Hence we have used PYTHON script for that preprocessing. Now logged file is in the form of records. Finally all this data is inserted into MYSQL for further processing.

4.1.2 Apriori association rule mining module

Association rule mining algorithm is used to generate frequent item sets from that data. Weka tool is used for that algorithm. Weka takes a special file format named as arff as input. Hence Java program is deployed as an interface between Mysql and Weka. This program first discretizes each column of database because apriori can be applied only to categorical data. All the attributes are already in discrete form. But we cannot take all port numbers as different classes as it will make 65535 classes. So we found all distinct port numbers from database and assigned them as individual discrete classes. Same thing has been done with all other attributes. With the help of above procedure, the interface generates a compatible arff file for Weka. Weka employs various association rule mining algorithms like apriori, predictive apriori etc. We have employed apriori due to its less time complexity.

4.1.3 RTRA in Snort module

Rule set generated by Weka has to be converted into the form so that rules can be comprehended by Snort. For this another python script is implemented as an interface. Not all Weka rules can be added to Snort because suppose if some of the Weka rules doesn't have any protocol field then that rule is not added to Snort. Finally rules are added dynamically in Snort in real time. This processing cannot be done on whole honeypot log at once because log is increasing continuously. So we have decided to follow this process repeatedly on a certain number of records at a time. A C program is

deployed between actual honeypot log and other modules. This program takes certain number of records and passes the data as a file to Shell script which comprises commands to run above modules. One main advantage of this design is that Snort continuously works in parallel to all this work. Hence no attack is missed which is already present in Snort rule base. But when rules are generated by our system, they are added in the rule base of Snort, and are used for detecting attack thereafter.

4.2 Feature Extraction Module

4.2.1 Preprocessing

Dataset used in feature extraction can be very huge. Hence employing a efficient data structure in very important. Hence I have used DBMS to store all the connection records. At first feature important with respect to all classes are to be found out. Here we are using KDD dataset. For this, all the attacks are distributed into five categories. Before applying, feature analysis steps, we have to convert continuous attributes into discrete attributes. A lot of algorithms are available like decision tree, Error, Entropy, Equal frequency method. Here I have used equal frequency interval method.

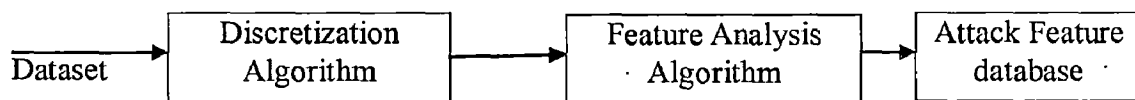


Figure 4.2 System design for feature extraction system

4.2.2 Discretization algorithm

Here by employing equal frequency intervals method, continuous features are partitioned into equal sized partitions.

- Decide the threshold value for partitioning.
- The feature space is partitioned into arbitrary number of partitions where each partition contains the same number of data points. It means, the range of each partition is adjusted to contain N dataset instances.
- If a value occurs more than N times in a feature space, it is assigned a partition of

its own. Threshold must be properly decided.

- After creating all classes, we have to change all the values of all attributes in the database.

4.2.3 Feature analysis algorithm

After completing the step of preprocessing, we have to quantify the importance of each attribute or feature for the discrimination of major attack classes. This step is broken into two step.

- 1) Feature analysis with respect to all classes of attacks.
- 2) Feature analysis with respect to individual classes.

With the help of first, we are able to determine, which features are most relevant with respect to all classes of attacks. With the help of latter, we are able to determine relative importance of each feature with respect to a particular attack class.

Algorithm Used:

Here Information Gain is used as the method by which we quantify the discriminating characteristic of a feature. Information gain uses the concept of Entropy. Entropy can be defined as the amount of information stored in an event.

- Let S be a set of training set samples with their corresponding labels. Suppose there are m classes and the training set contains s_i samples of class I and s is the total number of samples in training set. Hence expected information needed to classify a given sample is calculated by

$$I(s_1, s_2, \dots, s_m) = -\sum (s_i / s) \log_2(s_i / s) \text{ where } I = 1, 2, \dots, m \quad \dots eqn(4.1)$$

Now this is the total information needed to classify a given sample.

- Now for each feature F with values $\{ f_1, f_2, \dots, f_v \}$, it can divide the training set into v subsets $\{ S_1, S_2, \dots, S_v \}$ where S_j is the subset which has the value f_i for feature F. Furthermore let S_j contain s_{ij} samples of class I. Collect all these information.

- Calculate the entropy of feature f .

$$E(f) = \sum (s1j + s2j + \dots + smj) / s * I(s1j, s2j, \dots, smj) \quad \dots eqn(4.2)$$

- Find out the information gain of feature f.

$$\text{Gain}(f) = I - E \quad \dots eqn(4.3)$$

Hence in this way we are able to quantify each features importance with respect to all classes. A higher information gain means higher discriminating characteristic which leads to more importance of a feature with respect to classes.

- *Feature Analysis with respect to all classes:*

Here we apply above algorithm on all attributes considering all classes. In general in this way we are able to determine the important features which we have to take during analysis of attacks.

- *Feature Analysis with respect to individual classes:*

Here we employ a binary discrimination. It means, a feature is considered to be in class if it has the same label otherwise it is considered to outside. With the help of this we are able to determine the discriminating characteristic of each feature with respect to a given attack class.

4.2.4 Result analysis system

As we get all relative discriminatory characteristics of all features with respect to all classes. We must first put them into a graphical representation so that they can give good understanding of their behavior.

Here bar chart is used for the graphical representation. After obtaining this we first have to validate the results by examining the discriminatory characteristics and their actual contribution in a particular network attack.

02/27-11:47:16.829663 ARP who-has 172.17.12.46 tell 172.17.11.248

As these statistics can't be directly used in entropy calculation module. Also there is various information which are of no use to us like ARP packets. Hence we have to convert it into a suitable format according to requirement. As shown in fig. 3.1 it has got some input from Attack-Feature database. Attack-Feature database will tell which features we have to extract for further processing.

For efficient functioning, we have used python script for this purpose. As python works fast on text processing. Actually format is somewhat typical; hence we have used 6 small python scripts for this purpose. After going through all this output will be

```
2010-02-14 11:47:16.547297/172.17.14.94/138/172.17.255.255/138
2010-02-14 11:47:16.863052/172.17.11.222/137/172.17.255.255/137
2010-02-14 11:47:16.978951/172.17.12.231/50747/69.63.189.39/80
```

'/' acts as separator between fields. All python scripts are run by a shell script 'inter.sh'

4.4 Entropy based detection module with Suspect Index

This module is responsible for calculation of entropies of different features given by attack-feature dataset. Entropy has been already defined in previous design. We are using entropy based technique because attacks like port scanning, DDoS change the pattern of network traffic. They make network traffic more uniform or more random. Hence entropy based technique is used. As signature based systems use particular signatures for attack detection, hence these are ineffective when encountered with new patterns of attacks but with the help of anomaly based module in our system we can eliminate this problem. Entropy is used as a measure of anomaly in this module. Here we use the concept of normalized entropy.

Normalized Entropy:

Let N_0 be the number of distinct items present in a given measurement interval. The entropy attains its minimum value of zero when all the items are identical and its maximum value of $\log(N_0)$ when each item appears exactly once. Since different measurement intervals might observe a different number of distinct items, we normalize

H to be between zero and one by computing the normalized entropy: $H / \log N_0$. This measures the relative randomness within each measurement interval and allows us to quantitatively compare entropy values across time. For the remainder of the discussion we will use this definition of normalized entropy:

For any anomaly based algorithm needs some training data and learning methodology to fine anomalies. Similarly here also we need a learning algorithm

4.4.1 Learning algorithm

We consider a heuristic technique adapted from prior work [23, 24]. The high-level goal is to estimate the mean and standard deviation of the time series signal using historical data. For each future observation, we compute a deviation score:

$$\text{Score} = |\text{Observation} - \text{Mean}| \quad \dots \text{eqn}(4.4)$$

This score captures how far away from the mean value a particular observation is, expressed relative to the standard deviation. We flag an anomaly whenever any observation has a score greater than some threshold α . If the mean and standard deviation are calculated with historical traffic data that contains large anomalies, then these values are likely to over-estimate the true statistics. Therefore, this heuristic may miss anomalies in future observations, because the model of traffic accommodates more anomalies than it should. To avoid this bias, an iterative cleaning technique is used for learning the mean and standard deviation given possibly noisy training data. The approach works as follows.

- 1) In each iteration, we compute the mean and the standard deviation. For the current iteration, we find anomalous data points, i.e., those that are greater than $\alpha = 3$ standard deviations away from the mean.
- 2) We remove these anomalies from consideration for further iterations.
- 3) The iteration continues until the mean and standard deviations obtained are stable, meaning that values do not differ significantly across subsequent iterations.

This entropy based anomaly detection module can be applied in two ways.

4.4.2 Long term statistics based

In this method, historical data is used for finding these values (mean and standard deviation). It means we have to collect long term statistics of network traffic and build a model around that. This whole process works off line. Hence this method is biased towards historical data as there are different traffic pattern properties at different timings and different locations. Hence there is a need of repeating this process again and again to get best utilization. This problem occurs in experimenting these IDSes. We have collected Hostel data and Institute data. Both have very different properties. There is also another method in which we can do anomaly detection in real time.

4.4.3 Short term statistics based

This technique is quite similar to learning stage. Here we take a second time window (n) which is large from previous time window (m). Hence we point the anomalies in a particular m window with respect to its deviation in n . Hence we calculate both parameters mean and standard deviation considering n as historical data. This method creates problem when huge part in real time data is itself anomaly hence giving huge number of false positives. Also this lags behind in time as we have to wait for n for detecting anomalies in m .

As far as the implementation is concerned, learning algorithm is written in Java. Also MYSQL is used as the interface between Information extraction module and anomaly detection module so that efficient processing can be done. We have used the long term statistics based technique.

4.5 Signature based IDS and alert generation module

An intrusion detection system cannot be totally rely upon anomaly based module due to its limitation. Hence we are using Snort as its signature based system. A parallel Snort system is put with anomaly based module. As entropy based module can only detect attacks that change the network traffic pattern. Hence to detect other attacks we have to use a signature based system. For increasing the performance of system, a robust alert and logging system is integrated with Snort. With better alert and logging system we can

have improved alert visualization and management. Once alerts are generated by both systems, they must be integrated.

Snort is used as the Signature based IDS. ACID and Barnyard are used to provide robust alert and logging system.

4.6 Alert unification module

Here main power of our system lies. As snort generates huge number of alerts which cause problems to security analyst to analyze the results, alert unification module reduces the number of alerts generated by Snort and also classifies the alerts into some categories which help the analyst to better analyze the result. Anomaly based module gives the time stamps between which anomalies are detected. Now the suspect index is calculated for different features within that time stamps. Suspect index (SI) is the measure of how a particular feature is contributing to that anomaly. Once the suspect indexes of features are calculated, they are passed to alert unification module. Alert and logging system of Snort gives the full information corresponding to each alert to unification module. Each alert comprises of Signature-id, Src_ip, Src_port, Dst_ip, Dst_port, timestamp and other information. Here we use some threshold for suspect index. All alerts given by Snort within same timestamp as the anomaly based module and particular feature having valid suspect index (suspect index > threshold) are unified. We have used classifiers for alerts. Alerts can be of three types.

- 1) Alerts detected by Snort and Entropy based module
- 2) Alerts detected by Snort only
- 3) Alerts detected by entropy based module only.

Final alerts are shown to analyst according to their importance i.e. Snort and Anomaly based then Snort based, and then Entropy based. Entropy based can be moved up if deviation in network traffic pattern is very high.

Chapter 5

Experimental Results and Discussions

5.1 Signature generation system

This section shows results of various modules for automatic signature system.

5.1.1 Data logging and system parameters

The system was developed in Ubuntu 9.10. The various parameters used for experiment and analysis have been shown in table 5.1.

Table 5.1. Various parameters used for system analysis

S.No	Parameter		Value
1	No. of virtual systems created in honeypot		10
2	No. of log records considered at a time		1000
3	Support value in Apriori	Upper bound	1.0
		Lower bound	0.05
4	Confidence value in Apriori	Upper bound	1.0
		Lower bound	0.01

We have to consider a number of records logged by honeypot to generate rules. As soon as it reaches 1000 the process of RTRA starts. It calls a shell script by which whole of this process is accomplished. We chose 1000 because the rules generated were sufficient as well as alert generation was quicker. The system waits for these numbers of records to be logged before generating rules for the first time and alerts subsequently. A lesser value does not suffice to cover attack patterns. As every record logged by honeyd is significant, hence we chose a range starting from a very low value of support and confidence.

The shell script then generates a rule set which is added in Snort dynamically. After adding rules, Snort is restarted hence it is able to formulate a rule chain of newly added rules. This thing makes whole IDS capable of dynamic rule accession. This process

continues until all log records are read. Output of association rule mining algorithm depends upon various parameters like support, confidence, upper and lower bound on support and confidence, number of rules required. System is tested on these different parameters which have been shown in results.

Finally, Snort rules generated by the frequent item sets of Weka by java interface are shown in Table 5.2.

Table 5.2. Snort rules generated for RTRA

alert tcp 192.168.111.204 50953 -> 192.168.111.105 any (msg: "HoneyPot Detected Attack" ; sid: 100005;)
alert tcp 192.168.111.105 any -> 192.168.111.204 50953 (msg: "HoneyPot Detected Attack" ; flags: SA; dsize: < 60 ; sid: 100024;)

First Snort rule means that generate the alert whenever there is a tcp packet from 192.168.111.204 and 50953 port to 192.168.111.105 tagged with 100005 signature id. Similarly flags field in second rule denoted the set flags in tcp packets. Sid is assigned to each rule of honeypot so that they can be uniquely identified. After adding these rules, Snort must be restarted so that changes can take effect. Snort is restarted using sending SIGHUP signal to process.

ZENMAP tool is used for attacking the honeyd. Slow comprehensive scan is used. With this UDP scan, TCP SYN/ACK scan, ICMP echo, ICMP timestamp, OS detection, version detection, etc. can be done.

5.1.2 RTRA without optimization:

The results shown by above experiments have been depicted using Figure 5.1 and Figure 5.2. Once a stream of records (1000) is considered, the frequent item sets generated by Weka and rules added to Snort are shown in the form of a graphs. Figure 5.2 and Figure 5.3 show the values when number of attributes considered in frequent item sets in Apriori Algorithm is 6 and 7 respectively. The peaks in the graphs show the higher intensity attack while the troughs show lower intensity attack.

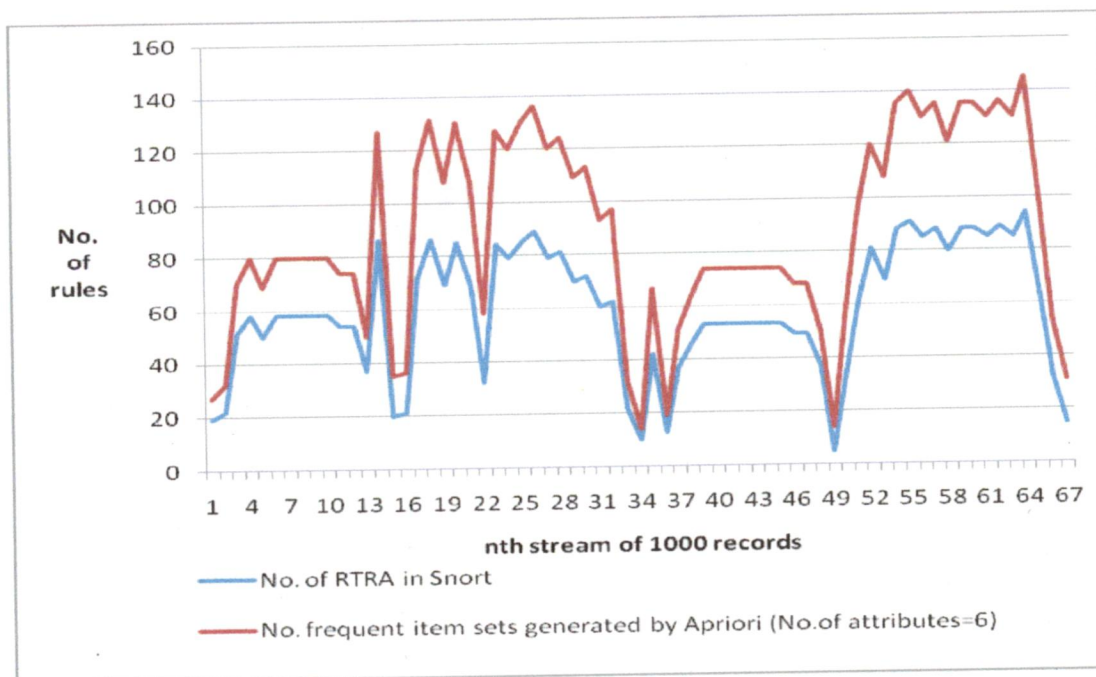


Figure 5.1 Rules generated by Weka and accession in Snort against successive stream of attack records (number of attributes considered in frequent item set is 6).

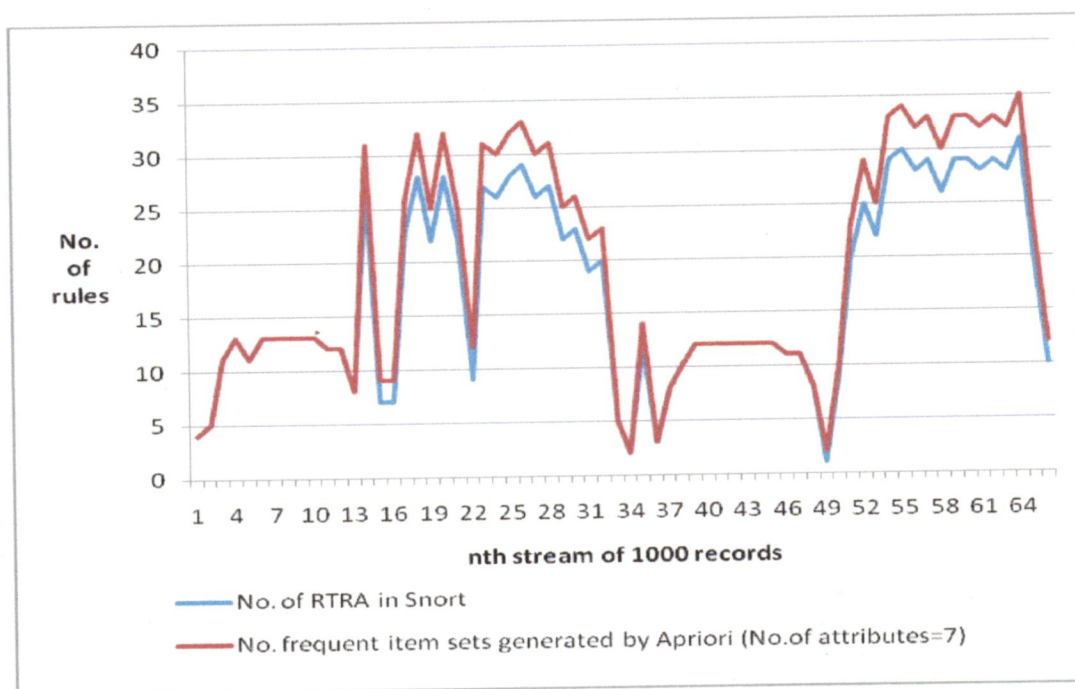


Figure 5.2 Rules generated by Weka and accession in Snort against successive stream of attack records (number of attributes considered is 7)

In each graph, the difference between number of frequent item set generated by Weka and number of rules added to Snort is due to the fact that we cannot convert all frequent item sets into Snort rules. For example, if any frequent item set does not contain any protocol

field, we cannot convert that set to a meaningful Snort rule. Hence only meaningful sets can be converted to Snort rules.

The number of rules in Figure 5.1 is very higher than Figure 5.2. This variation in number of rules in the graphs is due to the difference in number of attributes considered in frequent sets. The relevant number of attributes logged by honeyd is seven. If we consider all the attributes the rules generated are less but of high value. While decreasing the number of attribute to six even though the number of rules increase but redundancy creeps in.

Alerts were generated in both the cases but numbers of rules fired in the case of seven attributes were less than the case of six attributes. Hence the efficiency of our IDS will be best when considering all seven attributes.

5.1.3 RTRA with optimization:

In Figure 5.3 we can see that the number of rules added to Snort are very less as compared to the number of rules obtained by Apriori association rule mining. This is due to the redundancy in rules generated by Apriori. Once the first set of rules is added in Snort which is non-redundant, the rules which repeat in the subsequent processing are removed. Hence, in the graph we see many a times that the rules accession in Snort is nil. This step not only removes redundancy in rules but also improves computational efficiency of the overall system. As the number of rules decreases the rule chain formulated by Snort decreases and hence the matching of traffic with this rule chain becomes more efficient. Alerts are generated in both the cases but numbers of rules fired in this case (Figure 5.3) are less than the previous cases (Figure 5.1 and Figure 5.2). Hence the efficiency of our IDS will be best after applying redundancy removal procedure.

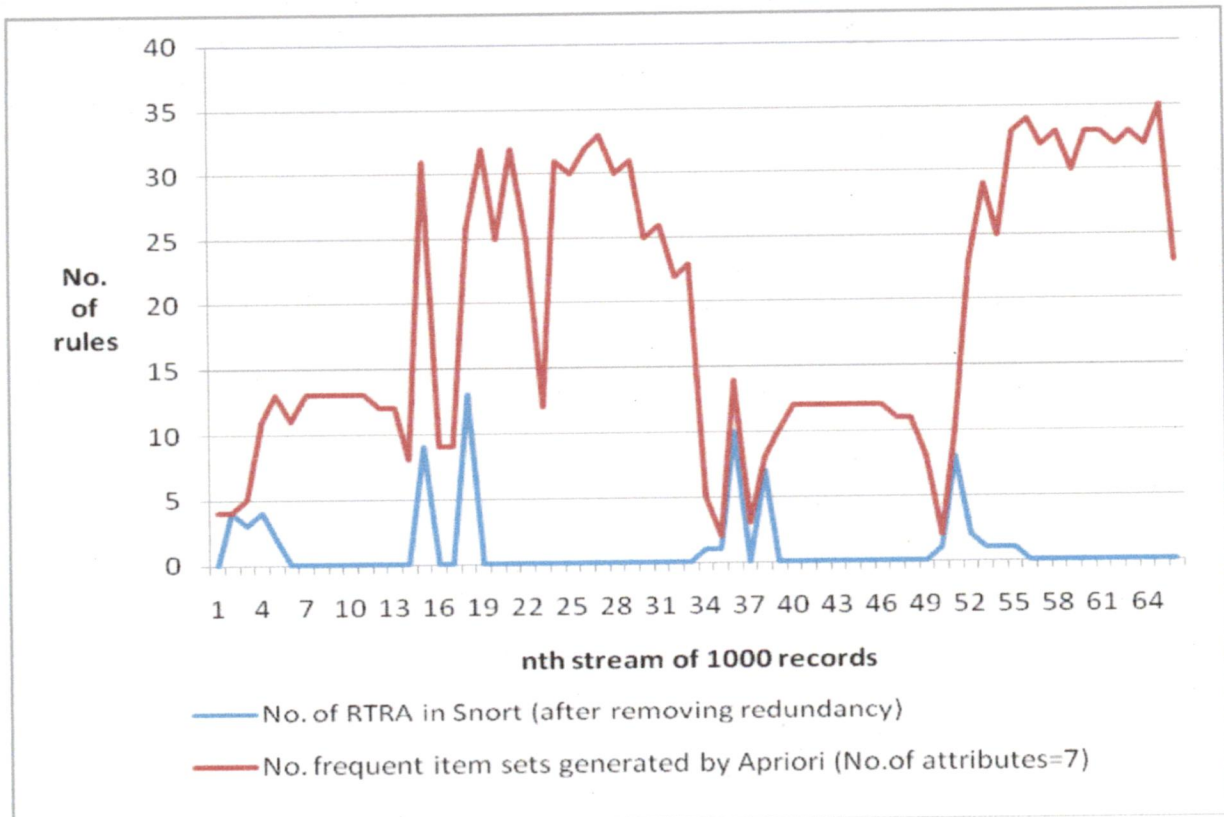


Figure 5.3 Rules generated by Weka and accession in Snort (after removing redundancy) when number of attributes considered in frequent item set is 7.

5.1.4 Comparison with Honeycomb

Table 5.3 shows a couple of alerts generated by our system corresponding to the rules in Table 5.4. These alerts show the timestamp when the attack was detected, the rule Id (100005 and 100024) and the rules. These alerts verify that our system is capable of detecting the attack with the help of RTRA. Hence, this shows that the rules added in real time in Snort rule base are actually detecting the attack. This makes it capable of detecting novel attacks also.

Table 5.3 Alerts generated by our IDS

<pre> alert tcp 192.168.111.204 50953 -> 192.168.111.105 any (msg: "HoneyPot Detected Attack" ; sid: 100005;) </pre>
<pre> alert tcp 192.168.111.105 any -> 192.168.111.204 50953 (msg: "HoneyPot Detected Attack" ; flags: SA; dsize: < 60 ; sid: 100024;) </pre>

Table 5.4 shows the comparison of honeycomb with our system with respect to number of rules generated. For comparison, we have used three types of port scanning attacks performed by Nmap tool. “Quick scan plus” scans fewer ports compared to other two. It probes open ports to determine service/version info. “Intense scan plus UDP” probes on some well-known ports in addition to Quick scan. Slow comprehensive scan includes more number of well-known ports along with ICMP scans. Out of a number of attacks we choose these three because they covered most of the scanning attacks performed by the adversaries.

Table 5.4 No. of rules generated

Type of Attack by Nmap	Honey-Comb	Weka	Proposed System
Quick Scan plus	1200	230	37
Intense Scan plus UDP	4321	190	41
Slow comprehensive scan	2206	154	23

The results show that the rules generated by honeycomb are very large in comparison to our system. The logfile considered was same for both the systems. For “quick scan plus” the number of rules by proposed system was found to be only 3.08% of the total number of rules generated by honeycomb. Honeycomb generates the rules based on content matching; hence it considers the packets individually. This results in a large number of rules.

Table 2.1 shows the rules generated by honeycomb on the logfile considered in our experiment. Albeit, the number of rules generated by our system are fewer but they are complete to detect the attacks that follow. Hence, proposed system generates non-redundant rules which suffice to cover all attacks. The main problem with rules generated by honeycomb is that they are not compatible to Snort. It means that they cannot be directly added to the rule base of Snort IDS. This is due to the absence of fields like Sid. Our system takes care of compatibility issues. It starts Sid value at 100000 which is assigned by Snort to user defined rules. Another limitation we come across is that honeycomb does not generate any ICMP rule on the logfile considered while proposed

system takes care of this too.

5.2 Feature extraction with KDD

For validation of our feature extraction system, we have used KDD dataset for feature analysis. The attacks are mainly classified into 4 major categories in KDD. The statistics of KDD are shown in Table 5.5. Each major category of attack contains several attacks.

DoS: This category includes Smurf, Neptune, Back, TearDrop, Ping of Death (PoD), and Land attack.

Probe: This category includes Satan, ipsweep, portsweep and nmap.

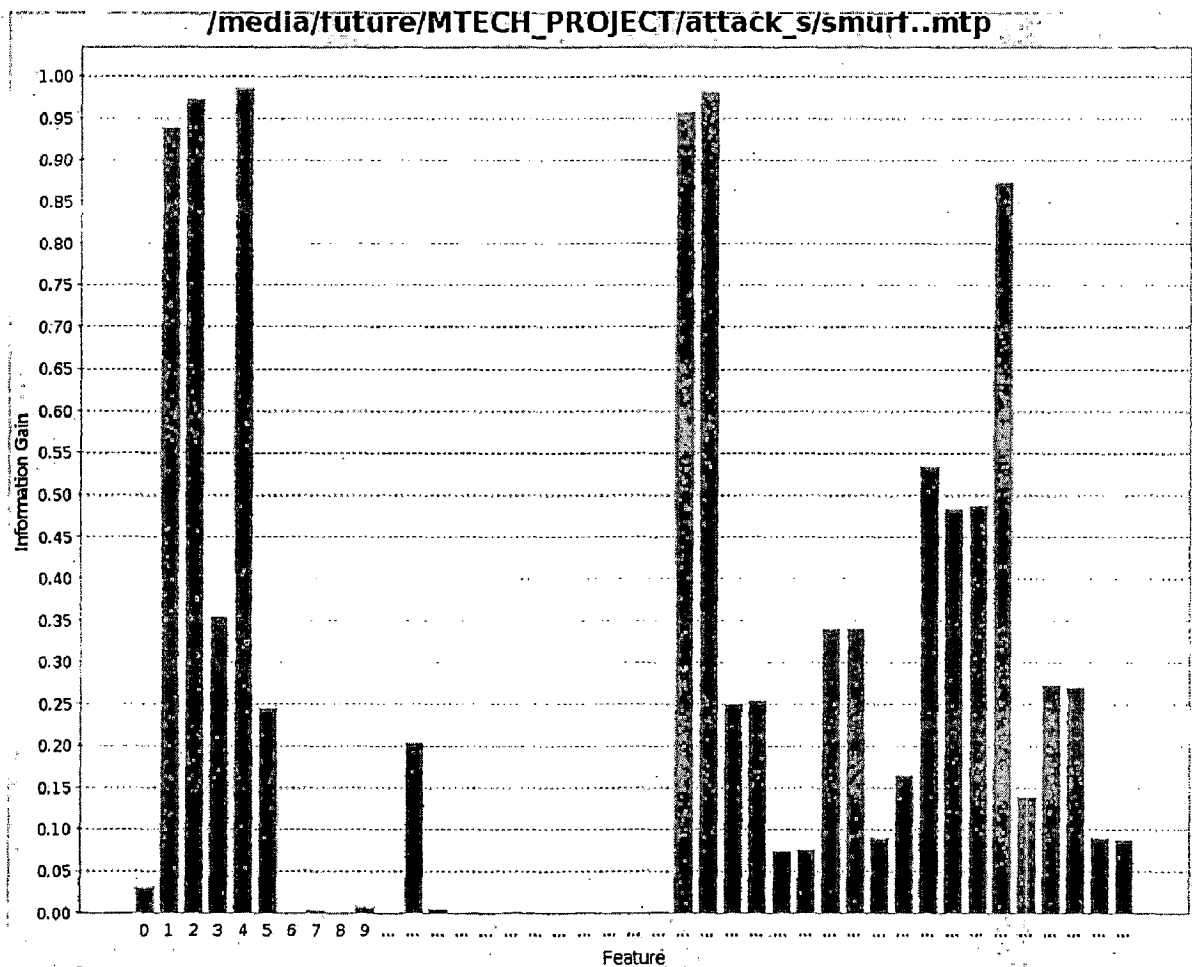


Figure. 5.4 Information gain of various features of Smurf attack

R2L: In this, warezclient, guess_password, warezmaste, imap, ftp_write, multihop, phf,

U2R: This includes buffer_overflow, rootkit, loadmodule and perl

Table 5.5 KDD Statistics

Attack Class	No. of Records
DoS	391458
Probe	4107
u2r	52
r2l	1126
normal	97277

Figure 5.4 shows the importance of various features of KDD in classification of any connection as Smurf attack. This feature extraction part already covered in project. So, here we will only focus on the results with respect to its inclusion in hybrid IDS. As it is shown in the graph most important attributes are 1, 2, 4, 22, and 23. Hence mainly service, count is the main features which are very useful in discriminating a connection from other records. Also analysis of other results shows that service is very important feature. Source IP, destination IP is already very important feature (trivial). Hence in this attack feature database, we have only taken Source IP, Destination IP, Source port, destination port along with timestamp. But mainly we have focused upon Source IP for entropy based anomaly detection. But on the basis of results obtained from this system, we can take important features for entropy calculation.

5.3 Hybrid IDS

5.3.1 Data collection

It is very important part of our dissertation because without valid data, we can't validate our system. As this type of data for testing purposes rarely available, hence we have to collect our own data for testing. For this we have tried to collect data at two places. First is our ISL lab and second is in the hostel. Data collection also posed some of the problem. As data collected in lab does not show the proper randomness of data due to lab conditions and users of the system. Hence entropy values are low showing less randomness of data. Hence we have tried to do that thing in our hostel. We have started

snort in packet sniffing mode with command

```
snort -v -d -e -i wlan0
```

wlan0 is the interface (wireless LAN) on which packet sniffing is done. We have collected the data from 11:48 AM to 7:20 PM. And within this time, port scanning attack is done from system 172.17.12.231 at time 11:49 and 7:19. All this data is stored in a file called "data.txt". Total data collected is 77.4 MB.

In parallel to this another instance of Snort is run on the same interface. Here Snort is run in NIDS mode. So that we can detect those attacks and validate our system. All snort alerts generated are stored in two files "alert.txt" and also into mysql database through Barnyard utility.

5.3.2 Entropy calculation

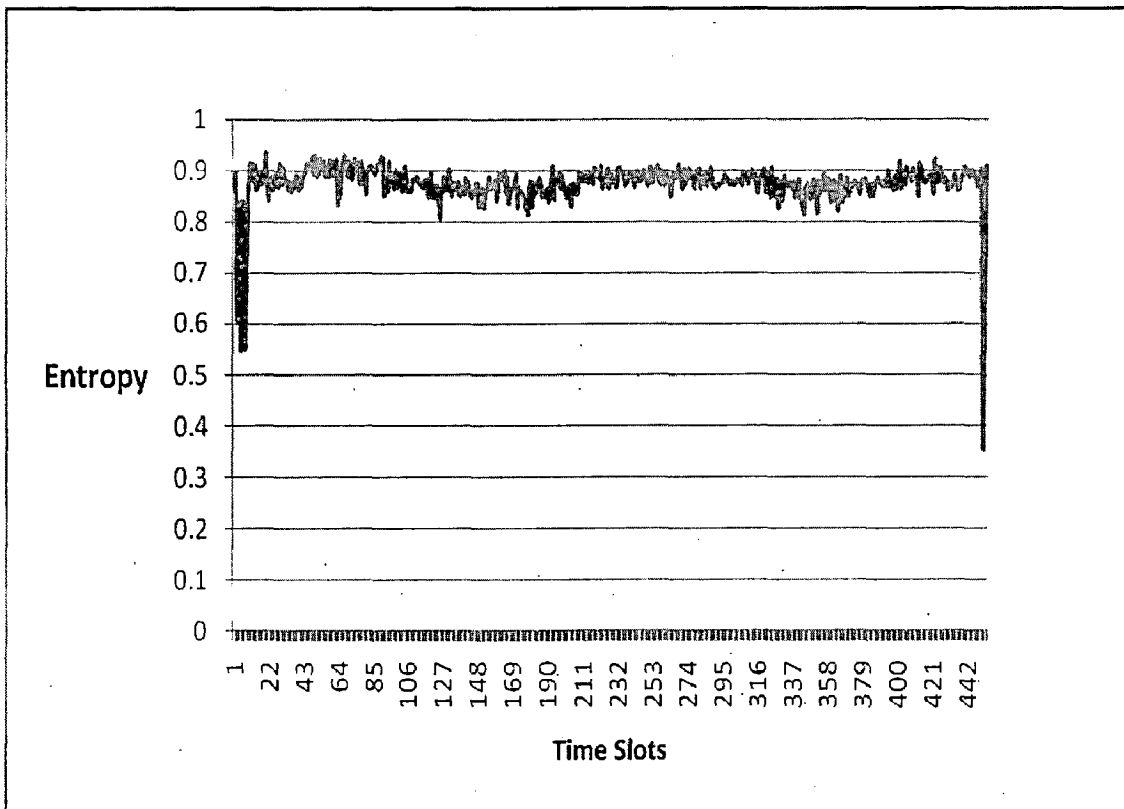


Figure 5.5 Graphs representing relation between Entropy and Time Stamp for feature Source IP

First we have to prove the validity of entropy based module. For this network traffic of Institute Hostel is used. Institute is also used as the historical data. Figure 5.5 and Figure 5.6 give the entropy change in different features of the network traffic of a particular day. Graph is between entropy in a fixed time window (e.g. 60 seconds). As we have seen in the figure, Source IP is most uniform and entropy is very high most of the time due to randomness of data. But here there are major drops in the entropy values which show the drastic change of traffic patterns due to port scanning attacks is done on different systems. Other features are also showing the deviation in entropy of that feature value but not very uniform.

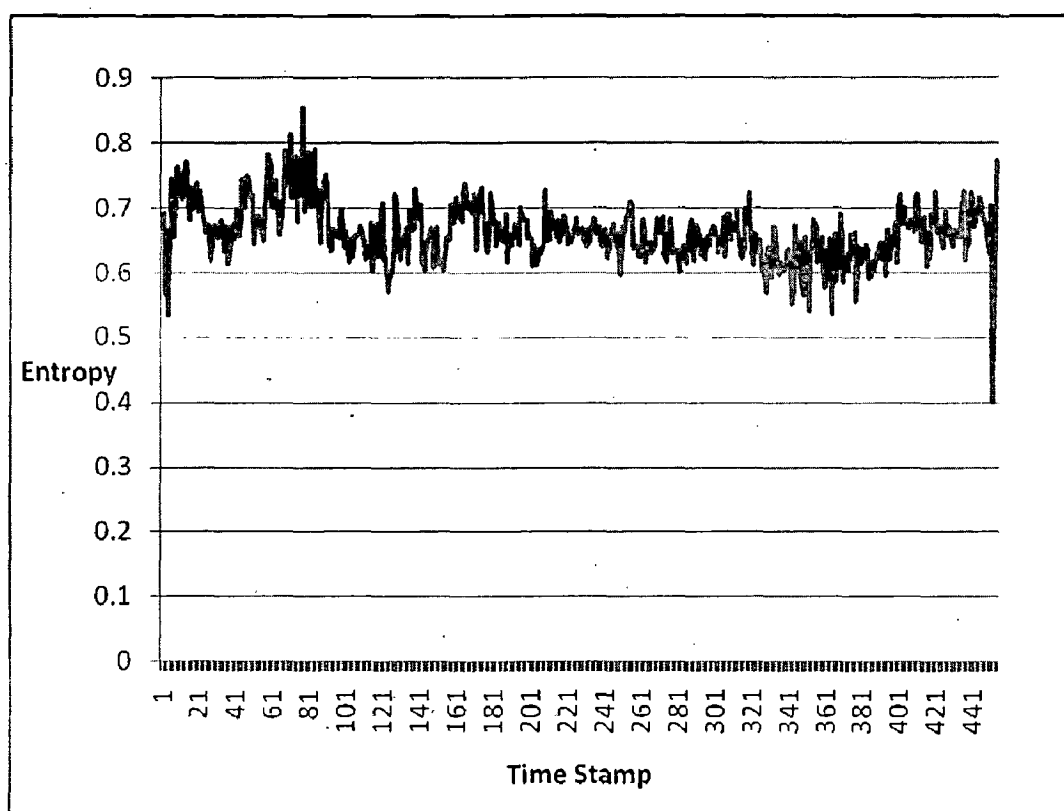


Figure 5.6 Graphs representing relation between Entropy and Time stamp for feature Source Port

This is due to property of port scanning attack which is done. So we first train our entropy model on this data find the threshold values for anomalies. As it is the most difficult thing to decide the threshold factor so we have must have data for deciding this. Threshold factor means deciding mean and standard deviation value with the help of historical data. After finding threshold values, we will check each observation with respect to these values.

Table 5.6 Relation between threshold and no. of anomalous point

Feature	No. of Anomalous Point Detected when t =2	No. of Anomalous Point Detected when t=3
Src_IP	96	9
Dst_IP	94	1
Source Port	96	4
Destination Port	71	6

Table 5.6 shows the drastic impact of threshold values used for anomalies detection. Number of anomalous points drastically increases when it is decreased to 2 from 3. This causes a lot of points to be declared as anomalous points

5.3.3 Suspect Index of hosts

Table 5.7 shows the results when suspect index is calculated for different feature values within a single time window.

Table 5.7 Feature values and their Suspect Index

Feature	Suspect Index
172.17.12.231	0.85
172.17.14.25	0.31
172.17.12.236	0.29
172.17.12.49	0.27
.....

As at five points anomaly is detected, we calculate which feature is contributing most in

anomaly. Here we are calculating suspect index for Src_ip but it can be used for other features also. As results show that for each anomaly point, it is showing all suspect indexes. IP address 172.17.12.231 has highest suspect index with in a particular time window as attack is done from this IP address. All other IP addresses have very low suspect indexes. Hence we have incorporated a threshold value for this also. Here it is taken as .50.

5.3.4 Alert reduction and ranking system

Table 5.8 shows the number of alerts generated by Snort. Now all the Snort alert in the same time window in which anomaly is detected and having same value as feature value for which anomaly is detected is grouped into one because Snort generates huge number of alerts between same pair of systems during attack and some of them can be false.

Table 5.8 Comparison between Snort and Proposed System

Time-Stamp	No. of Snort Alert	No. of Unified Alert	Alert Rank
11:49:16-11:50:16	5	1	1
13:51:16-13:52:16	0	0	3
19:18:16-19:19:16	5	1	1
19:19:16-19:20:16	18	1	1
.....

Now we have verified with the anomaly based module that it is attack hence classified as class1 attack. Similarly attacks which are not verified by Anomaly based module are given the class2 rank and similarly class3 rank is given to those that are not verified by Snort. Class1 is the highest rank alerts. This type of ranking system of alerts gives the analyst to better analyze the result.

Chapter 6

Conclusions and Scope for Future Work

6.1 Conclusions

In this dissertation, an Intrusion detection system was designed and implemented with the help of Snort (Signature based system) and Entropy based (Anomaly based) system. The anomaly based module was validated and an alert reduction and ranking system was designed for better management of alerts.

Also efficiency of proposed IDS was improved by developing automatic signature system. This system was developed using honeypots and association rule mining techniques which detected attack traffic in the network timely and effectively. The honeypot logged the various activities of attackers. The log was then processed using the Apriori association rule mining technique to generate various rules. The existing rule database of the Snort IDS was updated dynamically. This was different from the previous method of off-line rule base addition.

The following conclusions can be drawn from this dissertation work:

- The proposed signature generation system makes proposed IDS efficient in detecting the attacks at the time of their occurrences even if the system was not previously equipped with rules to detect it.
- Figure 5.4 shows that feature extraction system can be used to study any offline network traffic with slight modifications. It can provide useful information about important features required for entropy calculation to build better IDSs.
- Entropy based module was developed to find anomalies. Figure 5.5 and Figure 5.6 show its validity against port scanning attacks. Hence this system can be used to detect new type of attacks as well.
- Table 5.8 shows that alert reduction and ranking system makes proposed system better than Snort.

6.2 Scope for Future Work

There is obviously significant room for improving the methods that we used for the intrusion detection system. The possible improvements in the future are listed as below:

- We will try to extend anomaly system to find anomalies for diverse type of attacks by incorporating the anomaly based routines which are not totally based upon traffic patterns.
- We will try to optimize the rules generated by ASSG system.
- We will find applicability of more data mining algorithms to generate efficient rules.
- For feature extraction system, we will use it on different network traffic patterns to find out features for more attacks.
- We will take into consideration more features in anomaly detection like in degree and out degree of hosts for better performance.

REFERENCES

- [1] Internet System Consortium, "ISC Domain Survey: Number of Internet Hosts." [Last accessed: May, 2011]. [Online] Available: <http://ftp.isc.org/www/survey/reports/2010/04/>.
- [2] Internet World Stats, "Internet User Statistics – The Internet Big Picture: World Internet Users and Population Stats." [Last accessed: May, 2011]. [Online] Available: <http://www.internetworldstats.com/stats.htm>.
- [3] R. Miller, "Twitter is Latest Victim in Series of Attacks." [Last accessed: May, 2011]. [Online] Available: <http://www.datacenterknowledge.com/archives/2009/08/06/twitter-is-latest-victim-in-series-of-attacks/>.
- [4] Denning, D.E.; , "An Intrusion-Detection Model," *IEEE Transactions on Software Engineering*, vol.SE-13, no.2, pp. 222- 232, Feb. 1987.
- [5] Min Yang; Da-peng Chen; Xiao-Song Zhang; , "Anomaly detection based on contiguous expert voting algorithm," *In Proceedings of International Conference on Apperceiving Computing and Intelligence Analysis, ICACIA '09.*, pp.158-161, 23-25 Oct. 2009.
- [6] E. Eskin, A. Arnold, M. Prerau, L. Portnoy, and S. Stolfo, "A Geometric Framework for Unsupervised Anomaly Detection: Detecting Intrusions in Unlabeled Data," *Published in Applications of data mining in computer security.*, Kluwer , Boston, vol. 6, pp. 77-101, 2002.
- [7] Shuai Liu; Zhang, D.Y.; Xiao Chu; Otrok, H.; Bhattacharya, P.; , "A Game Theoretic Approach to Optimize the Performance of Host-Based IDS," . *In Proceedings of IEEE International Conference on Wireless and Mobile Computing Networking and Communications. WIMOB '08*, pp.448-453, 12-14 Oct. 2008.
- [8] Lunt, T.F.; Jagannathan, R.; , "A prototype real-time intrusion-detection expert system," *In the Proceedings of IEEE Symposium on Security and Privacy*, pp.59-

66, 18-21 Apr 1988.

- [9] Garuba, M.; Chunmei Liu; Fraitcs, D.; , "Intrusion Techniques: Comparative Study of Network Intrusion Detection Systems," *In Proceedings of Fifth International Conference on Information Technology: New Generations, ITNG '08.*, pp.592-598, 7-9 April 2008.
- [10] Pfleeger, C. F. and Pfleeger, S. L., "Security in computing (3rd ed.)." Upper Saddle River, NJ: Pearson Education, 2003.
- [11] Wenke Lee; Stolfo, S.J.; Mok, K.W.; , "A data mining framework for building intrusion detection models," *In Proceedings of the IEEE Symposium on Security and Privacy*, pp.120-132, 1999.
- [12] L. Wenke and J. S. Salvatore, "Data mining approaches for intrusion detection," *In Proceedings of the 7th conference on USENIX Security Symposium*, San Antonio, Texas vol. 7, pp. 6-6, 1998.
- [13] D. Barbara, J. Couto, S. Jajodia, L. Popyack, and N. Wu, "ADAM: Detecting intrusions by data mining," *in Proceedings of the IEEE Workshop on Information Assurance and Security*, pp. 11-16, 2001.
- [14] Abraham T., "IDDM: Intrusion Detection Using Data Mining Techniques", *Technical report DSTO-GD-0286, DSTO Electronics and Surveillance Research Laboratory*, 2001.
- [15] Valde A. and Skinner K., "Adaptive, model based monitoring for cyber attack detection", *In Proceedings of Recent advances on Intrusion Detection*, France, Springer Verlag, pp 80-92, 2000.
- [16] Portnoy L., Eskin E., Stolfo S.J., "Intrusion Detection with unlabelled data using clustering", *In Proceedings of ACM CSS Workshop on Data Mining Applied to Security*, pp. 5-8, 2001.
- [17] Eeto L., Eilertscn E., Lazarevic A., Tan P., Dokes P., Kumar V., Srivastava J., "Detection of Novel Attacks using Data Mining", *In Proceedings of IEEE*

- Workshop on Data Mining and Computer Security*, pp. 53-58, 2003.
- [18] Wang Yunwu; , "Using Fuzzy Expert System Based on Genetic Algorithms for Intrusion Detection System," . *International Forum on Information Technology and Applications*, . *IFITA '09*, vol.2, pp.221-224, 15-17 May 2009.
- [19] G. Nychis, V. Sekar, D.G.Anderson, H.Kim, H. Zhang, "An empirical evaluation of entropy-based traffic anomaly detection", *In Proceedings of the 8th ACM SIGCOMM conference on Internet measurement*, New York, USA, pp. 151-156, 2008.
- [20] M. Celnek, T. Conley, J. Willis and J. Graham , "Predictive Network Anomaly Detection and Visualization", *IEEE Transactions on Information Forensics and Security*, vol. 5, No.2, , pp. 288-299, June 2010.
- [21] Rafeeq Ur Rehamn, "Intrusion Detection Systems with Snort, Advanced Intrusion Techniques using Snort, PHP, MySQL, Apache and ACID", Pearson Education ISBN 0-13-140733-3, 2003.
- [22] Snort Manual. [Online] Available <http://www.snort.org/docs>.
- [23] F. Pouget and M. Dacier, "Honey-pot-based forensics", *In AusCERT Asia Pacific Information technology Security Conference (AusCERT2004)*, Brisbane, Australia, pp. 320-325, May 2004.
- [24] C. Kreibich and J. Crowcroft, "Honeycomb: creating intrusion detection signatures using honeypots," *ACM SIGCOMM Computer Communication Review*, vol. 34, pp. 51-56, 2004.
- [25] C. Kreibich and J. Crowcroft, "Honeycomb—creating intrusion detection signatures using honeypots", *In 2nd Workshop on Hot Topics in Networks (Hotnets-II)*, Cambridge, Massachusetts, pp. 451-455, November 2003.
- [26] K. Hyang-Ah and K. Brad, "Autograph: toward automated, distributed worm signature detection," *in Proceedings of the 13th conference on USENIX Security Symposium*, CA, USENIX Association San Diego, vol. 13 , pp. 19-19, 2004.

- [27] S. Singh, C. Estan, G. Varghese, and S. Savage, "Automated worm fingerprinting", *In Proceedings of 6th Symposium on Operating Systems Design and Implementation OSDI '04*, pp. 4-4, December 2004.
- [28] Y. Vinod, T. G. Jonathon, B. Paul, and J. Somesh, "An architecture for generating semantics-aware signatures," *in Proceedings of the 14th conference on USENIX Security Symposium*, USENIX Association, Baltimore, MD, vol. 14, pp. 7-7, 2005.
- [29] Yun Yang; Jia Mi, "Design and implementation of distributed intrusion detection system based on honeypot," *In Proceedings of 2nd International Conference on Computer Engineering and Technology ICCET '10*, vol.6, no., pp.260-263, 16-18 April 2010.
- [30] T. Zang; X. Yun; Y. Zhang; , "A Survey of Alert Fusion Techniques for Security Incident," *In Proceedings of the Ninth International Conference on , WebAge Information Management, WAIM '08*. pp.475-481, 20-22 July 2008.
- [31] Lance Spitzner, "Honeypots, Definition and value of honeypots", [Last accessed: May, 2011]. [Online] Available: <http://www.tracking-hackers.com/papers/honeypots.html>

LIST OF PUBLICATIONS

- [1] Abhay Nath Singh, **Shiv Kumar** and R. C. Joshi. "Intrusion Detection System based on Real Time Rule Accession and Honeypot", *International Conference on Computer Network Security and Applications*, 15-17 July 2011. LNCS Springer. (Accepted)
- [2] **Shiv Kumar** and R. C. Joshi. "Design and implementation of IDS using Snort, Entropy and Alert ranking system" *IEEE International Conference on Signal, Communication and Network*, Kanyakumari, 21-22 July 2011. (Accepted)