VISUALIZATION OF SECURITY VULNERABILITIES THROUGH INTRUSION DETECTION SYSTEM

Ibrar Ahmed
Student ID: 03201079

Farhana Faruqe
Student ID: 03101094

1.1.1 Department of Computer Science and Engineering

1.2 September 2007

DECLARATION

We, Ibrar Ahmed (ID: 03201079) and Farhana Faruqe (ID: 03101094) have completed the thesis titled *Visualization of Security Vulnerabilities through Intrusion Detection System*, under the course, CSE400, regarding the partial fulfillment of our undergraduate degree of Bachelor in Computer Science and Engineering.

We, therefore, declare that this work has been published previously neither in whole nor in part in any thesis work or any conference or journals. We also mentioned work found by other researcher in the reference.

……..

…..…… …..……

Supervisor Authors

ACKNOWLEDGEMENT

We are grateful to Allah for giving us the strength and energy to start this project and finally finish it successfully.

We are really very grateful and take the honor to express our special thanks to our supervisor Risat Mahmud Pathan, M. Sc. for all sorts of supportive suggestions and opinions. Without his support, co-operation and resources the completion of our research in this due time would not have been possible. He is a very industrious person and has tried his best to help us complete our thesis work.

We would also like to thank the senior brothers of university and friends who helped us in every possible way. Especially, we are very grateful to Mr. Tapan Biswas, an alumni and present Lab Technical Officer for his advice and support.

Lastly, we give our special thanks to our department for giving us the opportunity and honor to undertake this thesis, a partial fulfillment of the requirement for the Degree of Bachelor of Science in Computer Science and Engineering and Bachelor of Science in Computer Science.

BRAC UNIVERSITY **ABSTRACT OF BEACHOLOR'S THESIS**

Author: Farhana Faruqe and Ibrar Ahmed

Title: Visualization of Security Vulnerabilities through Intrusion Detection System

Date: September 5, 2007

Department: Department of Computer Science and Engineering

Supervisor: Risat Mahmud Pathan

Security in computer and computer network is of great importance now-a-days. Identifying attacks and taking appropriate measure by system administrator is of special concern. This paper is a study and proposal of an Intrusion Detection System (IDS) for a hypothetical computer network that provides security to Transport and Network Layer attacks in a computer network protocol stack. The proposed system uses visualization (Graphic User Interface) to notify a System Security Officer (SSO) of possible threats and help him/her to take appropriate action to mitigate the effect of attack or to protect the attack before harm is being done. A detailed design of the network IDS has been proposed and criteria for evaluating an IDS is demonstrated.

Keywords: IDS, TCP/IP, Intrusion, Distributed system, Firewall, Visualization

Keywords: IDS, TCP/IP, Intrusion, Distributed system, Firewall, Visualization

Language: English

Table of Contents

2 INTRODUCTION

In our daily lives, security is an issue of huge concern. We are worried about security when we cross roads, when we exchange words on the telephone, when we send our children to school, when we do monetary transactions and so on. We are always worried about exposure to threats that trespass into our privacies, obtain illegal accesses to resources and misuse illegally accessed resources. The motive behind these threats varies from financial gains, political gains, a feeling of power and importance or simply inquisitiveness. Therefore, we strive everyday to protect our near and dear ones, our businesses and organizations, our resources and ourselves from exposure to such threats.

The need for preventive measures varies from situation to situation and organization to organization. We have to decide what kind of threats we are vulnerable to and what preventive measures to take.

We use the term “security” in many ways in our daily lives. A “Financial security” involves a set of investments that are adequately funded; we hope that the investments will grow invaluably over time so that we have money to survive later in life. And we speak of a child’s “physical security”, hoping he or she is safe from any potential harm. Just as each of these terms has a very specific meaning in the context of its use, so too dose the phrase “computer security.”

Security is a very difficult topic. Everyone has a different idea of what ``security'' is, and what levels of risk are acceptable. The key for building a secured network is to *define what security means to an organization*. Once that has been defined, everything that goes on with the network can be evaluated with respect to that policy. Projects and systems can then be broken down into their components, and it becomes much simpler to decide whether what is proposed will conflict with your security policies and practices.

When we talk about “computer security,” we mean that we are addressing three very important aspects of any computer-related system: confidentiality, integrity and availability.

Confidentiality ensures that computer-related assets are accessed only by authorized parties. That is, only those who should have access to something will actually get the access. By “access” we mean not only reading but also viewing, printing, or simply knowing that a particular asset exists. Confidentiality is sometimes called secrecy or privacy.

Integrity means that assets can be modified only by authorized parties or only in authorized ways. In this context, modification includes writing, editing, changing status, deleting and creating.

Availability means that assets are accessible to authorized parties at appropriate times. In other words, if some person or system has legitimate access to a particular set of objects, that access should not be prevented. For this reason, availability is sometimes known by its opposite, denial of service.

As the world becomes more connected by networks, the significance of network security will certainly continue to grow. Network security consists of the provisions made in an underlying computer network infrastructure, policies adopted by the network administrator to protect the network and the network-accessible resources from unauthorized access and the effectiveness (or lack) of these measures combined together.

Network security starts from authenticating any user. Once authenticated, firewall enforces access policies such as what services are allowed to be accessed by the network users. Though effective to prevent unauthorized access, this component fails to check potentially harmful contents such as computer worms being transmitted over the network.

An intrusion detection system (IDS) is a system of software and hardware that ensure the security of a system by identifying malicious or suspicious events. It raises an alarm when such a behavior is experienced. Based on the alarm and response by a System Security Officer (SSO), changes are made to the IDS to accommodate further newer threats. To raise an alarm, the IDS analyzes the access made to a system and classifies either as a safe access or an intrusion.

An IDS can be classified as either stand-alone (or strictly-centralized) or distributed IDS. Stand-alone IDS can either be host-based (residing on a single host) or networked-based (obtaining data from the network traffic). A distributed IDS collects data from various points in a network and sends it to a central host.

3 OBJECTIVE: FOCUS OF RESEARCH

In our quest to the proposition of an IDS that will notify the concerned of possible threats and advice to safeguard against then, we have focused our attention on certain issues. We had come up with a few questions the answers to which have led us to the understanding of threats, their effects, and detection of and protection against such threats. Through the answers to these questions, we have been able to direct our research towards our goal of proposition of visual IDS.

3.1 What Are The Possible Attacks And Their Effects On Users?

Users are vulnerable to ever increasing threats. The variety of attacks ranges from minor ones to severe ones based on the motives of attackers. The most common types of attacks are those where a user is unable to use his system as the system becomes too slow or it cannot fulfill request of the users. This type of attacks kills invaluable time of users.

Another very common but much more severe case is one where the attacker gains supreme control over a user's system. In such cases, a user may find himself a stranger to his own system. He may also send valuable information to the attacker, breaching his privacy. A similar effect is instigated by spying attacks where an attacker silently monitors each and every step of the user. Such an attacker may intercept a user's message, use it for his own gain or modify the message.

There are attacks in which cases, a user loses data unknowingly. In certain other threats, a user exchanges data with the attacker posing as a trusted user to share data and information with. In such cases, users are left unprotected against many kinds of vulnerabilities.

A user may also send and receive data without his consent as a result of spamming. Spamming is a very irritating intrusion and it can also send personal information to organizations or individuals without the consent of the user.

Last, but not the least, a user may succumb to intrusion through social engineering. Social engineering is obtaining a user's information such as password, secret questions, credit card numbers, etc through trickery. A user may reveal such information to an intruder when the intruder puts the user in a state of confusion or disguises himself or herself as an administrator or some other official urgently needing the user's information. This information is then used by the imposter to hack into accounts, steal money off credit cards, and cause numerous other monetary and/or social hazards.

CSI/FBI reports that the number of incidents of threats has not changed much in 2006 from the previous year [13]. The report reveals that a huge number of organizations and individuals are not sure whether they are vulnerable to threats or not. This number accounts to 28% of all who have been surveyed (Table 1). This is an issue of huge concern since they may not be aware of what loss they have incurred through such threats. Moreover, about 48 % of all surveyed organizations and individuals have detected between one to five incidents. This number has increased by 14 % from 1999, an indication beckoning urgent need of better security measures such as improved firewalls, anti-viruses, IDS, Fraud Detection System (FDS, a subset of IDS) and so on.

The report also concludes that financial loss amounted to a little less than sixteen million US dollars due to virus contamination. Figure 1 represents the findings of the report which also states that other major losses are afflicted by threats such as unauthorized access to information, theft of proprietary information, denial of service (to be explained in Section 4) and financial fraud.

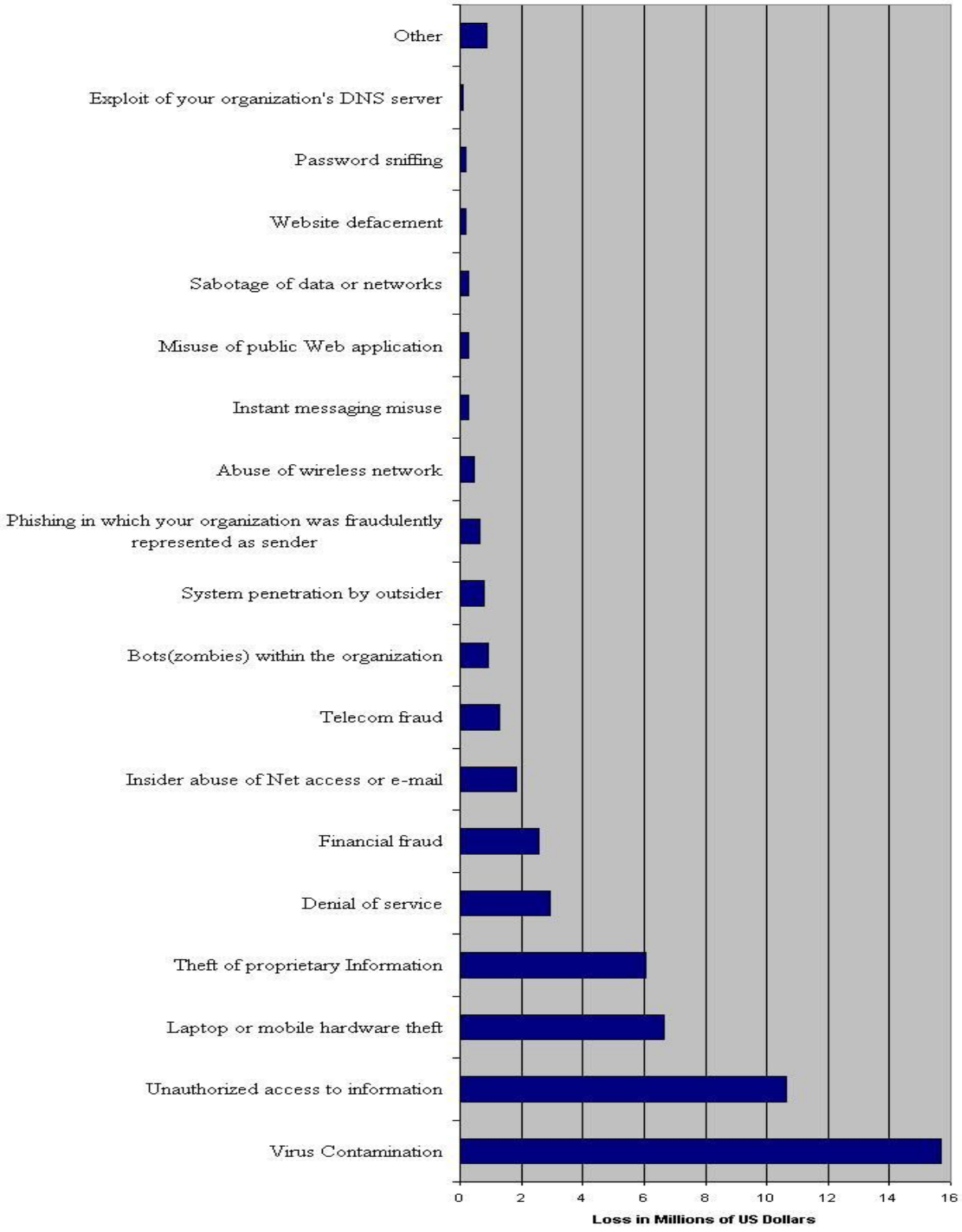


Fig 1: Financial Loss in Millions of US Dollars

Table 1: Number of Incidents

How many incidents? (by % of respondents)	1–5	6–10	>10	Don't know
2006	48	15	9	28
2005	43	19	9	28
2004	47	20	12	22
2003	38	20	16	26
2002	42	20	15	23
2001	33	24	11	31
2000	33	23	13	31
1999	34	22	14	29

From the above statistics, it is quite clear that the threats are increasing in number and their effects of various threats can cause all kinds of damage including financial. Therefore, it is very crucial to protect users from various threats.

3.2 How to Protect?

Before we can discuss ways of protecting users from intrusions, threats that we are concerned with for the purpose of this research, it is necessary to define intrusion itself. There are various ways to express the meaning of intrusion, but we take the definition of "wikipedia", as part of their discussion of IDS, as a formal definition for our purpose. In computer science, intrusions are attacks against vulnerable services in a distributed (networked) system, data driven attacks on applications, host based attacks such as privilege escalation, unauthorized logins and access to sensitive files, and malware (viruses, trojan horses, and worms) [12]. Keeping the definition in mind, we venture on various ways to protect users against such attacks.

Users can be protected in variety of ways. Usually, more than one method is combined to safeguard users from intruders/attackers. The methods vary, just as attacks do, in a number of ways depending on the types of attacks.

The most effective method is a preventive measure where a user is protected from various threats. The user must be educated about threats and about safety measures against them. Users are enlightened with knowledge of

common tricks attackers play during social engineering. They must be taught the importance of secrecy regarding their passwords, credit card numbers, and other personal information. These methods would help them identify possible social engineering techniques that may prove socially and/or monetarily hazardous if ignored.

Additionally, various protective systems can be used that detect threats and either take decisions on their own or inform users or an administrator or an SSO about them. Such systems make use of software and/or hardware to uphold users against intrusions. One of the most common systems used nowadays is a firewall. This system acts as a barrier between a user's system and the rest of the network. The firewall allows only trusted data to reach a user.

In most cases, a firewall only cannot prove sufficient in securing a system against all kinds of threats. Layers of defense mechanisms are used to protect a system from possible hazardous and seemingly less or non hazardous intrusions. Various organizations and individuals adopt intrusion detections systems (IDSs) to team up with firewalls to safeguard computers on a network. In Table 2, some of the basic differences between IDS and firewall have been stated.

Table 2: IDS Vs Firewall

IDS	Firewall
Warn against suspicious traffic	Drop proven attack packets
Logs packets	Logs packets
Examines whole stream of packets	Examines a single packet
Reassembles and normalizes application messages	Does not do so
Deep packet inspection	Does not do so
Generates alarms when attack packets identified	Does not generate alarm while dropping packet
Less precise (Alarm set off on mere suspicion)	Precise (Packets dropped only when sure)

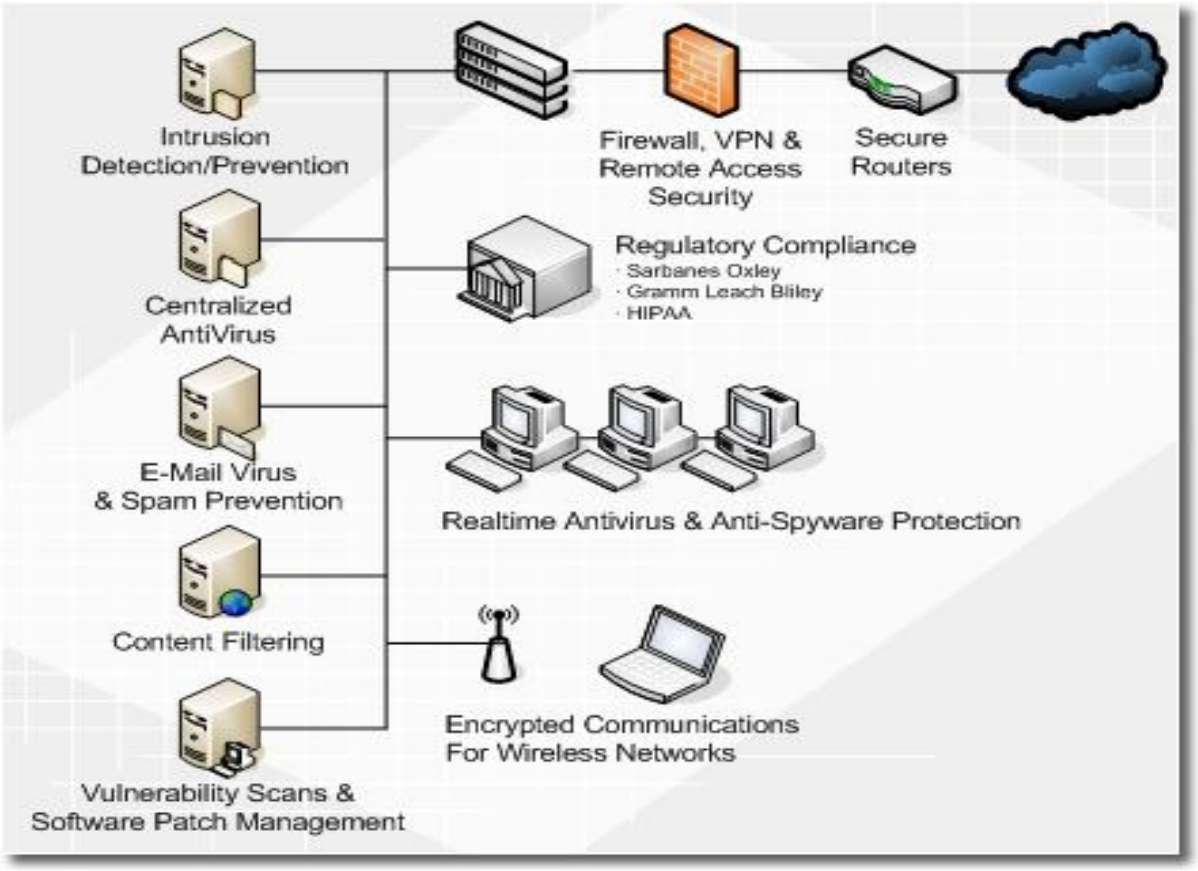


Fig 2: Distribution of Security Tools over a Network

Moreover, users have to update themselves with information about new severe threats and their methodologies with protection mechanisms against these novel attacks. Failing to do so expose the user's system to threats whose effects may not be known until it is too late.

3.3 How to detect a possible intrusion?

To keep a system free from intrusions, the intrusions have to be detected as early as possible. At the same time, it should also be ensured that valid and harmless accesses to the system are not prevented or detected as intrusions causing. Accordingly, various detection techniques can be used in order to keep a system intrusion free without hampering its day-to-day activities. Similarly, various tools ranging from anti-virus programs to firewalls and IDS can be used to keep a system risk-free.

Intrusion detection can be achieved by misuse detection or anomaly detection or a hybrid technique which is a mixture of both the mentioned techniques [7].

Whatever the technique used in intrusion detection, the first activity that has to be carried out is the collection of data. Data are collected from various points in a network. Sensors are used to collect data from points that lead to nodes that are more likely to be attacked [1, 7].

To provide security at its best, layers of security steps should be taken providing defense in depth. A network's doorkeeper is a border firewall that prevents only those network traffic from an outside source (usually the internet) which proven to be malicious. This is called ingress filtering [10]. Screening router firewalls can also be used when traffic is routed from the outside world to the inside. This router acts as the border firewall for a subnet of a bigger network, but generally, a screening router firewall can be followed by the main firewall. After this, an internal firewall can be used. A third layer of firewalls can be used to protect individual hosts. This is called host firewalls and an example could be ZoneAlarm. Host firewalls can be client host firewalls or server host firewalls. Various servers, such as proxy server, file server, web server, and so on, can be grouped together. This group is called "demilitarized zone" [10]. This makes it easer to protect the servers by using sensors at the incoming links to be logged for analysis by an IDS. Thus, IDS can be used with all these layers of firewalls to engulf the system with yet another layer of protection. The choice of IDS can also vary. A system can have host-based IDS. This means that the responsibility of protection via detection lie at the hands of the individual hosts that may be attacked. On the contrary, network-based IDS can be used where the protection responsibilities do not lie solely on the hosts. Here, the whole networked is attempted to be kept secured.

In addition to firewalls and IDS, anti-virus programs must be installed on individual hosts, both clients and servers, to ensure protection against viruses and Trojans. This protection is necessary as transfer of files by physical means such as tapes, optical discs and mobile drives can spread such threats. Additionally, physical guard is also necessary to ensure that sabotages of links do not take place as this may render the whole system vulnerable if the

sabotaged link disconnects one or more of the protective layers already mentioned above.

Protection of systems from threats and vulnerabilities can take place using a combination of layers mentioned above. What combination should be used depends on an organization or an individuals priorities and the network architecture used. An ideal security system could be using border firewalls, network-based IDS and host firewalls and anti-virus programs. Using more layers is a decision of the organization based on its priority between speed and security because it is almost certain that increasing the number of layers of protection keeps the possibility of fall in data transfer speed quite open. This is a trade-off most organizations will gladly accept considering the severity of effects successful threats can have.

The collected data is then used to analyze so as to be classified either as intrusion or safe access. The analysis is carried out by various means depending on the technique used. Most commonly, the collected data is used to check if it satisfies with data obtained from previous safe accesses to the system and with rules for other safe accesses. Signature and pattern matching can also be used to check if the data collected indicates safe access. If the check fails, the access is classified as intrusion.

3.4 How to report intrusion?

Once an intrusion has been detected, a report has to be generated. This report is used to alert an SSO who can take appropriate decisions to confirm safety of the system. The method of reporting to an SSO can also vary. An intrusion can be reported actively or passively. An active report would notify the SSO immediately by invoking an interrupt or alarm while a passive report may involve storing of the intrusion related data in a log file so that the SA can look at it whenever it suits him.

Whether it be active or passive, reporting to the SA about the intrusion is very crucial and it should be formatted in a way that will enable him to make appropriate decisions. The format of the report can be either textual or visual. In a textual report, the data are arranged in a table while in a visual report, it can be done using charts and figures. A visual report can be more productive since it

can easily pinpoint vulnerabilities while a textual report can cause crucial data to be missed or lost if the volume of data is too large. In Figure 3, an output for RST attack (discussed in Section 4) has been shown. As can be seen, numerous lines of packet information has been shown before the actual output is printed. This may cause many to overlook most crucial data/output and not take proper action or decision. On the contrary, Figure 4 and Figure 5 represent a visual output and it precisely highlights the vividness a graphical output presents to an SSO. To visualize a graph that indicates the output of analysis of network traffic, visualization tool Tulip (Figure 4) can be chosen [16]. To perform the actual detection, the lowest scoring accesses is visualized using this three dimensional general graph. Figure 5 shows one of the views displayed in the tool called Advanced Analytics. This is a tool use in paper [17] within the authors' organization to perform visualizations with alert data.

2007081111:17:15:046 192.168.0.01 192.168.0.51 1089 80 4 20 60 ICMP 0 000010 8
2007081111:17:15:046 192.168.0.01 192.168.0.51 1089 80 4 20 60 ICMP 0 000010 8
2007081111:17:15:046 192.168.0.01 192.168.0.51 1089 80 4 20 60 ICMP 0 000010 8
2007081111:17:15:046 192.168.0.01 192.168.0.51 1089 80 4 20 60 ICMP 0 000010 8
2007081111:17:15:046 192.168.0.05 192.168.0.51 1089 80 4 20 60 ICMP 0 000010 8
2007081111:17:15:046 192.168.0.41 192.168.0.51 1089 80 4 20 60 ICMP 0 000010 8
2007081111:17:15:046 192.168.0.81 192.168.0.51 1089 80 4 20 60 TCP 0 010000 -
2007081111:17:15:046 192.168.0.91 192.168.0.51 1089 80 4 20 60 TCP 0 000001 -
2007081111:17:15:046 192.168.0.91 192.168.0.51 1089 80 4 20 60 TCP 0 010000 -
2007081111:17:15:046 192.168.0.41 192.168.0.51 1089 80 4 20 60 ICMP 0 000010 8
2007081111:17:59:046 192.168.0.41 192.168.0.51 1089 80 4 20 60 TCP 0 000100 -
2007081111:17:59:046 192.168.0.41 192.168.0.51 1089 80 4 20 60 ICMP 0 000010 8
2007081111:17:59:046 192.168.0.41 192.168.0.51 1089 80 4 20 60 ICMP 0 000010 8
2007081111:17:59:046 192.168.0.41 192.168.0.51 1089 80 4 20 60 ICMP 0 000010 8
2007081111:17:59:046 192.168.0.41 192.168.0.51 1089 80 4 20 60 TCP 0 000010 -
2007081111:17:59:046 192.168.0.41 192.168.0.51 1089 80 4 20 60 ICMP 0 000010 8
2007081111:17:59:046 192.168.0.41 192.168.0.51 1089 80 4 20 60 ICMP 0 000010 8
2007081111:17:59:046 192.168.0.41 192.168.0.51 1089 80 4 20 60 ICMP 0 000010 8
2007081111:17:59:046 192.168.0.41 192.168.0.51 1089 80 4 20 60 ICMP 0 000010 8
2007081111:17:59:046 192.168.0.41 192.168.0.51 1089 80 4 20 60 ICMP 0 000010 8
2007081111:17:59:046 192.168.0.50 192.168.0.51 1089 80 4 20 60 ICMP 0 000010 8
2007081111:17:59:046 192.168.0.49 192.168.0.51 1089 80 4 20 60 TCP 0 000010 -
2007081111:17:59:046 192.168.0.16 192.168.0.51 1089 80 4 20 60 ICMP 0 000010 8
2007081111:17:59:046 192.168.0.41 192.168.0.51 1089 80 4 20 60 TCP 0 000010 -
2007081111:17:00:046 192.168.0.33 192.168.0.51 1089 80 4 20 60 ICMP 0 000010 8
2007081111:17:00:046 192.168.0.41 192.168.0.51 1089 80 4 20 60 ICMP 0 000010 8
2007081111:17:00:046 192.168.0.41 192.168.0.51 1089 80 4 20 60 ICMP 0 000010 8
2007081111:17:00:046 192.168.0.41 192.168.0.51 1089 80 4 20 60 ICMP 0 000010 8
2007081111:17:00:046 192.168.0.41 192.168.0.51 1080 80 4 20 60 ICMP 0 000010 8
2007081111:17:00:046 192.168.0.41 192.168.0.51 1089 80 4 20 60 ICMP 0 000010 8
2007081111:17:00:046 192.168.0.41 192.168.0.51 1099 80 4 20 60 ICMP 0 000010 8

REPORT

Attack Identified: PING Flood (DoS)

At

Destination IP:
192.168.0.51

Destination Port:
80

Type: *Internal attacker*

From

Source IP (port):
192.168.0.01 (1089)
192.168.0.05 (1090),
192.168.0.41 (1099),
192.168.0.50(1089) ,
192.168.0.16 (1099)

Advice

1. Add rule to ban the above IPs.
2. Send Email notification to user of destination IP for attack notification and to attacker for warning.

Fig 3: Textual Output

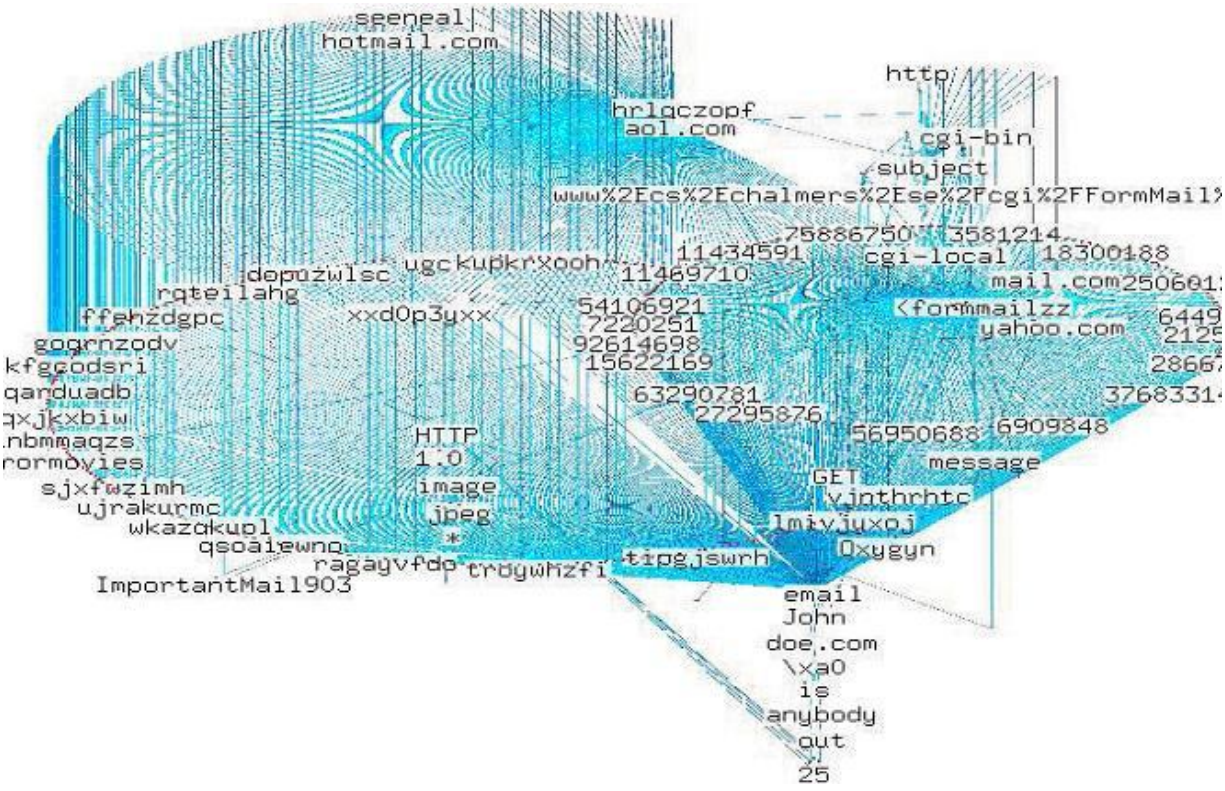


Fig 4: Graphical Output 1 (Spam Attack) by Tulip

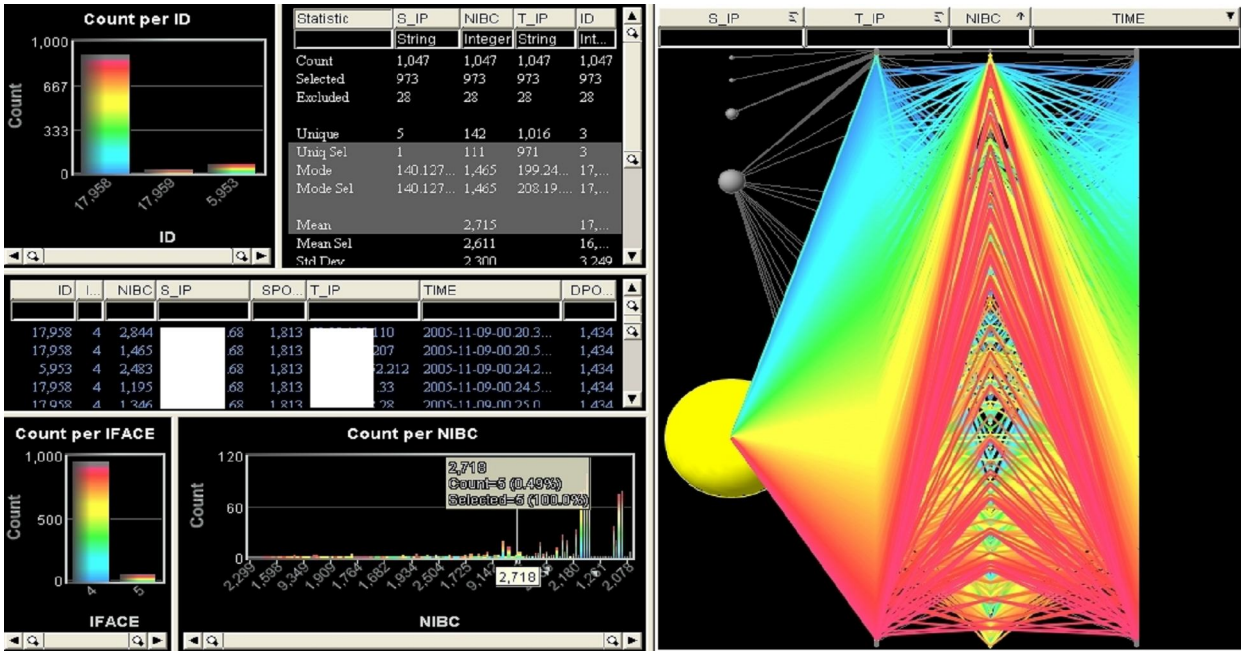


Fig 5: Graphical Output 2 by Advanced Analytics

4 LITERATURE REVIEW (PREVIOUS WORK)

Axelsson presents a research of protecting valuable computer resources through intrusion detection that will most likely take on an increasing role in protecting computer systems over the next few years [1]. The computer security field is primarily concerned with protecting one particular resource: valuable data, and ultimately valuable information. In this computer security section briefly describe about CIA (confidentiality, integrity, availability) of computer security. We have tried to keep these criteria in designing our proposed system. For example, ping attack; the objective of ping attack is to make the resources unavailable to the client; and in our system we have tried to detect this attack and set the alarm to notify the security officer. Additionally, we have not extracted any data/information from one or multiple packets that will give the SSO access to confidential data. Stefan also lists six general, non-exclusive approaches to anti-intrusion techniques: preemption, prevention, deterrence, detection deflection, and countermeasures and describe them elaborately with a figure, which is very useful. In our thesis we have borrowed this section and analyzed them to look for areas where we could work and use in our research. Details of this have been discussed in Section 5.

Furthermore, intrusion detection systems, as explained in the paper, are designed to detect computer security violations made by important types of attackers such as attackers using pre-packed ‘exploit scripts’ (Primarily outsiders), attackers operating under the identity of a legitimate user, for example by having stolen that user’s authentication information (password) (Outsiders and insiders), insiders abusing legitimate privileges, etc. The author defines intrusion, intrusion detection, malicious behavior, security policy, external agent, automated detection and alarm, delivered to the proper authority, intrusion has taken place. He groups intrusion detection systems into two overall classes: those that detect anomalies, hereafter termed anomaly detection systems, and those that detect

the signatures of known attacks, hereafter termed signature based systems. Our proposed IDS can be classified as an anomaly detection system. In the paper, a generic architectural model of an intrusion detection system describes the figure of organization of a generalized IDS and its each and every part, such as: audit collection, audit storage, processing, configuration data, reference data, active/processing data and alarm. In designing our classes, we have tried to include those units in our classes and packages.

Antilla discusses the background of the IDS [2]. The author states that the number of companies that have intrusion detection systems have increased from 42 to 73 percent between years 1999 and 2003, according to the annual computer crime and security survey released by the Computer Security Institute (CSI) and FBI. On that same period, the number of companies that have firewalls in place has increased from 91 to 98 percent. The paper categorizes IDS and describes them elaborately. According to Gartner Group, 75 % of all attacks on the Web occur at the application level. These statistics highlight the increased need of security and this serves as the motivation behind our research.

The paper describes top ten attack classes (Application Buffer Overflow, Backdoors and Debug Options, Cookie Poisoning, Cross Site Scripting, Forceful Browsing, Hidden Field Manipulation, Known Vulnerabilities, Parameter Tampering, Stealth Commanding, Third Party Misconfiguration) of application layer with brief explanations how they could be executed. It does not describe any attacks of other layers. The statistics provided have motivated us to direct our thesis towards network-based IDS since most companies and organizations which need extensive security would be most benefited from network-based IDS. Also, after going through a discussion of the presentation layer attacks, we have decided to skip application layer attacks to design our basic IDS. We have come to the conclusion that to design network-based architecture, it would be best if we started working at Network and TCP layers first and then later, incorporate other attacks from other layers.

There is a discussion in the paper about product survey. The topic is related to e-business environment. Internet Security Systems (ISS), Cisco Systems, Symantec, Enterasys, NFR, Entercept / Network Associates, Intrusion, Snort and other products have been described and then a list of all evaluated

products is presented. The survey has given us an idea whether it is still fruitful to work on IDS even at the presence of such softwares.

The most important part of paper [2] is the evaluation and analysis part. The evaluation is divided into three phases where the product list is cut smaller after every phase until there is the final solution to be implemented on the reference system. There are certain criteria based on the reference system, which limit the possible products. The reference system is a system whose purpose is to provide web-based services for the customers who are accessing the system from the public Internet. Presentation and business logic are separated from the customer data. A number of rounds of checking are discussed in the paper to evaluate whether they meet the above mentioned criteria.

Paper [2] is a very useful one for e-business environment, and from it, we have come with the criteria to keep in mind while designing the system. From this review, we have analyzed the feasibility of implementation of IDS that should possess certain criteria to satisfy the security measures of the referenced system and to be feasible economically. The design of a system is only logical if the implementation is feasible economically and competitively.

In order to make our system economically feasible and competitive, our proposed system is extensible as it has a component-based design. This also makes it possible to lower maintenance cost and since no specialized hardware is used, the installation of the system is also expected to be less expensive. Further advantages of the system have been discussed in Section 8.

In their papers, Hedbom et al say that security extensions are usually put into operation on a perceived notion of benefit without any consideration of the risks [3]. It presents an overview of the functionality of three different security extensions, i.e., anti-malware softwares, firewalls, and intrusion detection systems (IDS). Anti-malware tool acts as an internal defense mechanism. Roughly, three different defense strategies may be used: activity monitoring, scanning, and integrity checking in this tool. Firewalls are typically of two different kinds: packet filters and proxies. In section 3, they discuss the possible vulnerable points of these three mechanisms. When an anti-malware tool is installed on a computer system, there is always a risk that the system owner and its users believe that they are more or less immune to virus attacks. This is a

false sense of security because there are some risks and dangers such as early activation, Unknown Viruses and Signature Files, and Dynamic Files. In case of firewalls, they discuss about some of the important issues of risks: Configuration Files, The Negative Side of Chokepoints, One-Way Protection, Problems with Proxies. Then they describe the IDS with some important points: detection policy, log files, the problem of alteration and distribution. The paper also attempts towards a framework for classification of the risks that they believe are added to the system. They have divided the risks into two categories: system risks and privacy risks. These categories are each further divided into three sub-categories: high, medium, and low. The research further says that security extensions cannot be added to an insecure system or used to patch shortcomings in the underlying system if they are dependent on that system for their own security. We think this is really a good point. We can relate this point to our design where currently we have not considered the security of the log and referenced file. This has provided motivation for further research which is beyond our current research. Additionally, the problems of anti-malware tools and of firewalls have helped us in deciding to design network-based IDS and not host based IDS. Also, through this, we have included in our design, a filtering mechanism through a firewall to provide layers of protection.

Hedbom in another paper addresses the self-protection problem and discusses how to avoid the risks and dangers of employing security extensions [4]. His paper gives some definitions of the terminology used in this thesis such as computer security, intrusion, attacker or intruder, object, subject, detection policy, filtering policy, distributed vs interconnected systems, network operating systems, security mechanism and security extension. This section also discusses possible risks and shortcomings in security extensions and elaborates on some of the security issues involved in distributing security extensions. They have discussed about firewalls, IDS, risks of security extensions, distributing security extensions. When the packet comes from the internet we use firewall in our system for partial filtering. Distributed attacks have been discussed and suggested that cooperating security extensions may be used as part of defense against them.

Lindqvis et al illustrates the complexity of the system characteristics that makes intrusions possible, and thereby to shed light on the corresponding intrusion process, which in turn may help to design tools for intrusion detection [5]. Their paper presents the schemes used for description of intrusions. In the analysis of such complex events as computer security intrusions, it is important to determine exactly what dimension (aspect, attribute) of the event the analysis concerns. It gives a straightforward presentation of five intrusions on three different systems like a database system: Ordering of records, UNIX: Keyboard snooping etc. It further presents a refined analysis of the underlying flaws. In most cases, there are two or three types of reasons why intrusion is possible: related to the design of a specific functional (software) module, integration and setting up of the system, and the administration and use of the system. Some of the intrusions presented early in the paper have been re-investigated in view of this decomposition. A part of the paper discusses the outcome of this analysis and the problem of referring an intrusion to a single cause. It is said that tools are available for both the packet authentication problem in Novell, the xterm logging vulnerability and the keyboard snooping flaw in UNIX. Such tools make it possible for even a user with minimal system knowledge to abuse system security. Paper [5] clearly shows that an intrusion is a function of not only one, but a number of vulnerabilities and characteristics of the system and the organization. This makes the problem quite complex, but complexity is a problem not only for developers and integrators but also for users and administrators, as we have seen in this work by Lindqvis et al. It should also be noted that none of the problems that have been presented here are really technically difficult to solve. Solutions exist, but the problem is to spread this knowledge, use it, and use it correctly.

In our research, the complexity issue has been given special attention to. We have add a visual tool in the IDS to monitor the referenced system with very little complexity. The time it takes an SSO to respond to an intrusion matters greatly in maintaining the security of the over network system. The reduced complexity in monitoring the system helps reduce the time it takes to interpret an attack. The discussion of the intrusion process has also helped in our design of the analysis engine that has the responsibility of identifying and classifying intrusions.

Hedbom et al in another of their papers address the security implications and requirements that the IDA (Intrusion Detection Architecture) puts on the IDS (Intrusion Detection System) in various distributed environments [6]. The paper claims that, although they are more accentuated in what we call a fully distributed architecture, these requirements hold for any type of IDS that consists of interconnected cooperating components. The authors also believe that those requirements have in large part been overlooked in today's systems as a consequence of the bias toward detection mechanisms. The paper also highlights some terminology like target, detection policy, nodes, domain, and events. There are a number of different architectures for intrusion detection systems like centralized and distributed architectures and combinations thereof. The paper also discusses the "knowledge" of distributed IDS within these topics. It discusses distributed knowledge of the detection policy, distributed knowledge of security events and audit logs, and confidentiality concerns in distributed IDS. A part of paper [6] basically discusses the knowledge needed by different components in an IDS and the flow of information between components that this knowledge produces in different IDAs. It further presents a number of security requirements for an IDS based on the flow of information and security implications previously discussed in other section. "Information dominance - Toward a stronger notion of security for IDS" this is a very useful topic for our thesis. The meaning of information dominance for IDS according to the author is that, information (knowledge) contained within an intrusion detection entity must be kept private to malicious adversaries (confidentiality requirement). In addition, the information must be protected from unauthorized alteration, fabrication and deletion (integrity requirement) as it may lower the operational advantage of the IDS. They propose that the property of information dominance for IDS should include: Confidentiality of audit data, confidentiality of the detection policy, integrity of audit data, and integrity of the detection policy. These properties are then explained in details. The paper also compares modern IDS' in order to judge how well they adhere to the requirements presented in paper [6]. It analyzes the information flow induced by organizing event collection and detection in various intrusion detection architectures (IDA). The knowledge needed, and the information flow produced by components of an IDA, lead to new security implications.

We have acknowledged the need for privacy and security of various components of the IDS, but for the purpose of this research, we have overlooked these for the time being. We have, however, discussed this briefly in our section “Future Work” and stressed the importance these activities. Moreover, the proposed IDS has been designed keeping in mind that with only a few changes or addition, various components of the IDS can be secured and kept invisible from other users some of which may be prospective threats or outsiders accessing the referenced system through internet or extranet.

Based on these studies, we have identified attacks, created our design and implemented the design. In the next section, we have discussed the network protocol we have considered, the possible attacks at different layers and the attacks we have handled.

5 NETWORK PROTOCOL STACK: POSSIBLE ATTACKS

Network is built as *layers*. The job of each layer is to offer *services* to higher layers, and *hide* from those higher layers the details of how the services are implemented. Protocols of the various layer is called the protocol stack.

The figure below shows a comparison of the Open Systems Interconnection (OSI) model and the TCP/IP protocol suite. The TCP/IP set of protocols maps to a four-layer conceptual model: application, transport, Internet and network interface. This model is referred to as the Internet Protocol Suite or the ARPA model. As shown below, each layer in the darker green Internet Protocol Suite corresponds to one or more layers of the white OSI model.

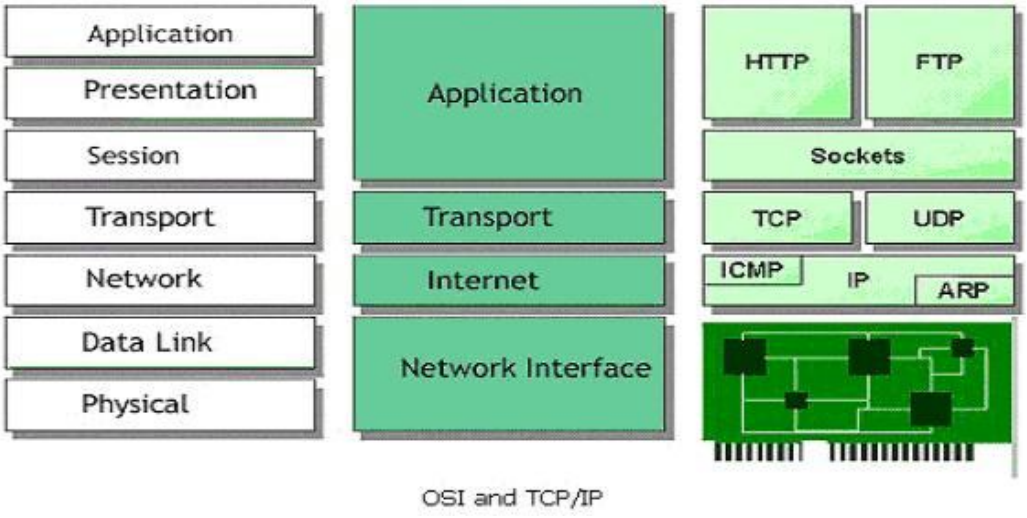


Fig 6: OSI Model and TCP/IP Protocol Stack

The original TCP/IP reference model consists of 4 layers , but is now viewed by many as a **5-**layer model . It is called the **Internet protocol suite** (Table 3) which is the set of communications protocols that implement the protocol stack on which the Internet and most commercial networks run. It has also been referred to as the TCP/IP protocol suite, which is named after two of the most important protocols in it: the Transmission Control Protocol (TCP) and the Internet Protocol (IP), which were also the first two networking protocols

defined. Today's IP networking represents a synthesis of two developments that began in the 1970s, namely LANs (Local Area Networks) and the Internet, both of which have revolutionized computing. But the OSI model describes a fixed, seven-layer stack for networking protocols. Comparisons between the OSI model and TCP/IP can give further insight into the significance of the components of the IP suite.

Table 3: Five Layer TCP/IP Model

Layer	Protocol
Application layer	DHCP, DNS, FTP, Gopher, HTTP, IMAP4, IRC, POP3, SIP, SMTP
Transport layer	TCP, UDP, DCCP, SCTP, RSVP
Network Layer	IP (IPv4, IPv6), IGMP, ICMP, OSPF, ISIS
Data link layer	802.11, ATM, DTM, Token Ring, Ethernet, FDDI, Frame Relay, GPRS
Physical layer	Ethernet physical layer, ISDN, Modems, PLC, SONET/SDH, G.709

In our thesis we follow the 5 layers of TCP/IP model because we think it's simpler and more independent. The abstraction of layer is more pronounced in this model and unnecessary layers are minimized into less making it easier to manage.

5.1 Examples of Attacks

Attack can cause damage to a system at different layers discussed in the sub-section above. Through a survey, we have find out the following attacks that can exploit a system at its different network layers.

Physical Layer: Cable cut, Spectrum, and Jamming/fade.
Data Link Layer: Flooding Attacks, and Topology Engagement Attacks.
Network Layer: DOS (Denial of Service), Spoofing, Smurfing and Sequence Number Guessing (part of spoofing).

Transport Layer: ACK Denial-Of-Service Attack, Sniffing, SYN Attack, RST Attack, FIN Attack, Tear Drop Attack, Session Hijacking Attack, Port Scan.
Application Layer: Vulnerable CGI Programs, Spam (for Email), Nimda worm & Mutations, Malicious URLs, Spyware and Ad ware Attacks, Back door, Trojan horses, FTP Bounce Attack.

5.2 Attacks Handled

We have worked on providing safeguards against intrusions at the software level and have not considered hardware-level protections. We have also safely assumed that a stand-alone PC not connected to a network is not vulnerable to intrusions. Therefore, we have tried to establish methods to safeguard systems on a network, a hypothetical network for our research, from intrusions.

Of all the attacks, we have tried to work on some of those affecting Network Layer and TCP Layer. We have tried to identify some of the most severe attacks at the mentioned layers and have examined how they are constructed and carried out. We have worked on the following attacks: RST Attack and FIN Attack (in TCP Layer), PING (Flood) Attack (in Internet Layer).

5.2.1 Transport layer

SYN Flooding: The basis of the attack is to not complete the 3-way handshake necessary to establish communication. Specifically the attacker (client machine A in figure 6) refusing to send the ACK signal to the host server (B) after receiving the SYN/ACK from Host B. Such a connection is called a half open connection.

Instead of sending an ACK, attacker A sends another SYN signal to the victim server. The server again acknowledges it with a SYN/ACK and B again refuses to send the final ACK signal. By repeating this several times the attacker tries to overflow the data structure of the host server. The data structure is built in the memory of the host server with the purpose of keeping records of connections to be completed (or half open connections). Since the data structure

is of a finite size, it is possible to overflow it by establishing a large number of open connections.

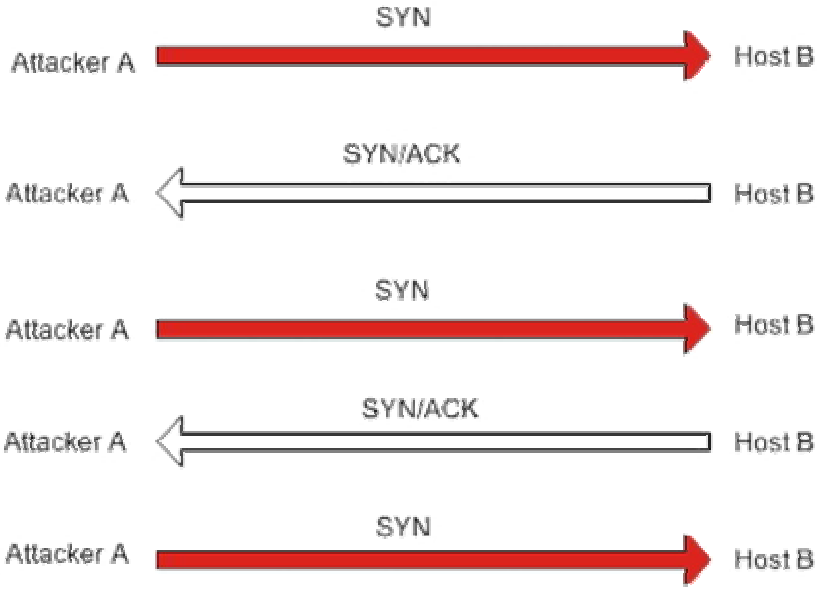


Fig 7: Attacker A flooding Host B with SYN

Once overflow occurs the host server will not be able to accept new connections thus resulting in a denial of service. There is however a time-out associated with each of the connections (approximately 3 minutes) after which the host server will automatically drop the half open connection and can start accepting new connections. If the attacker can request connections at a rate higher than the victim servers ability to expire the pending connections then it is possible to crash the server.

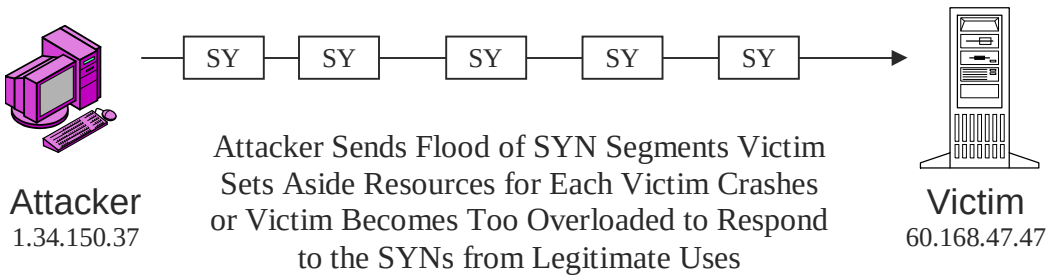


Fig 8: DoS Attack - SYN Flood

Thus the objective of SYN flooding is to disable one side of the TCP connection which will result in one or more of the following:

- The server is unable to accept new connections.
- The server crashes or becomes inoperative.
- Authorization between servers is impaired.

Reset (RST) Attack: Whereas SYN flooding attacks are carried out at the beginning of the connection, RST attacks usually occur in the middle of it. The RST flag in the TCP packet is used to reset the connection. If two machines C and B are in the middle of a connection and an attacker A decides to attack machine C then all he has to do is calculate/guess the correct sequence number using the methods described above. (there is no ACK in a RST packet). After that the attacker can disrupt the connection by sending a spoofed packet with RST flag set to B. The attacker then assumes B's identity and starts attacking C.

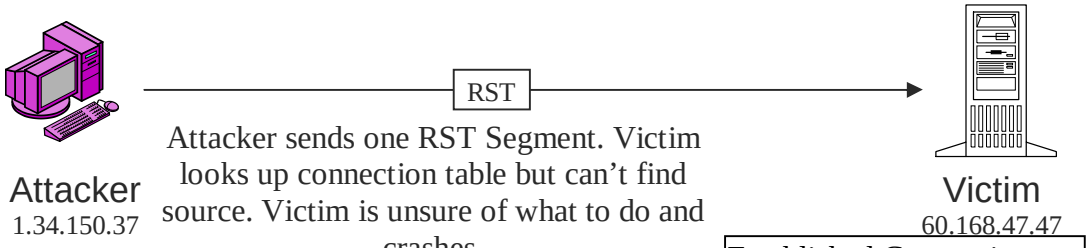
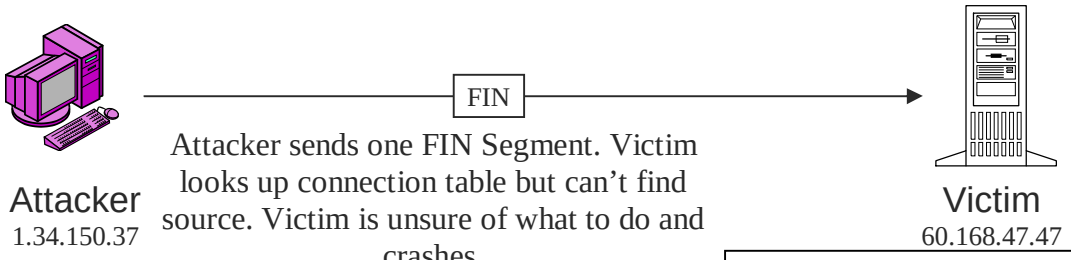


Fig 9: RST Attack

FIN Attack: It is similar to RST attack, the analyzer obtains packet information from the log file to find out if a packet containing the FIN bit of the FLAG field set has been sent from a source IP which never actually sent a SYN packet in the "Three Way Handshaking" rule of opening [8]. This scenario is represented in Figure 10.



Established Connections

Source(S) IP	S.Port	Port
192.168.0.5	4444	321
1.34.150.36	2021	80
1.34.150.35	1111	23

Fig 10: FIN Attack

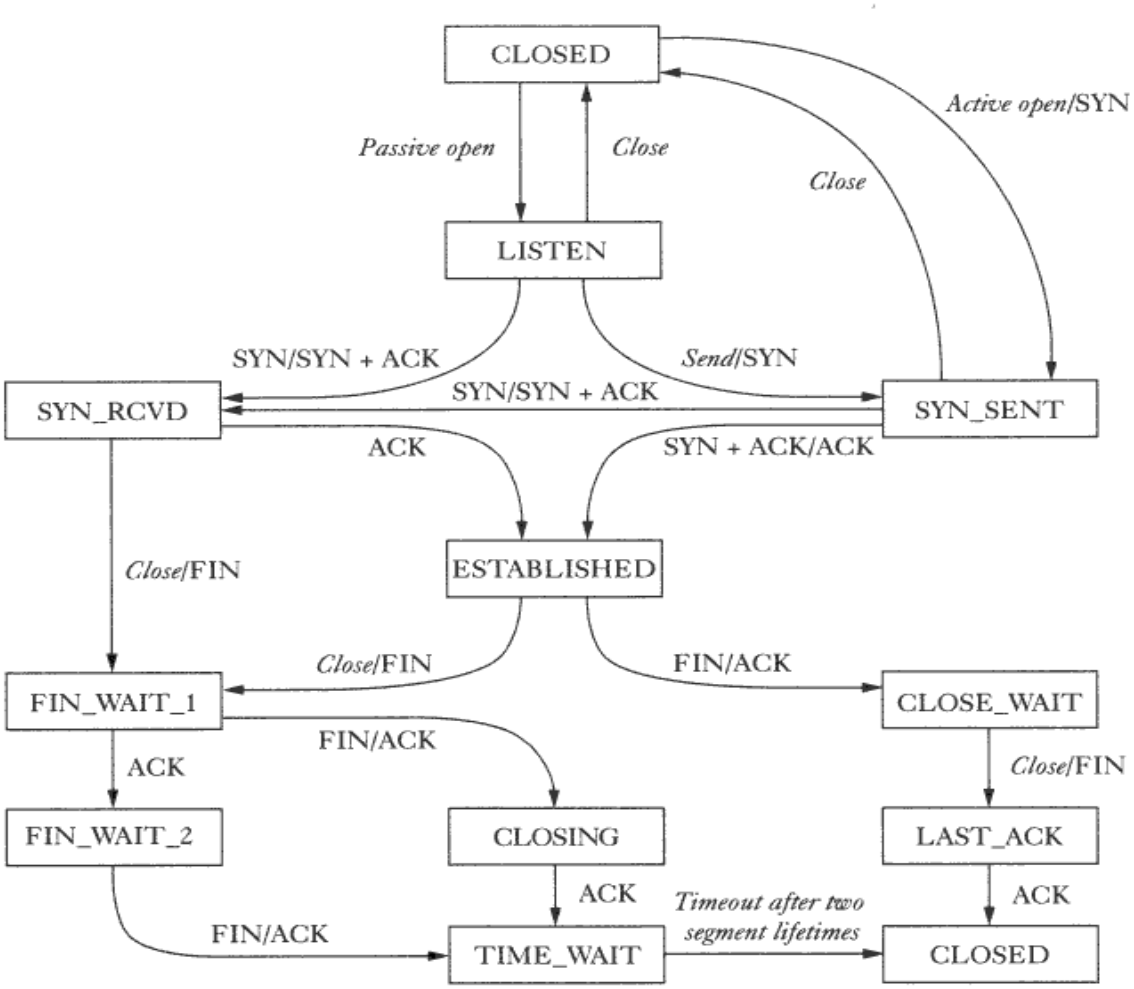
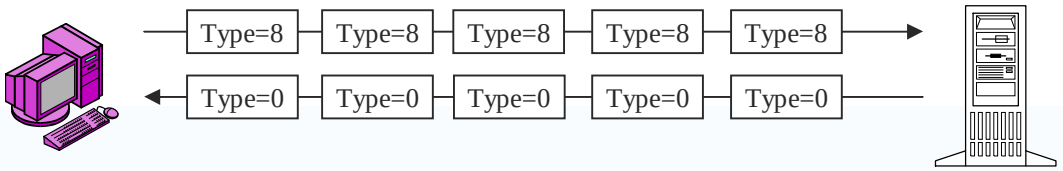


Fig 11: TCP State Diagram

A FIN attack is used to disconnect the client. However it concentrates on the end state of a TCP connection. The attacker tries to establish a series of new connections and closing them immediately without any data transfers. The idea is to keep the server busy maintaining the connection rather than actual or needed connections and eventually crash it with a large number of open and close connection requests.

5.2.2 Network layer

Ping Flood: Attacker simply sends a huge number of "ICMP Echo Requests" to the victim. This is an easy attack because many ping utilities support this operation, and the hacker doesn't need much knowledge. However, since it tends to overload network links, it is usually as detrimental to the attacker as to the victim, unless the attacker has a MUCH faster link than the victim



Attacker Sends Flood of ICMP packets with type = 8. Victim keeps responding to ping request through ICMP packets with type=0. Victim Crashes or Victim Becomes Too Overloaded to Respond to other legitimate users

Fig 12: DoS Attack - PING Flood

To reduce the effects of a ping flood, a victim can use a firewall to filter the incoming ICMP Echo Request packets entirely, or if a large number of requests are received at one time. Refusing to send ICMP Echo Reply packets produces two benefits:

- Less bandwidth is wasted by not answering these packets.
- It is more difficult for the attacker to measure the effectiveness of the attack.

However, such a filter will also prevent the measuring of latency from legitimate users which may be undesirable. A compromise solution may be to only filter large ICMP Echo Request packets, or to limit the rate at which your firewall will pass ICMP Echo Request packets.

Note that one cannot trust the source IP address to be the address of which the packets are originating from since it can be spoofed to make it appear as if it is coming from another address. Each packet can also be spoofed to contain a randomly generated address.

After evaluating the mechanisms of the above mentioned attacks, we tried to chalk out procedures that would identify such attacks, pinpoint vulnerabilities and take preventive measures. To do so, we started with the design of the proposed system, which has been discussed in the next section.

After identifying the attacks, understanding their mechanisms and identifying possible detection techniques, we discuss the design of the system in the next section.

6 DESIGN

The design of the system was initiated with the evaluation of various anti-intrusion techniques and then choosing from them the ones that could be used. Next, the overall architecture of the system was finalized and based on this architecture, components of the IDS has been placed at different points of a hypothetical network giving us the network architecture for the system. Finally, logics for the attacks that we have handled were worked out.

6.1 Defense Mechanism

According to [1], there are six general, non-exclusive approaches to anti-intrusion techniques: preemption, prevention, deterrence, detection deflection, and countermeasures (see Figure 13).

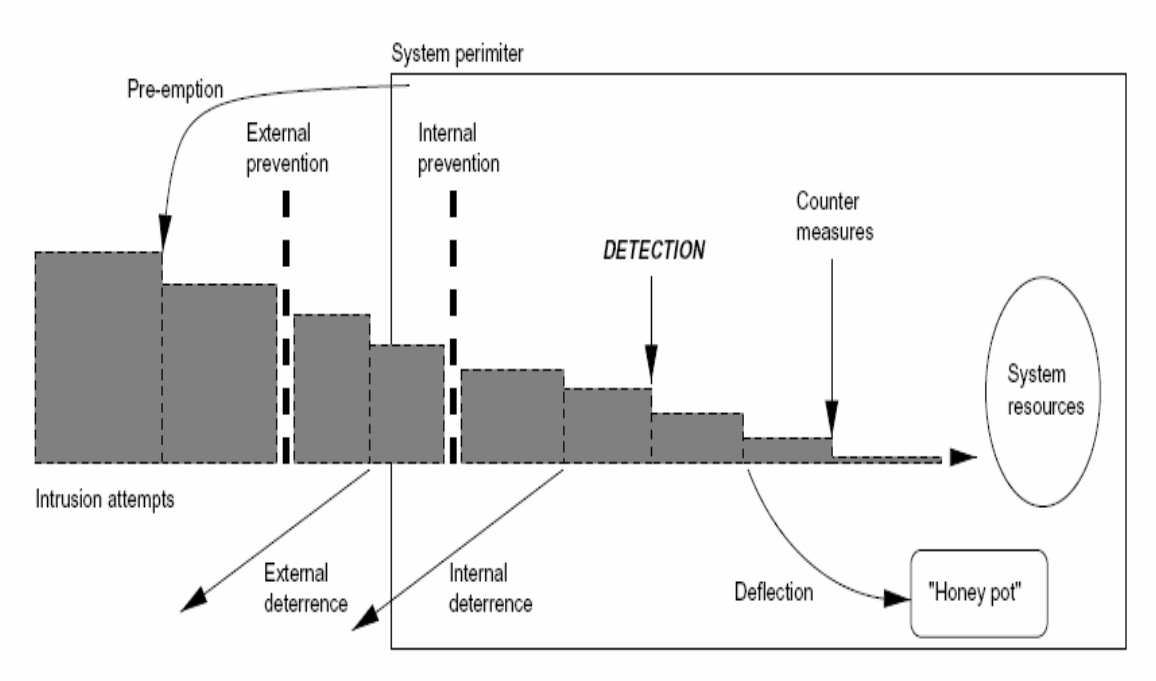


Fig 13: Anti-intrusion techniques (from [1]) or Defense Mechanism

- Pre-emption- This approach involves striking of against the threat before it can launch its own attack. This is an aggressive approach, but can be socially and legally inadvisable as both the prospective attacker and the innocent may fall victim. One way we are using preemption in our system is making the switch drop packets after it has reached it. This way, a preemption is enacted that stops the passage of malicious packets in a network before it reaches the destination.
- Prevention- This requires a system to undergo certain measures that eradicates any possibility of an experience to an attack. For example, for fear of external attacks, a system may be forced not to use the Internet or if a firewall is used restrictions are imposed on the system. These may ensure safety, but it can be expensive and awkward at the same time. In our system the use of firewall makes it possible to facilitate prevention. Firewall filters out the most obvious attacks preventing a possible attack. We have also used a preventive measure after the system has been infiltrated by an intrusion. The source IP from where the attack has been originated is recorded in a file by the analyzer and the switch that sniffs (sensor) for packets looks up the file when the same source sends more packets. In that case, the switch acts intelligently by dropping the packet from the attacking source preventive a possible attack.
- Deterrence- This approach is used to persuade a prospective attacker from launching an attack or an attacker in action to discontinue an attack. This is done by a number of ways varying from warning banners and alarms that threaten an attacker of server consequences to legal ways restricting computer crimes. This is technique has been used in our system as the SSO can send warning to the attacker or the domain under which the attacker is logged in if the attacker is unreachable. The warning can be sent in email or by phone. This step may scare the attacker and he may not attack the network afterwards.
- Detection- It identifies possible intrusion attempts and vulnerable hosts and data links so that suitable responses can be made. This takes place once an attack has been launched. During this, it is ensured that false alarms or inability to raise an alarm is avoided. Detection is used to identify a successful or an attempted security breach. The proposed system is primarily based on

this technique. Our system generates an alarm once an attack has been launched the first time. The system works more on the detection of threats and informing the SSO about them.

- Deflection-It is a method that tricks the attacker away from an area of a system, where he could effectively cause damage. It is very difficult to use this approach in diverting attentions of an experienced attacker for a considerable length of time. In our system, to divert attention of the attackers, the SSO can use a machine as “honey pot” which means a machine to attract attackers. He can set a rule in the firewall or configure the switches in such a way that any packet to the machine set as “honey pot” is sent directly to it and not any other machine. If the attackers see that that machine can easily be attacked, they will direct their attacks towards that direction. Additionally, the SSO can change the name of the machine to one that shows it is a very important machine. Examples of such names could be “Financial Server”, “database server” and so on. This way the attackers will forget about attacking other machines which may be very important to the network.
- Countermeasures-This is another aggressive approach where intrusions are ‘actively’ and ‘autonomously’ countered. In this approach, it is not taken into consideration whether a user of attack is a legitimate one or an intruder. For such scenarios, detection is not needed. An SSO using the proposed system this paper presents makes use of the report that highlights the attacker IP and Port. Even if an attacker cannot successfully attack after the firewall has been tuned to block such attacks, the attacker may render the firewall unusable by sending too many packets that the firewall cannot handle. This way the firewall will be too busy dropping packet. In such cases, countermeasures have to be taken by attacking the attacker whose IP address and port numbers has been obtained from the report. As already mentioned, innocent IP addresses being spoofed by an original attacker may also suffer, but it would be necessary to do so as otherwise, the network may get jammed with too much attacks. Moreover, this technique will scare the attacker off permanently.

6.2 Overall Architecture

Figure 14 illustrates some of the most essential components of a ‘typical’ IDS. We use the term a ‘typical’ IDS because this illustration is based on some of the available ones in the field. Please, keep in mind that the diagram does not show all data/control flows but only the important ones.

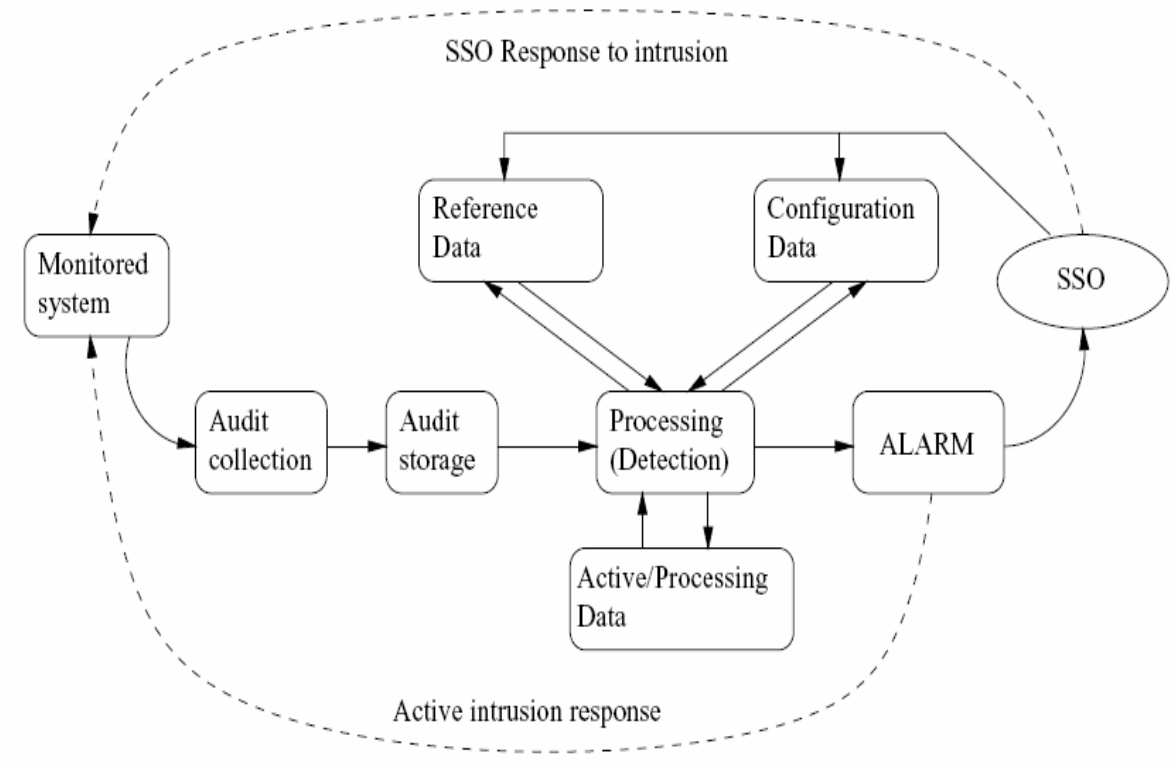


Fig14: Organisation of a generalised intrusion detection system

The components are:

- Audit collection- It is also called collecting log data which is used to make intrusion detection decisions. In the proposed system, switches are assigned responsibility to collect audit.
- Audit Storage-The collected data is permanently or temporarily stored in log files which are extremely large. These files are very important components in any IDS. Some researchers consider this as one of the problems in most IDS and an area which is not as often answered as it should be [7]. We have gone

by the advice of these researchers and have kept a included a logging module in our system. To decide what kind of data to store, we used Ethereal, a packet sniffing tool, to gather information about network traffic. Based on this, decisions regarding the type of data to be audited have been made.

- Processing- This is the core of an IDS. It uses various algorithms and techniques to detect traces of intrusion. In our proposed system, this is basically a program containing a number of classes written in Java. This program can be stored in a different machine connected to the server or in the server itself. Details of this step will be discussed in the next section.
- Configuration Data- This data is used to control the operations of an IDS. It provides information about the location and process of collecting data, how to attend to intrusions, what type for response, active or passive, is required, etc. This data can also be secretly used by attackers who use it to modify the IDS' behavior to suit their needs. Therefore, this data should also be protected from possible attacks. In our system, we have kept this category of data static as the SSO will not be able to modify this data to change any of the settings. This is an issue we are considering for future research.
- Reference Data- This stores data about previously encountered and identified types of intrusion. It reflects signatures and patterns of known intrusion types. Moreover, through the course of time, it is advisable that this data is updated when new intrusions are detected. To do so, information from an outside source is used. The research IDS uses a file to store information of the types of attack we have worked on. This information tells the analysis engine whether one or more packets contribute to a threat.
- Active/processing Data- This state helps in storing intermediate results. An example can be information about partially fulfilled intrusion signatures or patterns. This data can grow to a great volume, which has to be kept in mind when an IDS is designed. This is another area, we haven't been able to work on due to lack of resources.
- Alarm- This state handles outputs from the system and can be either passive, an interrupt generated to acquire immediate attention of an SSO and/or a report that he can view at a time of convenience to him, or active that responds automatically without the SSO's involvement. Our work involves

passive reaction to an attack. The proposed tool simulates the attack in graphical representation which will be discussed later in the paper.

To sum the whole architecture of the IDS used in our system, we have included audit collection, audit storage, processing, reference data and alarm at the moment and kept the option of combining the remaining two units: configuration data and active/processing data open for the future. At the end of the discussion of the design of the network architecture, a mapping has been provided showing at what points of the referenced network, these components have been used.

6.3 Network Architecture

The design of the “proposed” system is based on hypothetical network, following a star network topology, which is similar to the architectures typically followed in Bangladesh. A hypothetical network architecture has been selected due to lack of time and access to a real network. Moreover, the chosen architecture can be considered as a subset to almost any type of architecture. The chosen network architecture is composed of a server computer, a number of switches and host computers. Therefore, if a proposed IDS for such a network is successful, other IDS’ based on similar design will also be successful in effectively safeguarding a system from intrusions. Figure 3 represent such a network with various units of our IDS.

There are two possible scenarios of attacks: an external attack via the internet or an internal attack via another host of the same system. In our research, we have tried to work on some of the attacks generated both externally and internally.

In Figure 15, the “Filter” unit is for the purpose of external attacks. External packets entering into the system via the internet are first filtered. Proven harmful packets, such as packets coming from blocked addresses, are blocked while allowing the rest. The server then disseminates the packets within its system. Audit collection, as mentioned in above in Figure 14, is achieved by placing sensors at various points to capture network data (packets) by these sensors and logged in a file lying at a remote computer used by the SSO for maintaining

security. In order to do so, the data collected by the sensors are then sent to the remote host where they are stored in text files in a specified format.

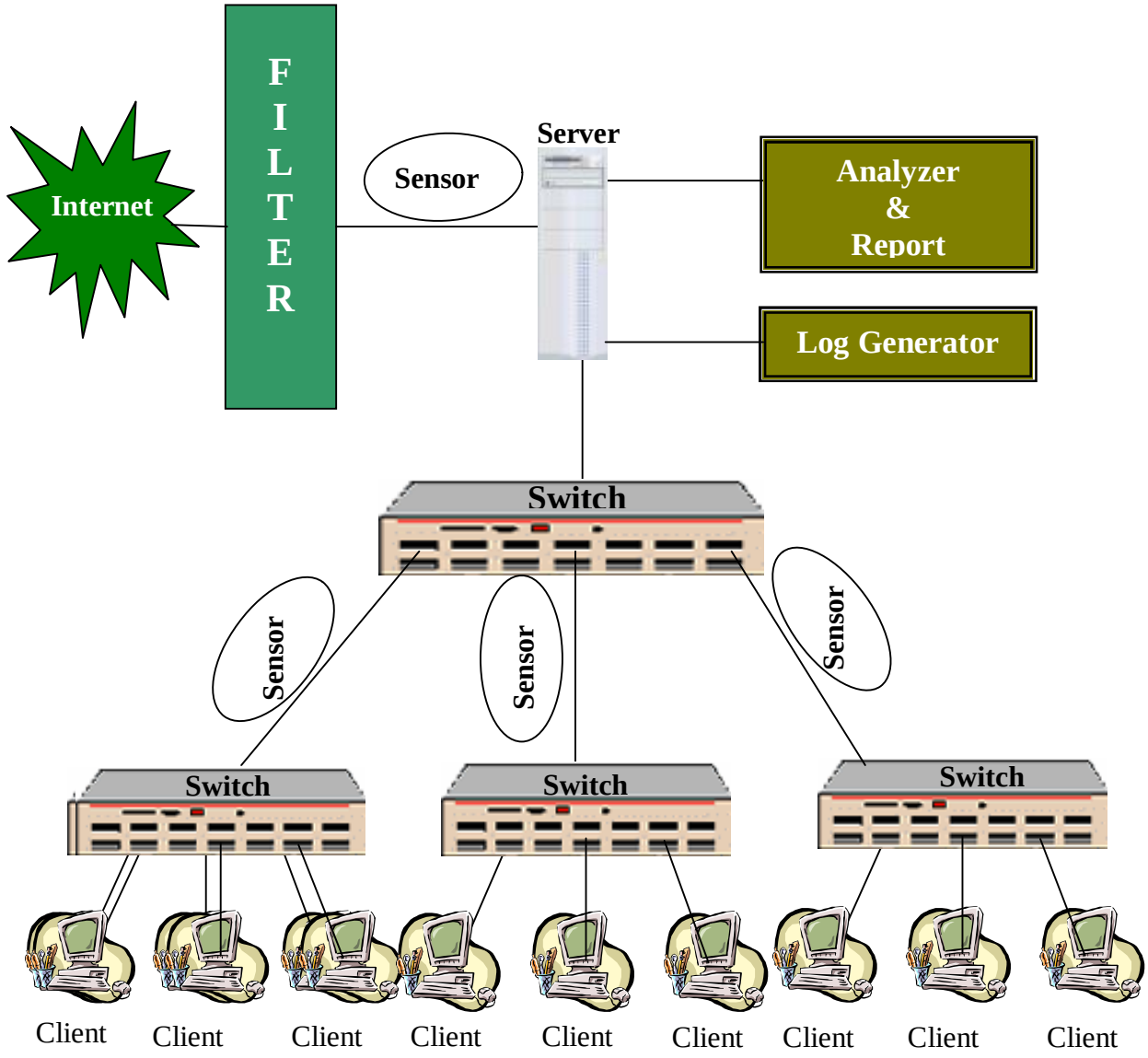


Fig 15: Hypothetical Network Architecture

The “Analysis Engine” uses the logged data from log files called as “Audit Storage” in Figure 14. The analysis is done based on these logged data. Once the analysis is done, an alarm and/or report is generated. Based on the IDS Configuration, an alarm and report is generated when the SSO wants an active reporting while only a report is generated when a passive report is desired.

The distribution of the various components of the IDS is very crucial in determining the expected performance from the IDS. The actual distribution may vary from network to network . For example, the placement of sensors can vary depending on points crucial to the security of the network. The “Report Generator” and the “Analysis Engine” can be placed in the server or on a different host depending on the requirement of the SSO and the system administrator of the network.

Figure 15 represent a mapping of various components of the network architecture with those of the IDS architecture. Audit collection is carried out by sensor in the network architecture. Audit storage is done by the log generator while the processing is done using reference data from a file is carried out by the Analyzer. The Report Generator is responsible for generating alarms (and reports) existing in the IDS architecture.

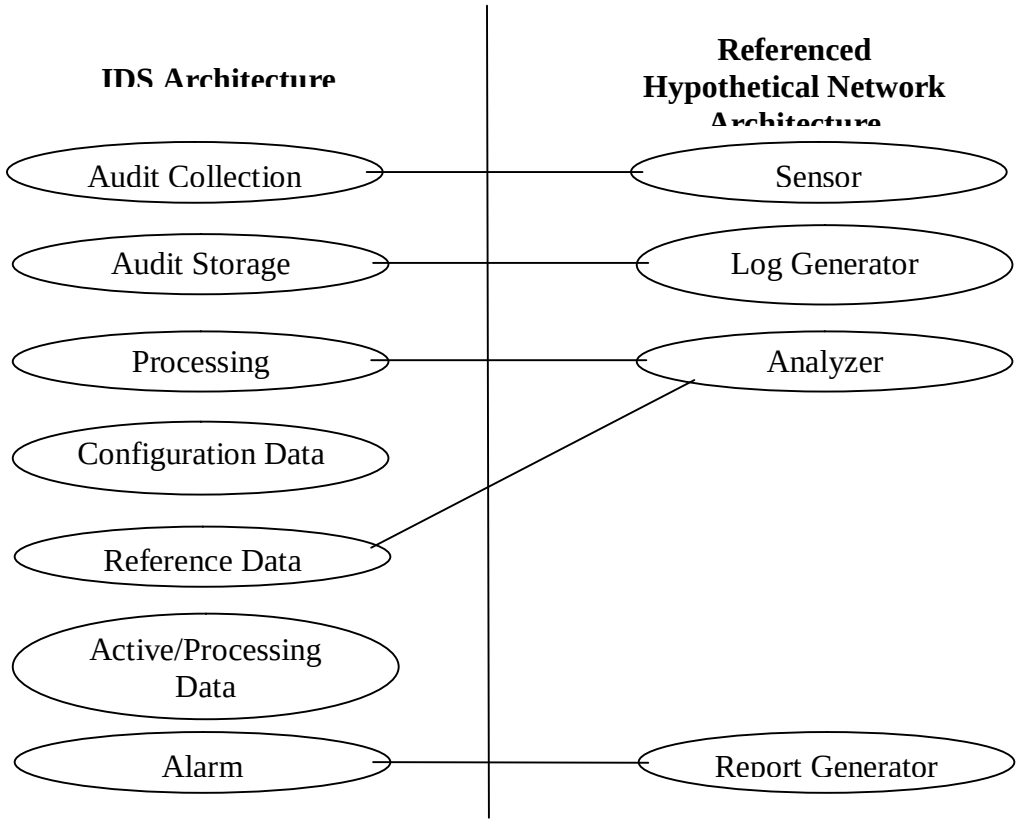


Fig16 : Mapping of IDS Components with Referenced Network Components

In the next section, we discuss the overall system based on the designs of this section. The system, in principle, follows the components of a typical IDS.

7 THE OVERALL SYSTEM

Our proposed system is based on a topic that has not yet been researched on as much. We have tried to design a system that will effectively tell an SSO which areas of the network has been compromised and what should be done in order to maintain the security of the overall system. This is done using Graphical User Interface. Diagrams modeling computers, servers and switches will indicate the operation of the network. Compromised hosts are marked in red and an advice note is attached at the bottom of the model in a list. Based on the advise note, an SSO takes action. The reports generated by the system will have data on types of attacks, source IP (attacker or intermediate node IP), destination (victim) IP, port numbers, and other useful information.

The whole system has been divided into a number of units, based on the organization of a typical IDS [1]. The units are:

1. Sensor: This is a program that sniffs packets from the network. Since we haven't had the opportunity to use a real network for data collection, this module is essentially a Client-Server program using Socket programming in Java. This program is used in a switch which receives data by opening one of its socket. To simulate sending of malicious packet, we have used another program that transforms packet information, both safe and malicious, into streams of bytes which is sent to the sensor. The sensor collects this information and simply forwards this to the recipient and the logging unit, the "Log Generator".

This unit also transforms the switch into an intelligent one by reading a file where the Analysis Engine writes. The file contains IP addresses of source machines that have previously launched attacks on the system. The switch uses this program to drop packets from such sources or allow those from others.

2. Log Generator: This unit takes data sent by the "Sensor" and filters those out that are for sure not attacking packets. In our case, every UDP containing IP packets are safe, so the unit filters them out. Thus, the remainder of the packets, both safe and unsafe, is logged.

3. Analyzer: This is the analysis engine, the heart of the system. It reads logged data and compares them to referenced data. Based on this comparison, it

comes to decision whether one packet or a collection of packets are aimed at intrusion. It also classifies an intrusion as one of the types: PING Flood (DOS), FIN Attack or RST Attack. It sends the information back to the “Network IDS Simulator”.

4. Network IDS Simulator: This is the interface between the SSO and the system. This is the control center of the whole system. The graphical window shows models of the computers in the network, a static representation for the time being. It has simulating dialogs to create and send safe packets and packets aimed at an attack. It has the responsibility to generate alarms originated by the “Analyzer”. It simulates the whole scenario in GUI modeling of a network. It simulates the movement of packets, as they move along the network. It highlights through use of colors (for example, red for attack, green for safe) and alphabets safe packets or attack packets. It also sets off the alarm by highlighting the victim host in red and adding an alert to a list of alerts at the bottom of the window. Moreover, it manages reports. It also is used to simulate creation of one or more packets aimed at attack or a safe packet. The actual IDS should not have this feature, but since this is a simulator and we haven’t had the opportunity to send real attack packets, we have used this to send information of attack and some safe packets to the switch.

5. Report & Alarm: The report generation is part of the responsibilities of the “Network IDS Simulator”, but it is treated as a different unit. There are two types of reporting used in the system. One is a report of all the threats since the simulator is started and the other is a report of all the threats the system has encountered in its lifetime. The alarm indicates movement of alphabets, “F”, “R” or “P”, in red from source to the destination and Log Generator and Analyzer via the switch simulating the passage of one or more attack packets being moved over the network in reality.

The overall system is more technically discussed in the next section. There, we discuss the development components used such as class diagrams, logic and so on.

8 IMPLEMENTATION

The design of our system “Visualization of Security Vulnerabilities through Intrusion Detection System” is component and class based. During class design, special attention was given to the basic components of a typical IDS and the designed classes incorporate these components. As already mentioned, an object-oriented principle has been used in the design, the resulting classes of the design represent an IDS with its attributes and functionalities.

The system has the following packages as shown in Figure 17:

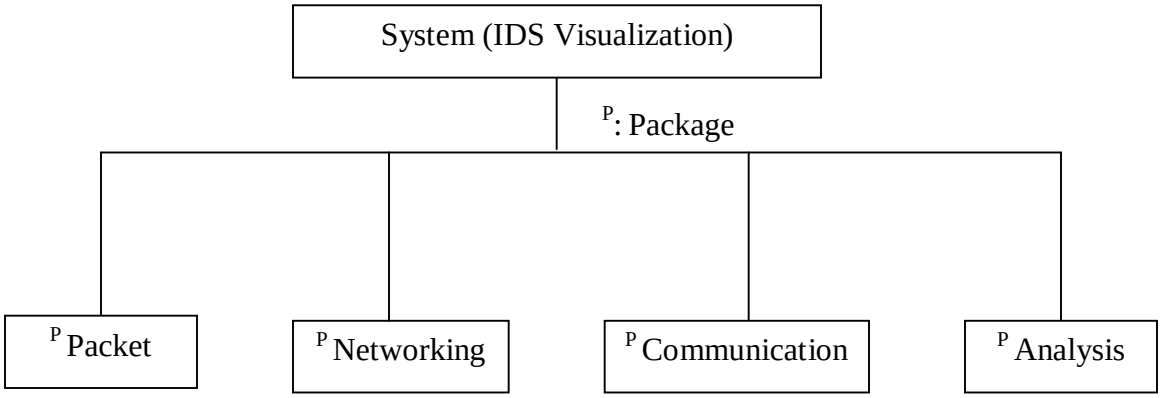


Fig 17: Packages of system

Each Package consists of several classes which are given in Table 4. The packages Networking and Analysis consists the components of the IDS. Additionally, the Networking package has classes that look after the simulation and the GUI. The package Communication consists of classes that help in establishing connections among remote hosts. The package Packet emulates frames and packets of Transport and Network layers of the TCP/IP protocol stack.

Table 4 : classes per package

Packet	Networking	Communication	Analysis
• IpPacket • ProtocolPacket • IcmpPacket • TcpPacket • UdpPacket	• Droppage • LogGenerator • Network • Node • OutputAttacks • Path • SimulatorDialog • Switch • ShowToday	• Client • ForwardServer • Server • ServerClientThread	• Attack • FinAttack • PingAttack • RstAttack • Analysis • Output

8.1 Class Diagram

Classes and objects do not exist in isolation from one another. Class Diagram shows a relationship which represents a connection among things. In UML, there are different types of relationships. Here we will represents two of them, which are: Generalization and Association Multiplicity. [19] [20]

8.1.1 Generalization

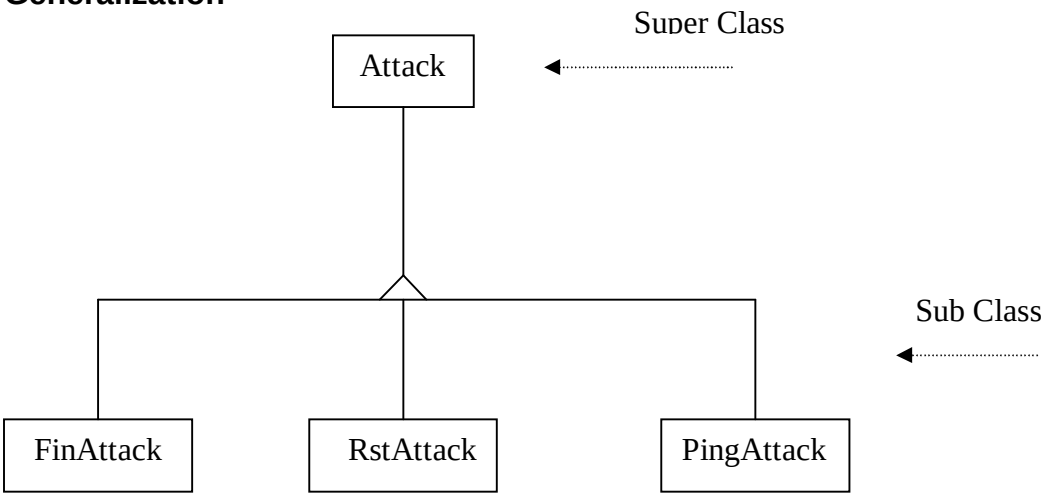


Fig 18: Attak Inheritance

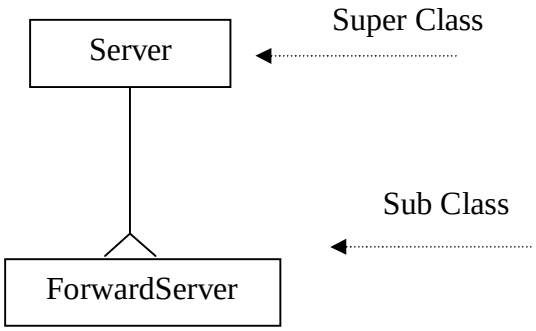


Fig 19: Server Inheritance

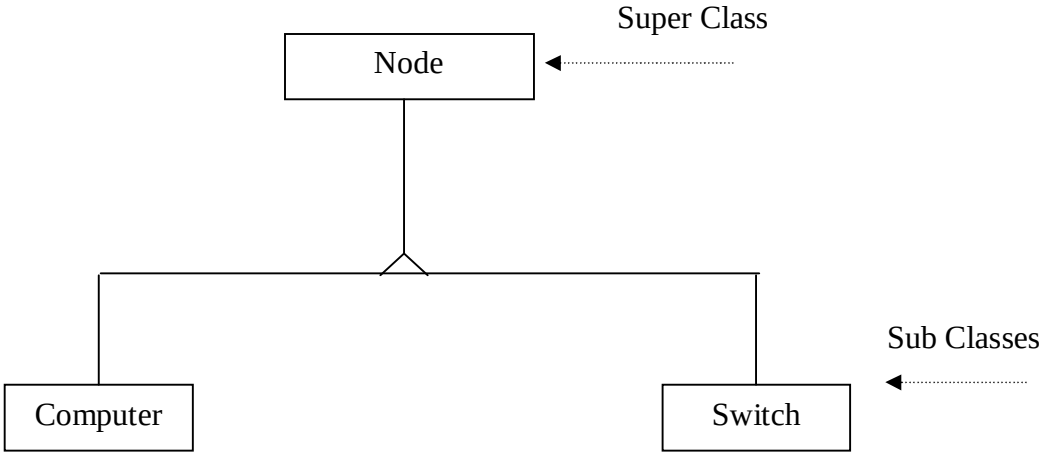


Fig 20: Node Inheritance

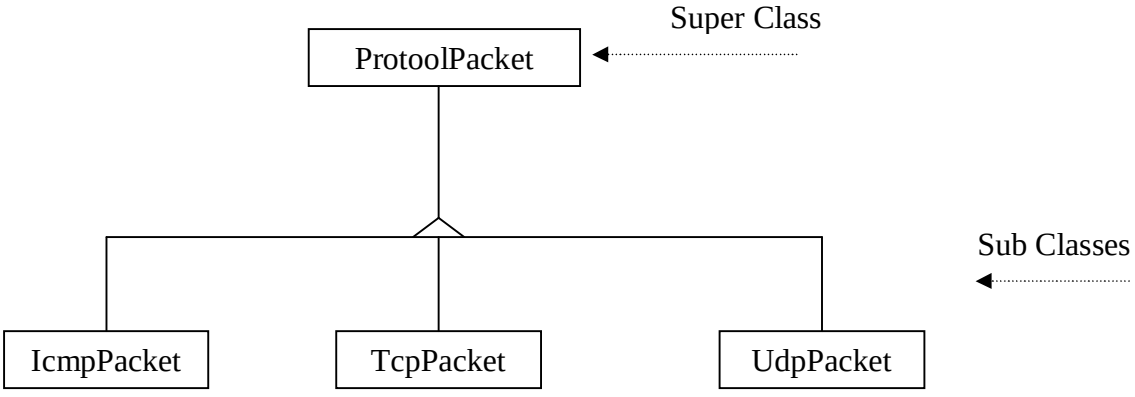


Fig 21: ProtocolPaket Inheritance

8.1.2 Association multiplicity

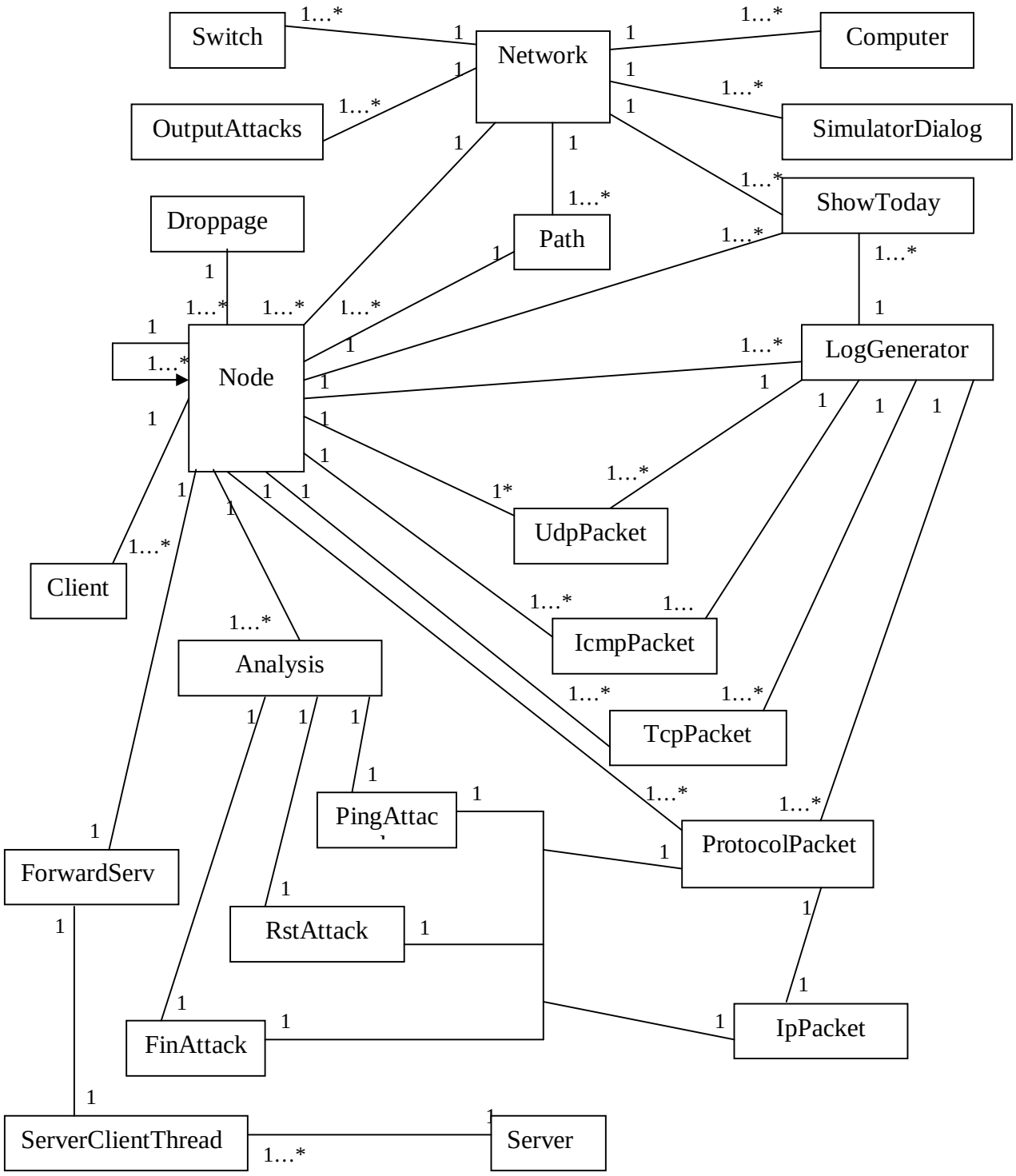


Fig 22: Relation between classes

8.2 Logic

PING FLOOD

Input: packets from log file, information from reference file

Process:

- read the ping attack row from reference file and take time limit, service type and maximum number of packets (which is safe within this time limit)
- read arrival time, source IP, destination IP, destination port of each packet from log file
- if service type is ping type (8) then count the number of packets from the arrival time of each packet to within time limit, which destination IP and destination port is same; also save the source IP of those packets
- check the number of packets with the maximum number of packets (from reference file)
- if it is greater than set alarm

Output: saved source IP

FIN ATTACK

Input: packets from log file, information from reference file

Process:

- read the fin attack row from the reference file and take fin flag, ack flag
- read ack flag from log file and create a table (for currently established connection)
- read fin flag from log file and create another table
- read source IP. destination IP, destination port from both of the table
- check the IPs and port of fin table with the the IPs and port of ack table, whether requested fin has a valid connection or not

- if the connection is not valid then save the source IP. destination IP, destination port of fin table and set alarm

Output: saved source IP. destination IP, destination port

RST ATTACK

Input: packets from log file, information from reference file

Process:

- read the rst attack row from the reference file and take rst flag, ack flag
- read ack flag from log file and create a table (for currently established connection)
- read rst flag from log file and create another table
- read source IP. destination IP, destination port from both of the table
- check the IPs and port of rst table with the the Ips and port of ack table, whether requested fin has a valid connection or not
- if the connection is not valid then save the source IP. destination IP, destination port of rst table and set alarm

Output: saved source IP. destination IP, destination port

8.3 GUI (graphical User Interface)

The software, IViS (acronym for Intrusion Visual Simulator) opens with a front window as shown in Figure 23. The components of the window are as follows: menu bar at the top, which gives the SSO the option of simulating generation of attack and safe packets and viewing of reports through the following options: Attack, Safe and View, list of alarms at the bottom, which includes all the alarms set off since the software is started and the main portion where the graphical simulation takes place.

At the left-bottom corner above the list of alarms, there is a legend that informs the SSO the meaning of each symbols used in the simulation. For example, a red “F” or “R” means FIN attack or RST attack respectively (as

already explained in the earlier section). Additionally, the red circle is carved over a host indicating the host has been attacked and should be disconnected from the network.

When the window starts, the middle portion draws a sub-set of the hypothetical network used in the research. In the center lies the switch with IP address 192.168.0.2 and right at it's top is the log generator and analyzer, a computer with IP address 192.168.0.3. The other three computers are clients.

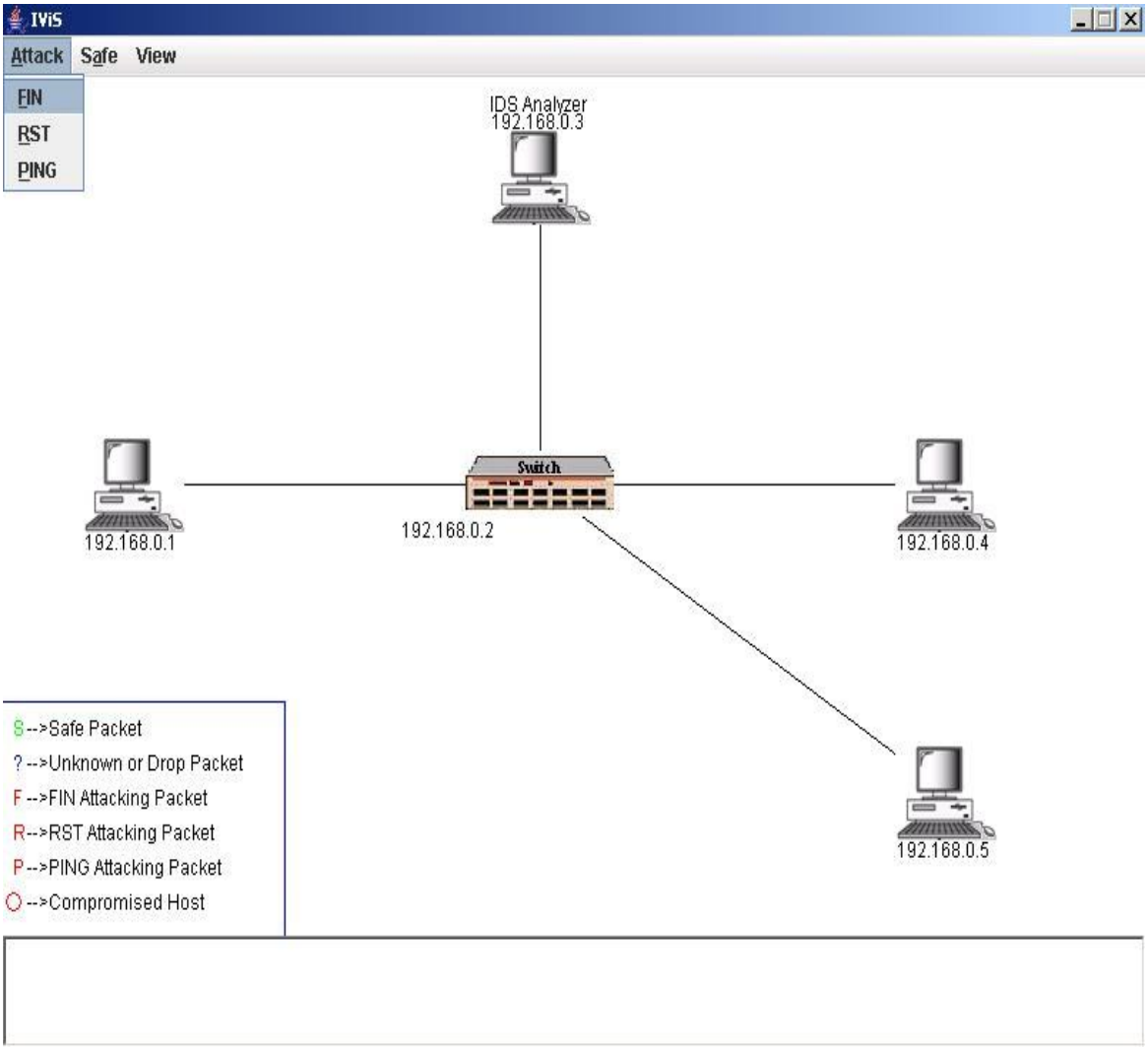


Figure 23: Front Window of IViS

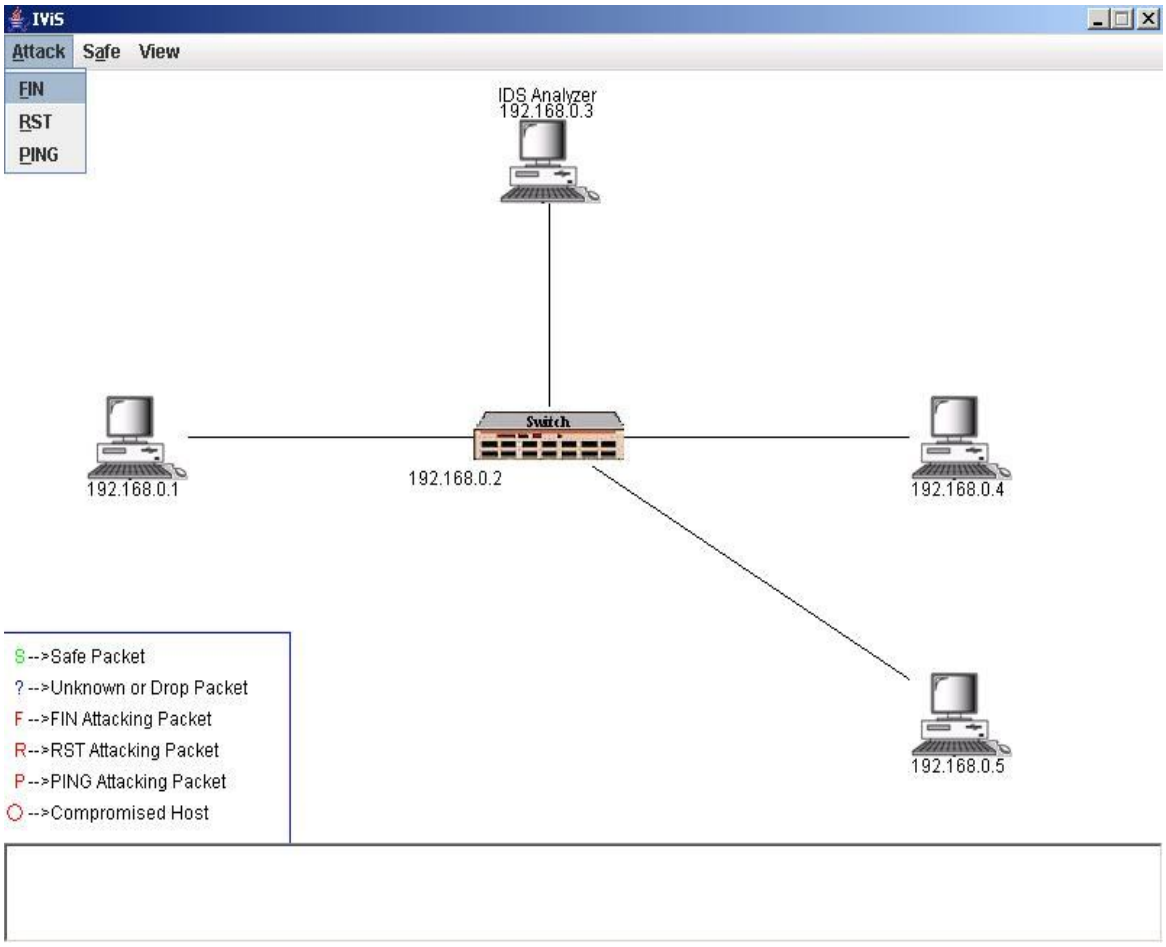


Fig 24 : Front Window of IViS

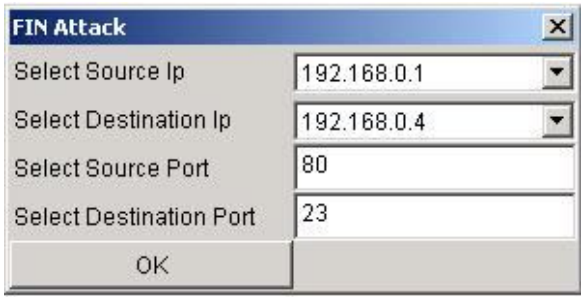


Fig 25: Dialog to Generate FIN Attack

Upon clicking on the item pointed by the mouse as shown in Figure 24, a dialog opens an example of which is the one in Figure 25. In the dialog, the user has selected one of the clients (hosts) as source, which is attacker and another as destination, which is victim. The user also has to choose port numbers, both

source and destination to launch attack. Once the “OK” button is pressed, the attack is launched and the actual task of the packet generation, transfer, sniffing, logging, analyzing and alerting begin. Other attack and safe packets can also be generated using similar dialog boxes.

In figure 26, the packet generated earlier is traced by the software. The network is yet does not know whether this packet sent from 192.168.0.1 (Figure 26) to 192.168.0.4 is FIN attack. So, the software shows its passage with a green colored S, meaning safe, from source to the switch.

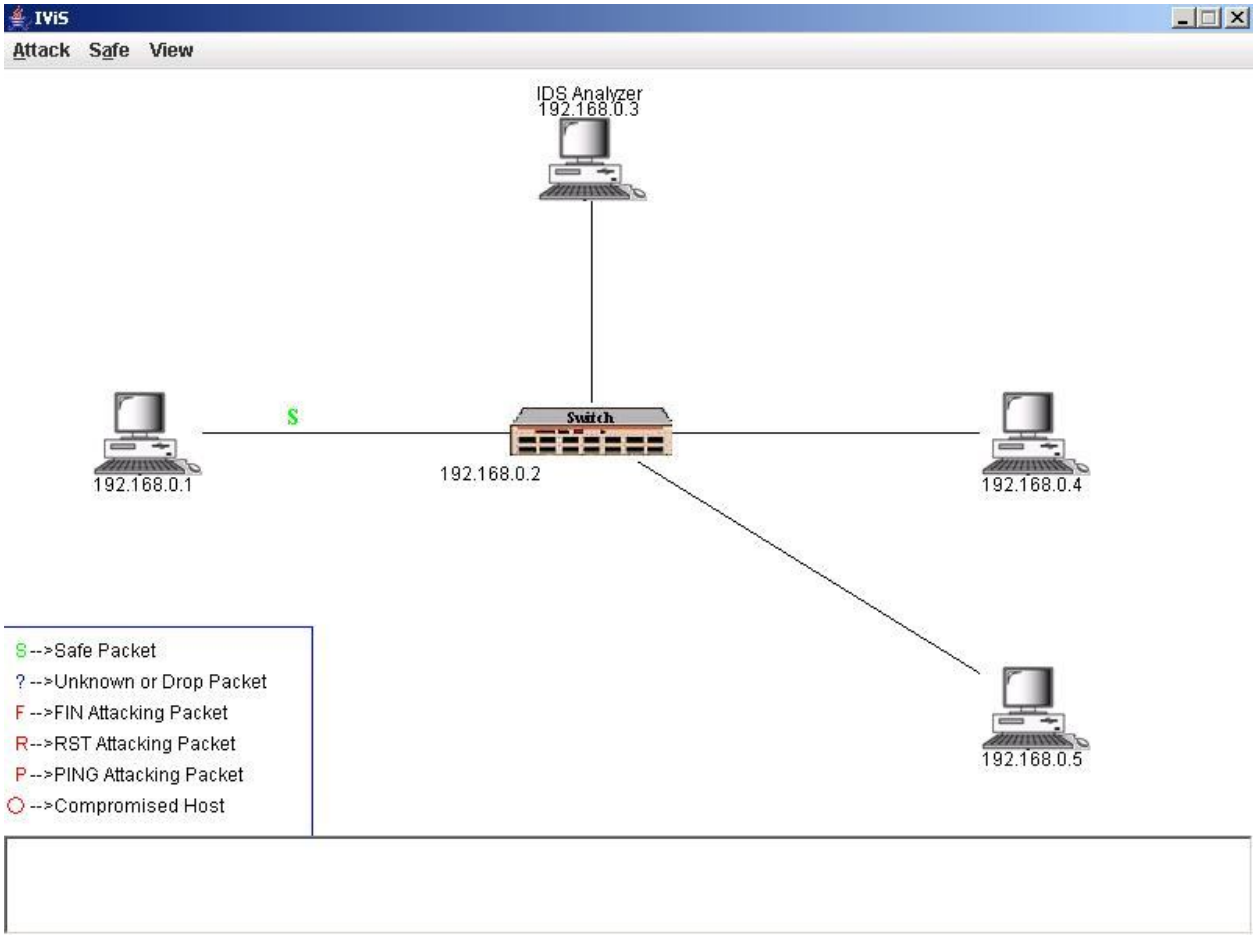


Fig 26: Packet from Host to Switch

The packet received by the switch is forwarded to the destination, taking it to be safe (green colored “S” in Figure 27). It also forwards a copy to the log generator/analyzer and it is represented as a blue “?” mark. This means that the network, more particularly, the analyzer does not know what type of packet it is: safe or attack.

After every interval of ten seconds, the analyzer gets log data from a file and analyzes the packets to check for attacks. For our example of FIN attack generation, the analyzer classifies the packet correctly as FIN attack. This is shown in Figure 28. Here, the packet transmission from attacker to victim is re-simulated but with real identity of the packet. An “F” in red is animated from attacker (192.168.0.1) via the switch (192.168.0.2) to the victim (192.168.0.4). Also, a copy of the packet is also traced, symbolized with a blue “?” mark, from the switch to the log generator/analyzer (192.168.0.3) showing that it does not worry about the identity of the packet until the analysis is done. This alert is generated by the Report Generator (in the background) which takes data from the analyzer and displays it on the main window. Moreover, the victim is marked with a circle in red attracting the attention of the SSO. A message is appended in the list at the bottom of the window showing the attacker and victim’s IP addresses and port numbers, the date and time of the attack and an advice note.

Similar to the FIN attack of our example, PING and RST attacks are simulated except that the PING attack is show as a flood of packets. In the same way, safe packet are simulated with the exception that no alert is generated indicating precaution taken by the system against false alarms.

If an attacker is identified, its IP address is stored in a file by the analyzer. This file is used by the switch to use in the preemption and preventive techniques discussed in Section 5. If such an attacker sends packets into the network via the switch, no matter what type of packets are these, looks up its IP address in the file and since it exists there, drops all maintaining the safety of the network (Figure 29). The question mark shows ignorance about the contents of the packet while dropping keeping safety as the first priority.

As Figures 30 and 31 show, we have two reports generated by the software for the time being. The first shows the IP addresses of hosts which have been blocked. The second shows history of all attacks. Currently, no sorting and searching options are available, but we plan to add them in the future.

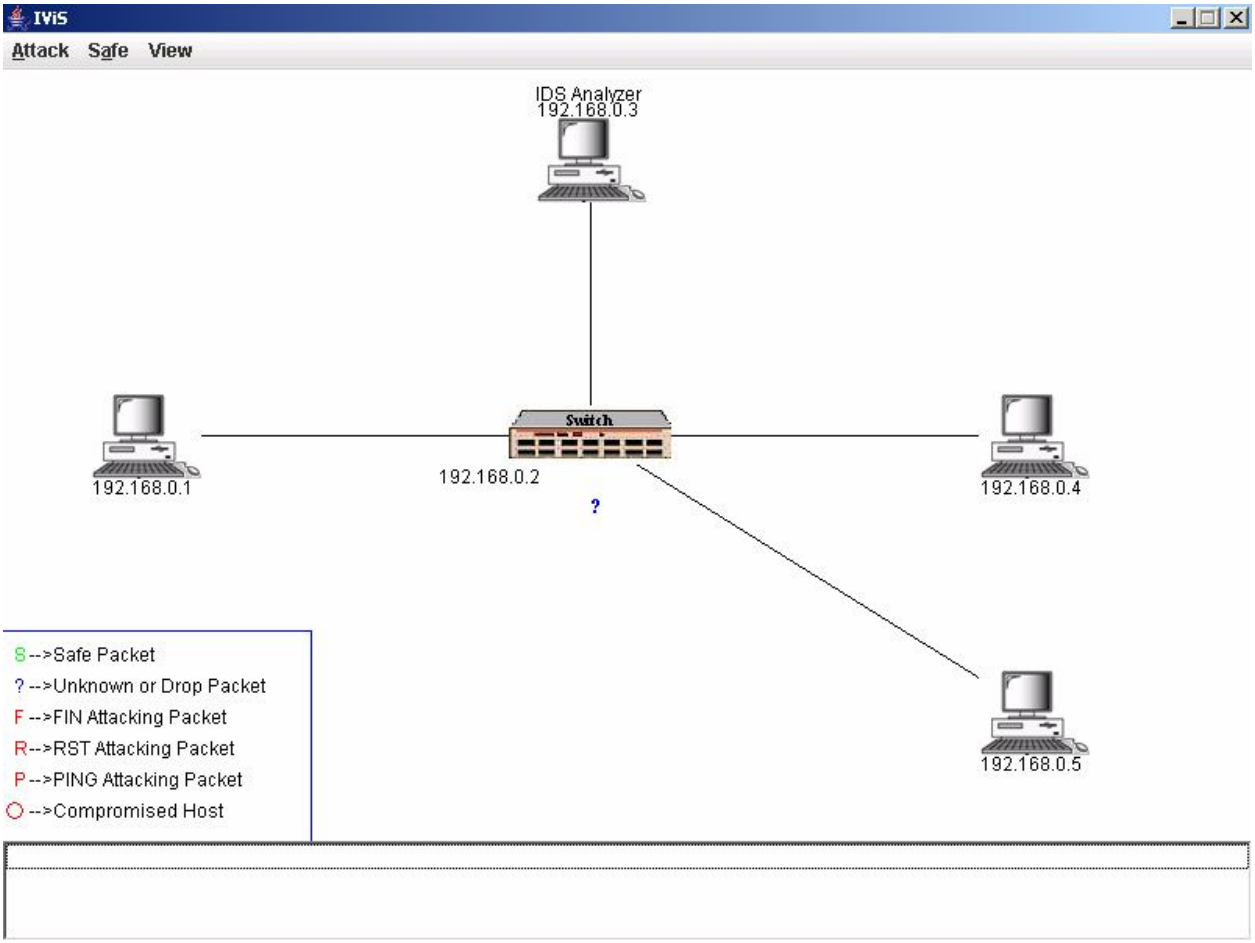
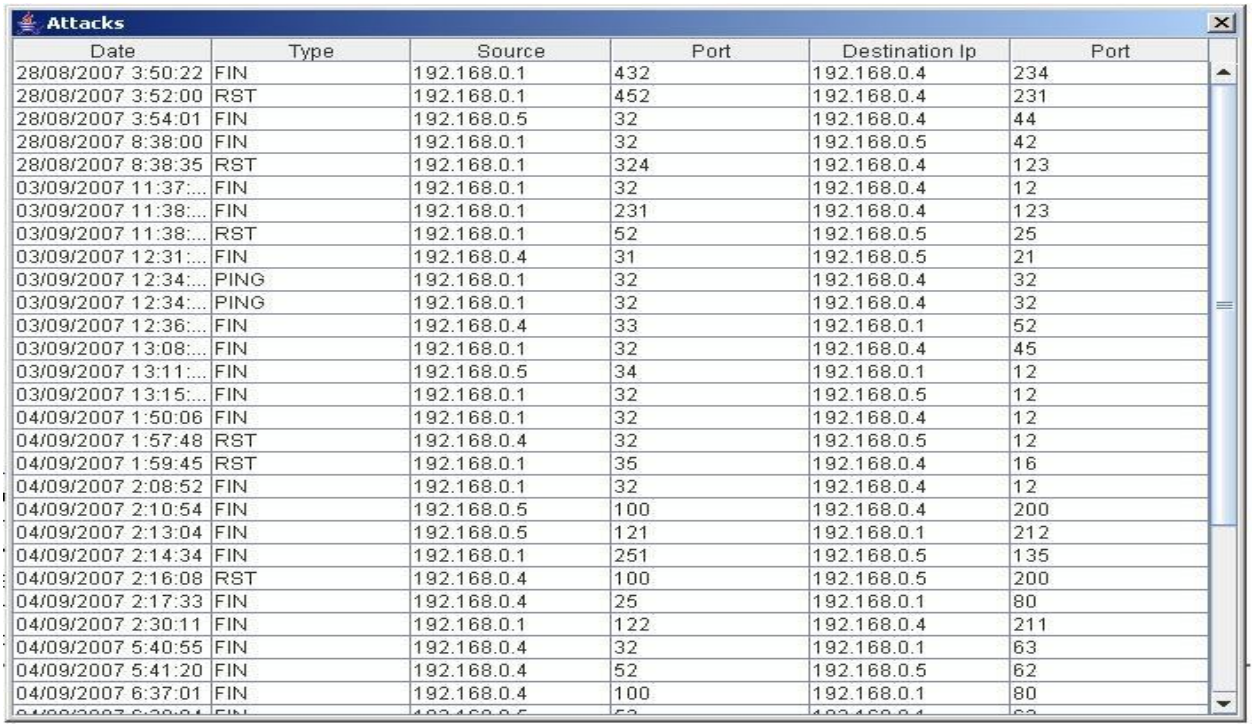


Fig 29: Packets Dropped

Fig 30: Report—Blocked Ips



Date	Type	Source	Port	Destination Ip	Port
28/08/2007 3:50:22	FIN	192.168.0.1	432	192.168.0.4	234
28/08/2007 3:52:00	RST	192.168.0.1	452	192.168.0.4	231
28/08/2007 3:54:01	FIN	192.168.0.5	32	192.168.0.4	44
28/08/2007 8:38:00	FIN	192.168.0.1	32	192.168.0.5	42
28/08/2007 8:38:35	RST	192.168.0.1	324	192.168.0.4	123
03/09/2007 11:37:...	FIN	192.168.0.1	32	192.168.0.4	12
03/09/2007 11:38:...	FIN	192.168.0.1	231	192.168.0.4	123
03/09/2007 11:38:...	RST	192.168.0.1	52	192.168.0.5	25
03/09/2007 12:31:...	FIN	192.168.0.4	31	192.168.0.5	21
03/09/2007 12:34:...	PING	192.168.0.1	32	192.168.0.4	32
03/09/2007 12:34:...	PING	192.168.0.1	32	192.168.0.4	32
03/09/2007 12:36:...	FIN	192.168.0.4	33	192.168.0.1	52
03/09/2007 13:08:...	FIN	192.168.0.1	32	192.168.0.4	45
03/09/2007 13:11:...	FIN	192.168.0.5	34	192.168.0.1	12
03/09/2007 13:15:...	FIN	192.168.0.1	32	192.168.0.5	12
04/09/2007 1:50:06	FIN	192.168.0.1	32	192.168.0.4	12
04/09/2007 1:57:48	RST	192.168.0.4	32	192.168.0.5	12
04/09/2007 1:59:45	RST	192.168.0.1	35	192.168.0.4	16
04/09/2007 2:08:52	FIN	192.168.0.1	32	192.168.0.4	12
04/09/2007 2:10:54	FIN	192.168.0.5	100	192.168.0.4	200
04/09/2007 2:13:04	FIN	192.168.0.5	121	192.168.0.1	212
04/09/2007 2:14:34	FIN	192.168.0.1	251	192.168.0.5	135
04/09/2007 2:16:08	RST	192.168.0.4	100	192.168.0.5	200
04/09/2007 2:17:33	FIN	192.168.0.4	25	192.168.0.1	80
04/09/2007 2:30:11	FIN	192.168.0.1	122	192.168.0.4	211
04/09/2007 5:40:55	FIN	192.168.0.4	32	192.168.0.1	63
04/09/2007 5:41:20	FIN	192.168.0.4	52	192.168.0.5	62
04/09/2007 6:37:01	FIN	192.168.0.4	100	192.168.0.1	80

Fig 31: Report----History of Attacks

8.4 Evaluation

The proposed system is very useful in serving as the prototype of a full-fledged IDS. The system has some unique features and other positive sides. At the same time, there are some short-comings that have to be overcome in the future.

The system has both textual and visual output. A visual output where the SSO can pin-point the problem graphically is very helpful in presenting the attack more vividly which would be hard to miss as usually with the case with textual output. Additionally, the outputs are stored in a file as history. This history stores information about all the attacks that have been experienced by the network.

As this IDS is network-based, it protects the overall network from both inside and outside attacks. Moreover, since the logging unit and analysis engine are centrally located and only the sensors are distributed, it would be easier to provide safety to the components of the IDS. This is so because it would be

easier to isolate the machines hosting these units from the rest of the host machines as only the sensors would know where to send data for logging and only the IDS console would know the whereabouts of the analyzer and the report generator. Otherwise, the IDS itself would be very vulnerable to intrusions.

The proposed IDS is built on object-oriented principles making it easy to maintain and add new features and components such as configuration data and active/process data. Another important advantage of the system is that, classes for new attacks can be written and easily incorporated in the system increasing the scope of handling attacks. Similar expansions are possible in report generation.

The proposed IDS is expected to be cheaper than the ones available in the market. The use of open-source programs such as ethereal makes it economically feasible to use them in the system without worrying about buying them. The system does not use any special hardware apart from existing machines such as switches, routers and computers. Therefore, no added cost of hardware will have to be borne. The cost of maintenance is also expected to be low due to lower complexity in adding components and features.

As already mentioned, besides the advantages, there are also some short-comings of the system. This includes a small volume of reference data that is suitable to handle only limited number of attacks. The number of attacks handled should be increased before the system can be used in reality. Also, it has not been possible to test the system on a real network. One other disadvantage of the system is that currently, no module has been written or included that can sniff real packets from the network. This has been further discussed in the next section.

9 CONCLUSION AND FUTURE WORK

The basic goal of the proposed system is to serve as a stepping-stone to a much efficient IDS. Moreover, the design issues of the system have been considered keeping companies and organizations of Bangladesh in mind. Due to lack of resources such accessibility of sensitive system resources and permission to conduct mock attacks, the implementation of the proposed system has not been carried out. The designed and implemented system has some unique features which can be utilized to develop a full-fledged IDS with a visual simulator to prevent and detect attacks and manage reports and alarms so that the three aspects of a system: confidentiality, integrity and availability can be maintained.

This system can serve as a prototype of a full-fledged IDS and a visual tool for generating alarms and reports of threats. More types of attacks can be handled by adding modules in the already existing basic one. Moreover, real packet sniffers, such as Ethereal which is an open source utility, can be used to detect packets from the network card. On the contrary, a module for the purpose can be written and included in the existing system.

Another issue that needs more research is in the area of logging. It is an issue of huge importance and an area of target by attackers [7]. The log and reference files should be secured by encryption or any other mechanism. Additionally, a configuration unit is also required to add dynamics to the system.

APPENDIX

Classes

P ANALAYSIS

Attack

private:
`sourceIp: String`
`destIp: String`
`protected:`
`attackOutputs: Vector`

public:
`Attack(String sourceIp,String destIp)`
`getDestIp():String`
`setDestIp(String destIp): void`
`getSourceIp():String`
`setSourceIp(String sourceIp): void`
`getAttackOutputs():Vector`

FinAttack
private: finflag: String ackFlag: String info: String
public: `String sip,String dip)` checkAttack(Vector v): int reference(String fileName): void getAckFlag() : String setAckFlag(String ackFlag) : void getFinflag() : String setFinflag(String finflag) : String toString():String

PingAttack
private: limit: int time: int info: String

public: checkAttack(Vector v): int reference(String fileName): void getLimit() : int setLimit(int limit) : void getTime() : int setTime(int time): void toString():String

RstAttack
private: rstflag: String ackFlag: String info: String dport: int
public: checkAttack(Vector v): int reference(String fileName): void getAckFlag() : String setAckFlag(String ackFlag) : void setRstflag() : void setRstflag(String rstflag): void toString():String

Analysis
private: output[]: String numAttack: int parent: Node *period*: int *filename*: String attackOutputs: Vetor numFile: String *lineCount*: int newAttack: Boolean
public: run(): void getOutput():String[] main(String args[]): void packetGeneration(String fileName): void defineAttack(Vector v): void

Output
privae: sourceIp: String

destinationIp: String sourcePort: int destinationPort: int attackType: String
public: Output(String sourceIp, String destinationIp, int sourcePort, int destinationPort, String attackType) getAttackType() : String getDestinationIp(): String getDestinationPort():int getSourceIp():String getSourcePort():int toString():String

P PACKET

IpPacket
private: version: int headerLen: int totalLen: int protocol: String source: String destination: String protocolPacket: ProtocolPacket dateTime: String
public: getDestination():String setDestination(String destination): void getHeaderLen(): int setHeaderLen(int headerLen): void getProtocol(): String setProtocol(String protocol): void getSource(): String setSource(String source) : void getTotalLen(): int setTotalLen(int totalLen): void getVersion(): int setVersion(int version): void getProtocolPacket() : ProtocolPacket setProtocolPacket(ProtocolPacket protocolPacket) : void getDateTime() : String

setDateTime(String dateTime) : void toString():String

ProtoolPacket
private: source: int destination: int
public: getDestination(): int setDestination(int destination) : void getSource() : int setSource(int source) : void toString() : String

IcmpPacket

private:
`sequence: int`
`type: int`

private:
`getSequence():int`
`getSequence(int sequence): void`
`getType():int`
`setType(int type): void`
`toString():String`

TcpPacket
private: sequence: int headerLen: int flag: String
public: getFlag():String setFlag(String flag) : void getHeaderLen():int setHeaderLen(int headerLen): void getSequence():int setSequence(int sequence) : void toString():String

P COMMUNICATION

Client
private: clientSocket: Socket message: String
public: waitForMessage(): void send(String message): void getClientSocket(): Socket setClientSocket(Socket clientSocket): void

DriverClient
private: client: Client
public: sendMessage(String message): void waitForMsg(): void main(String args[]): void

ForwardServer
private: caller: Node
public: caller: Node obtainMessage(ServerClientThread thread,String message): void

Server
private: serverSocket: ServerSocket clients: Vector clientMessage: String
public: run(): void sendMessage(String remoteHost, String message): int broadcast(String[] receipients,String message): void obtainMessage(ServerClientThread thread,String message): void getClientMessage(): String setClientMessage(String clientMessage): void

ServerClientThread
private: socket: Socket server: Server message: String
public: run(): void send(String message): void getServer(): Server setServer(Server server): void getSocket(): Socket setSocket(Socket socket): void

P NETWORKING

Computer
private: img: Image
public: Computer(String name, Network parent, int x, int y,String role) getImg(): Image

Droppage
private: Node source: Node Type: String G: Graphics
public: Droppage(Node node, String type, Graphics g) annimate(int sourceX, int sourceY, String symbol,Color symbolColor, Font font): void run(): run

LogGenerator
private: logFile: String packet: IpPacket
public: LogGenerator(String logFile) generate(String packetString): void stringToPacket(String packetString): IpPacket main(): void

Network
private: name: String

nodes: Vector
forwarder: String
analyzerLogger: String
display: list
aLPort: int
netClientPort: int
thread: Vector
menuBar: JMenuBar
public:
ipHLen: Vector
tcpHLen: Vector
udpHLen: int
switchPort: int

public:
Network(String name)
actionPerformed(ActionEvent e): void
draw(Graphics g): void
addComputer(String name,int x, int y,String role,Graphics g): void
addComputer(String name,int x, int y,String role,Graphics g): void
paint(Graphics g): void
tracePath(String source, String destination,String type): void
dropPacket(String source, String type): void
getComputer(String ip): Computer
getAnalyzerLogger(): String
setAnalyzerLogger(String analyzerLogger): void
getForwarder(): String
setForwarder(String forwarder): void
getReport(String output[]): void
generateAlarm(Vector attackOutputs): void
markNode(String victim): void
addThread(Thread th): void
main(): void

private:
innitializeNodes(): void

Node

private:
parentNet: Network
xPos: int
yPos: int
connectedNode: Vector
server: ForwardServer
clients: Vector
clientIps: Vector
client: lient
bannedIpFil: String

proteted:
name: Sting
public:
role: String

public:
Node(String name,Network parent,**int** x, **int** y,String role)
getName(): String
getRole(): String
setRole(String rol): void
setName(String name): void
getXPos): int
setXPos(**int** pos): void
getYPos(): int
setYPos(**int** pos): void
getConnectedNode(): Vector
setConnectedNode(Node connectedNode): void
makeServer(**int** port): void
makeClient(String remoteHost,**int** remotePort): void
getClient(): Client
setClient(Client client): void
stringToPacket(String packetString): IpPacket
forwardMessage(String receivedMsg): void
sendPacket(IpPacket packet): void
runAnalyzer(): void
passAttackOutputs(Vector attackOutputs): void
getClients(): Vector
setClients(Vector clients): void
getParentNet():Network
authenticateIp(String sourceIp): Network

OutputAttacks

private:
outputTable: JTable

public:
OutputAttacks(String title, Frame parent)

Path

private:
source: Node
destination: Node
nodes: Vector
type: String
g: Graphics
delta: **int**

public:
Path(Node source, Node destination, String type, Graphics g)
private:
annimate(**float** gradient,**int** sourceX, **int** sourceY, **int** destinationX,
int destinationY,String symbol,Color symbolColor, Font font): void
run(): void

PathTracer

private: source: Node destination: Node sw: Node la: Node start: Node end: Node type: String g: Graphics delta: int maxSymbol: nt count: int symbolCount: int
public: PathTracer(Node from, Node to, Node sw,Node la,String type, Graphics g) PathTracer(Node from, Node to,String type, Graphics g,**int** count,**int** symbolCount): constructor run(): void initialize(Node from, Node to): void **private:** annimate(**float** gradient,**int** sourceX, **int** sourceY, **int** destinationX, **int** destinationY,String symbol,Color symbolColor, Font font): void

SimulatorDialog
private: chSource: Choice chDestination: Choice txtSrcPort: TextField txtDstPort: TextField clientList: List
public: SimulatorDialog(String title, Frame parent) getClientsServer(): **void** actionPerformed(ActionEvent ae): **void** getChSource() : Choice setChSource(Choice chSource): Choice getClientList():List setClientList(List clientList): **void** getChDestination():Choice setChDestination(Choice chDestination): **void** getTxtDstPort(): TextField setTxtDstPort(TextField txtDstPort): TextField getTxtSrcPort(): extField setTxtSrcPort(TextField txtSrcPort): **void**

Switch
pivate: img: Image
public: Switch(String name, Network parent, **int** x, **int** y)

getImg(): Image

ShowToday
public: today():String demo():**void** easyDateFormat (String format): String

Codes used in analysis of attacks

Class : Analysis

```
package analysis;

import java.util.Timer;
import java.util.Date;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.FileInputStream;
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.io.IOException;
import java.io.RandomAccessFile;
import java.util.*;
import java.io.PrintWriter;
import java.io.FileWriter;
import java.io.FileOutputStream;


import packet.*;
import networking.*;
/***
 *
 * this is the class for analysis and detect the intusion type
 */
public class Analysis extends Thread{
      private String output[];
      private final int numAttack=3;
      private Node parent;
      private static int period=10;
      private static String fileName="log.txt";
      private Vector attackOutputs;
      private String numFile="LineNum.txt";
      private static int lineCount;
      private boolean newAttack=false;
      private final String bannedIpFile="Banned.txt";
      public Analysis(Node parent){
            output=new String[numAttack];
```

```
            this.parent=parent;
            attackOutputs=new Vector();
      }
      public void run(){
            while(true){
                  try {
                        Thread.sleep(period*1000);
                        attackOutputs.removeAllElements();
                        packetGeneration(fileName);
                        if(newAttack)
                              parent.passAttackOutputs(attackOutputs);
                  } catch (InterruptedException e) {
                        e.printStackTrace();
                  }
            }
      }
      ////////////////////////// Packet Generation
//////////////////////////
      ////////////////////////////////////////////////////////////////
////////
      public void packetGeneration(String fileName){
            /////variable declaration
            int srcPort,destPort,seq,hlen,type,ver,tlen;
            IpPacket iPacket=new IpPacket();
            Vector v=new Vector();
            BufferedReader bf=null;
            newAttack=false;

            try {
                  BufferedReader numberReader=new BufferedReader(new
InputStreamReader(new FileInputStream(numFile)));
                  try {

      lineCount=Integer.parseInt(numberReader.readLine());
                        numberReader.close();
                  } catch (NumberFormatException e) {
                        e.printStackTrace();
                  } catch (IOException e) {
                        e.printStackTrace();
                  }
                  bf = new BufferedReader(new InputStreamReader(
                  new FileInputStream(fileName)));
            } catch (FileNotFoundException e) {
                  e.printStackTrace();
            }
            FileInputStream reader=null;
            String line=null;
            try {
                  for(int i=0;i<lineCount;i++)
                        line=bf.readLine();
                  line=bf.readLine();
                  v.removeAllElements();
                  while(line!=null){
                        newAttack=true;
```

```
                        lineCount++;
                        String c[]=line.split(" ");
                         ////////////////have to check///////
                         srcPort=Integer.parseInt(c[3]);
                         destPort=Integer.parseInt(c[4]);
                         seq=Integer.parseInt(c[9]);
                         hlen=Integer.parseInt(c[6]);
                   ver=Integer.parseInt(c[5]);
                         tlen=Integer.parseInt(c[7]);
                         ///////////////////////
                        ProtocolPacket pPacket=new
ProtocolPacket(srcPort,destPort);
                        //create protocol(tcp/icmp) packet
                        if(c[8].equalsIgnoreCase("tcp")){
                              TcpPacket tp=new
TcpPacket(srcPort,destPort,seq,hlen,c[10]);
                              iPacket=new
IpPacket(ver,hlen,tlen,c[8],c[1],c[2],tp,c[0]);
                              v.add(iPacket);
                        }
                        else if(c[8].equalsIgnoreCase("icmp")){

                              type=Integer.parseInt(c[11]);
                              IcmpPacket icp=new
IcmpPacket(srcPort,destPort,seq,type);
                              iPacket=new
IpPacket(ver,hlen,tlen,c[8],c[1],c[2],icp,c[0]);
                              v.add(iPacket);
                        }
                        line=bf.readLine();
                  }
            } catch (IOException e) {
                  e.printStackTrace();
            }
            ////////// call defineAttack Funtion,v=vetor
pakets//////////////////////////////////
            if(newAttack)
                  defineAttack(v);
            try {
                  PrintWriter numWriter=new PrintWriter(new
FileOutputStream(numFile));
                  String currentLine=""+lineCount;
                  numWriter.write(currentLine);
                  numWriter.close();
            } catch (FileNotFoundException e) {
                  e.printStackTrace();
            }
      }

      ////////////////////Define
Attack//////////////////////////////////////////////
      ////////////////////////////////////////////////////////////////
////////////
      public void defineAttack(Vector v){
```

```
            IpPacket packet;
            TcpPacket tcp;
            IcmpPacket icmp;
            Vector vicmp=new Vector(); //vector for icmp
            Vector vtcp=new Vector();     //vetor for tcp
            ///////////////
            for(int x=0;x<v.size();x++){
                  packet=(IpPacket)(v.elementAt(x));
                  String protocol=packet.getProtocol();
                  if(protocol.equalsIgnoreCase("icmp")){
                  vicmp.addElement(packet);
                  }     //if icmp
                  else if(protocol.equalsIgnoreCase("tcp")){
                        vtcp.addElement(packet);
                        }           //if tcp
            }//for
            RstAttack rAttack=new RstAttack(null,null,0);
            FinAttack fAttack=new FinAttack(null,null);
            PingAttack pAttack=new PingAttack(null,null);

            if(pAttack.checkAttack(vicmp)>=1){
                  output[0]=pAttack.toString();
                  Vector output=pAttack.getAttackOutputs();
                  for(int i=0;i<output.size();i++){
                        attackOutputs.addElement(output.elementAt(i));
                  }
            }
            if(rAttack.checkAttack(vtcp)==1){
                  output[1]=rAttack.toString();
                  Vector output=rAttack.getAttackOutputs();
                  for(int i=0;i<output.size();i++){
                        attackOutputs.addElement(output.elementAt(i));
                  }
            }
            if(fAttack.checkAttack(vtcp)==1){
                  output[2]=fAttack.toString();
                  Vector output=fAttack.getAttackOutputs();
                  for(int i=0;i<output.size();i++){
                        attackOutputs.addElement(output.elementAt(i));
                  }
            }
            File file=new File(bannedIpFile);
            RandomAccessFile raf=null;
            try {
                  raf = new RandomAccessFile(file,"rw");
                  raf.seek((raf.length()));
                  for(int i=0;i<attackOutputs.size();i++){
                        Output
output=(Output)attackOutputs.elementAt(i);
                        raf.writeBytes(output.getSourceIp()+"\n");

                  }

                  raf.close();
```

```
            } catch (FileNotFoundException e) {
                  e.printStackTrace();
            }
            catch (IOException e) {
                  e.printStackTrace();
            }

      }
      public String[] getOutput() {
            return output;
      }

}
////////////////////////////////////////
```

Class: Attack

```
package analysis;

import java.util.Vector;
public class Attack {
      String sourceIp;
      String destIp;
      protected Vector attackOutputs;
      public Attack(String sourceIp,String destIp){
            this.sourceIp=sourceIp;
            this.destIp=destIp;
            attackOutputs=new Vector();
      }
      public String getDestIp() {
            return destIp;
      }
      public void setDestIp(String destIp) {
            this.destIp = destIp;
      }
      public String getSourceIp() {
            return sourceIp;
      }
      public void setSourceIp(String sourceIp) {
            this.sourceIp = sourceIp;
      }
      public Vector getAttackOutputs() {
            return attackOutputs;
      }

}
////////////////////////////////////////////////
```

Class: FinAttack

```
package analysis;

import java.io.BufferedReader;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
```

```
import java.io.IOException;
import java.io.InputStreamReader;
import java.util.*;

import packet.*;

public class FinAttack extends Attack {

	String finflag,ackFlag,info=" ";

	FinAttack(String sip,String dip){
		super(sip,dip);
	}
	public int checkAttack(Vector v){


		IpPacket packet;
		TcpPacket tPacket;
		reference("reference.txt");
		Vector finAttack=new Vector();
		Vector ackVec=new Vector();

		for(int x=0;x<v.size();x++){
			packet=(IpPacket)(v.elementAt(x));
			tPacket=(TcpPacket)(packet.getProtocolPacket());

	if(tPacket.getFlag().equalsIgnoreCase(this.getFinflag())){
				finAttack.add(packet);
			}

	if(tPacket.getFlag().equalsIgnoreCase(this.getAckFlag())){
				ackVec.add(packet);
			}
		}
		String sip,dip;
		int dport;

		int y=0;
		for(;y<finAttack.size();y++){

			packet=(IpPacket)(finAttack.elementAt(y));
			tPacket=(TcpPacket)(packet.getProtocolPacket());
			sip=packet.getSource();
			dip=packet.getDestination();
			dport=tPacket.getDestination();
			for(int z=0;z<ackVec.size();z++){
				packet=(IpPacket)(ackVec.elementAt(z));

	tPacket=(TcpPacket)(packet.getProtocolPacket());
				if((sip.equalsIgnoreCase(packet.getSource()))

	&&(dip.equalsIgnoreCase(packet.getDestination()))

	&&(dport==tPacket.getDestination())){
```

```
                              finAttack.remove(y);
                              y--;
                        }
                  }

            }
            if(finAttack.size()>0){
                  for(int i=0;i<finAttack.size();i++){
                        packet=(IpPacket)(finAttack.elementAt(i));

      tPacket=(TcpPacket)(packet.getProtocolPacket());

                        info=info+packet.getSource()+"
"+packet.getDestination()+" "+tPacket.getSource()
                        +" "+tPacket.getDestination()+"\n";
                        Output attack=new
Output(packet.getSource(),packet.getDestination(),

      tPacket.getSource(),tPacket.getDestination(),"FIN");
                        attackOutputs.add(attack);
                  }
                  return 1;
            }
            else
                  return 0;
      }
      public void reference(String fileName){
            BufferedReader bf=null;
            try {
                  bf = new BufferedReader(new InputStreamReader(
                              new FileInputStream(fileName)));
            } catch (FileNotFoundException e) {
                  // TODO Auto-generated catch block
                  e.printStackTrace();
            }

            FileInputStream reader=null;
            String line=null;
            String c[]=null;
            try {
                  line=bf.readLine();
                  while(line!=null){

                         c=line.split(" ");

                        if(c[0].equalsIgnoreCase("fin")){
                              this.setFinflag(c[2]);
                              this.setAckFlag(c[5]);


                        }

                        line=bf.readLine();
                  }
```

Class: Output

```
package analysis;

public class Output {
	private String sourceIp;
	private String destinationIp;
	private int sourcePort;
	private int destinationPort;
	private String attackType;
	public Output(String sourceIp, String destinationIp, int
sourcePort, int destinationPort, String attackType) {
		super();
		this.sourceIp = sourceIp;
		this.destinationIp = destinationIp;
		this.sourcePort = sourcePort;
		this.destinationPort = destinationPort;
		this.attackType = attackType;
	}
	public String getAttackType() {
		return attackType;
	}
	public String getDestinationIp() {
		return destinationIp;
	}
	public int getDestinationPort() {
		return destinationPort;
	}
	public String getSourceIp() {
		return sourceIp;
	}
	public int getSourcePort() {
		return sourcePort;
	}
	@Override
	public String toString() {
		// TODO Auto-generated method stub
		return sourceIp+" "+destinationIp+" "+sourcePort+"
"+destinationPort+" "+attackType;
		//return super.toString();
	}

}
////////////////////////////////////
```

Class: PingAttack

```
package analysis;

import java.io.BufferedReader;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.io.InputStreamReader;
```

```
import java.util.*;

import packet.*;

/**
 *
 * class for detect PING Attack
 */
      public class PingAttack extends Attack{

            int limit,time;
            String info=" ";
            public PingAttack(String sourceIp,String destIp){
                  super(sourceIp,destIp);
            }
            ////////////////////////////// detect the attack
//////////////////////////////////////

      ////////////////////////////////////////////////////////////////
//////////////////////
            public int checkAttack(Vector vpacket){
                  this.reference("reference.txt");
                  int second,time=0,dport=0,count=0;;
                  String dip;
                  Vector vping=new Vector();
                  IpPacket packet;
                  IcmpPacket ipak;
                  //Vector temp=new Vector();
                  Vector temp1=new Vector();
                  for(int x=0;x<vpacket.size();x++){
                        packet=(IpPacket)vpacket.elementAt(x);
                        ipak=(IcmpPacket)packet.getProtocolPacket();
                        if(ipak.getType()==8){
                              vping.add(packet);
                              temp1.add(packet);
                        }
                  }
                  //int z=0
                  int test=-1;
                  while(!vping.isEmpty()){
                        packet=(IpPacket)vping.firstElement();
                        ipak=(IcmpPacket)packet.getProtocolPacket();
                        String time1[]=packet.getDateTime().split(":");
                        second=Integer.parseInt(time1[2]);
                        dip=packet.getDestination();
                        dport=ipak.getDestination();
      //                  System.out.println("ref
"+packet.getDateTime()+"    "+dip+"          "+dport);
                        test++;

                        for(int y=0;y<this.getTime();y++){
                              for(int z=0;z<vping.size();z++){

      packet=(IpPacket)vping.elementAt(z);
```

```
      ipak=(IcmpPacket)packet.getProtocolPacket();
                                    String
time2[]=packet.getDateTime().split(":");
                                    time=Integer.parseInt(time2[2]);
                          //        System.out.println("time2
"+packet.getDateTime()+"         "+packet.getDestination()+"
"+ipak.getDestination());
                    //              System.out.println("second
"+second+"time+y "+(time+y));
                                    if(second==((time+y)%60)){

      if((dip.equalsIgnoreCase(packet.getDestination())&&(dport==ipak.g
etDestination()))){
                                                count++;

      packet=(IpPacket)vping.elementAt(z);
                                    //
      System.out.println("ount "+count);
                                                //
      System.out.println(packet.getDateTime());
                                                //info=info+"\nSource
IP: "+packet.getSource();

      info=info+packet.getSource()+" "+packet.getDestination()+"
"+ipak.getSource()
                                                +"
"+ipak.getDestination()+"\n";
                                                Output attack=new
Output(packet.getSource(),packet.getDestination(),

      ipak.getSource(),ipak.getDestination(),"PING");

      attackOutputs.add(attack);
                                                vping.remove(z);
                                                z--;
                                          }
                                    }
                              }
                        }//for
                        if(test==(this.getTime()-1)){

      //System.out.println("dfkjndkfndjkcfndnn"+count);
                                if(count>=this.getLimit()){

      //System.out.println("PINGGGGGGGGGGGGGGGGGGGG        "+count);
                                    return 1;
                              }
                              else
                                    count=0;
                        }
                  }//while
```

```
                  /*int z=0;
                  for(int y=0;y<this.getTime();y++){
                        while(!vping.isEmpty()){
                              packet=(IpPacket)vping.firstElement();

      ipak=(IcmpPacket)packet.getProtocolPacket();
                              String
time1[]=packet.getDateTime().split(":");
                              second=Integer.parseInt(time1[2]);
                              dip=packet.getDestination();
                              dport=ipak.getDestination();
                              System.out.println("ref
"+packet.getDateTime()+"    "+dip+"          "+dport);

                              for(z=0;z<vping.elementNO;z++){

      packet=(IpPacket)vping.elementAt(z);

      ipak=(IcmpPacket)packet.getProtocolPacket();
                                    String
time2[]=packet.getDateTime().split(":");
                                    time=Integer.parseInt(time2[2]);
                                    System.out.println("time2
"+packet.getDateTime()+"         "+packet.getDestination()+"
"+ipak.getDestination());
                                    System.out.println("second
"+second+"time+y "+(time+y));
                              if(second==((time+y)%60)){

      if((dip.equalsIgnoreCase(packet.getDestination())&&(dport==ipak.g
etDestination()))){
                                    count++;

      packet=(IpPacket)vping.elementAt(z);

                                    System.out.println("ount "+count);

      System.out.println(packet.getDateTime());
                                    vping.remove(z);
                                    z--;
                                    }
                              }
                              }
                        }
                  }
*/
                  return count;

            }

//////////////////////////////////////Reference/////////////////////
////////////////////////////////////////////////////////////////
            public void reference(String fileName){
```

```
		BufferedReader bf=null;
		try {
			bf = new BufferedReader(new InputStreamReader(
					new FileInputStream(fileName)));
		} catch (FileNotFoundException e) {
			// TODO Auto-generated catch block
			e.printStackTrace();
		}
		FileInputStream reader=null;
		String line=null;
		String c[]=null;
		try {
			line=bf.readLine();
			while(line!=null){
				//System.out.println(line);
				 c=line.split(" ");
			  if(c[0].equalsIgnoreCase("ping_dos")){
					this.setTime(Integer.parseInt(c[4]));
					this.setLimit(Integer.parseInt(c[5]));
				}
				line=bf.readLine();
			}
		} catch (IOException e) {
			// TODO Auto-generated catch block
			e.printStackTrace();
		}
	}
		public String toString(){
			//String str=     "\nType of Attack: PING DOS
Attack"+"\nSource IP: "+this.getSourceIp()+"\nDestination IP:
"+this.getDestIp();
			return info;
		}

		public int getLimit() {
			return limit;
		}

		public void setLimit(int limit) {
			this.limit = limit;
		}

		public int getTime() {
			return time;
		}

		public void setTime(int time) {
			this.time = time;
		}


	}
```

```
			} catch (IOException e) {
				// TODO Auto-generated catch block
				e.printStackTrace();
			}
		}
		public String getAckFlag() {
			return ackFlag;
		}
		public void setAckFlag(String ackFlag) {
			this.ackFlag = ackFlag;
		}
		public String getFinflag() {
			return finflag;
		}
		public void setFinflag(String finflag) {
			this.finflag = finflag;
		}
		public String toString(){
			return info;
		}

}
```

Class: RstAttack

```
package analysis;

import java.io.BufferedReader;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.io.InputStreamReader;
import java.util.*;

import packet.*;

public class RstAttack extends Attack{

	String rstflag,ackFlag,info="";
	int dport;
	RstAttack(String sip,String dip,int dport){
		super(sip,dip);
		this.dport=dport;
	}

	public int checkAttack(Vector v){
		IpPacket packet;
		TcpPacket tPacket;
```

```
            reference("reference.txt");
            Vector rsAttack=new Vector();
            Vector ackVec=new Vector();

            for(int x=0;x<v.size();x++){
                  packet=(IpPacket)(v.elementAt(x));
                  tPacket=(TcpPacket)(packet.getProtocolPacket());

      if(tPacket.getFlag().equalsIgnoreCase(this.getRstflag())){
                        rsAttack.add(packet);
                  }

      if(tPacket.getFlag().equalsIgnoreCase(this.getAckFlag())){
                        ackVec.add(packet);
                  }
            }

//
      System.out.println("//////////////////////////////////////////////
////////////////////////////////");
//          for(int i=0;i<rsAttack.elementNO;i++){
//                packet=(IpPacket)(rsAttack.elementAt(i));
//                tPacket=(TcpPacket)(packet.getProtocolPacket());
//                System.out.println(" rst   SourceIP:
"+packet.getSource()+"    Destination IP: "+packet.getDestination()
//                            +"    Destination Port:
"+tPacket.getDestination()+"");
//          }
//
//
      System.out.println("//////////////////////////////////////////////
////////////////////////////////");
//
//          for(int i=0;i<ackVec.elementNO;i++){
//                            packet=(IpPacket)(ackVec.elementAt(i));
//
      tPacket=(TcpPacket)(packet.getProtocolPacket());
//                      System.out.println("rstack    SourceIP:
"+packet.getSource()+"    Destination IP: "+packet.getDestination()
//                                        +"    Destination Port:
"+tPacket.getDestination()+""+"");
//                      }
//
//
            String sip,dip;
            int dport;
            int y=0;
            for(;y<rsAttack.size();y++){

      packet=(IpPacket)(rsAttack.elementAt(y));//elementAt(y));
                  tPacket=(TcpPacket)(packet.getProtocolPacket());
                  sip=packet.getSource();
                  dip=packet.getDestination();
                  dport=tPacket.getDestination();
```

```
                  for(int z=0;z<ackVec.size();z++){
                        packet=(IpPacket)(ackVec.elementAt(z));

      tPacket=(TcpPacket)(packet.getProtocolPacket());
//                      System.out.println("   loop SourceIP: "+sip+"
Destination IP: "+dip
//                                    +"    Destination Port:
"+dport+"\n");
//
//                      System.out.println("   matching SourceIP:
"+packet.getSource()+"    Destination IP: "+packet.getDestination()
//                                    +"    Destination Port:
"+tPacket.getDestination()+"\n");
//
                        if((sip.equalsIgnoreCase(packet.getSource()))

      &&(dip.equalsIgnoreCase(packet.getDestination()))

      &&(dport==tPacket.getDestination())){
                              //System.out.println("FVFDVFDVFDVF
"+y);
                              //if(y<0)
                                    //return 0;
//                            System.out.println("   matching SourceIP:
"+packet.getSource()+"    Destination IP: "+packet.getDestination()
//                                          +"    Destination Port:
"+tPacket.getDestination()+"\n");
                              rsAttack.remove(y);
                              y--;
//
                        }
                  }
                  //System.out.println("FVFDVFDVFDVF      "+y);

            }

      /*System.out.println("////////////////////////////////////////
////////////////////////////////////");


            for(int i=0;i<rsAttack.elementNO;i++){
                  packet=(IpPacket)(rsAttack.elementAt(i));
                  tPacket=(TcpPacket)(packet.getProtocolPacket());
                  System.out.println("    SourceIP:
"+packet.getSource()+"    Destination IP: "+packet.getDestination()
                              +"    Destination Port:
"+tPacket.getDestination()+"\n");
            }

      System.out.println("////////////////////////////////////////
////////////////////////////////////");
            for(int i=0;i<ackVec.elementNO;i++){
                  packet=(IpPacket)(ackVec.elementAt(i));
                  tPacket=(TcpPacket)(packet.getProtocolPacket());
```

```
                  System.out.println("    SourceIP:
"+packet.getSource()+"    Destination IP: "+packet.getDestination()
                        +"    Destination Port:
"+tPacket.getDestination()+"\n");
            }

      System.out.println("//////////////////////////////////////////
////////////////////////////////");
            */
            //System.out.println(rsAttack.elementNO+"\n");
            if(rsAttack.size()>0){

                  for(int i=0;i<rsAttack.size();i++){
                        packet=(IpPacket)(rsAttack.elementAt(i));

      tPacket=(TcpPacket)(packet.getProtocolPacket());
                        info=info+packet.getSource()+"
"+packet.getDestination()+" "+tPacket.getSource()
                        +" "+tPacket.getDestination()+"\n";
                        Output attack=new
Output(packet.getSource(),packet.getDestination(),

      tPacket.getSource(),tPacket.getDestination(),"RST");
                        attackOutputs.add(attack);
                  }
                  return 1;
            }
            else
                  return 0;
            /*int f=0,tr=0;
            for(int i=0;i<v.elementNO;i++){

                  rPacket=(IpPacket)(v.elementAt(i));
                  rtPacket=(TcpPacket)(rPacket.getProtocolPacket());


      if(rtPacket.getFlag().equalsIgnoreCase(this.getRstflag())){
                        System.out.println("RST "+this.getRstflag()+"
"+rtPacket.getFlag());
                        tr++;
                        for(int j=0;j<v.elementNO;j++){
                              packet=(IpPacket)(v.elementAt(j));

      tPacket=(TcpPacket)(packet.getProtocolPacket());
                              System.out.println("
"+this.getAckFlag()+" "+tPacket.getFlag());
                              System.out.println("
"+rPacket.getSource()+" "+packet.getSource());
                              //System.out.println("
"+tPacket.getFlag());

      if((tPacket.getFlag().equalsIgnoreCase(this.getAckFlag()))

      &&(rPacket.getSource().equalsIgnoreCase(packet.getSource())
```

```
	&&(tPacket.getDestination()==rtPacket.getDestination())

	&&(rPacket.getDestination().equalsIgnoreCase(packet.getDestinatio
n())))){
					System.out.println("No Attack");
					f++;

				}

				}
			}	//1st if

		}	//for

		if((tr-f)>0){
			return 1;

		}
		else
			return 0;
		//System.out.println("TEST"+tr+" "+f);
*/

	}

	public String toString(){
		return info;
	}
	public void reference(String fileName){
		BufferedReader bf=null;
		try {
			bf = new BufferedReader(new InputStreamReader(
					new FileInputStream(fileName)));
		} catch (FileNotFoundException e) {
			// TODO Auto-generated catch block
			e.printStackTrace();
		}

		FileInputStream reader=null;
		String line=null;
		String c[]=null;
		try {
			line=bf.readLine();
			while(line!=null){
				//System.out.println(line);

				 c=line.split(" ");
	//			System.out.println("-----------");

				if(c[0].equalsIgnoreCase("rst")){
					this.setRstflag(c[2]);
					this.setAckFlag(c[5]);
				}
```

```
                        line=bf.readLine();
                  }
            } catch (IOException e) {
                  // TODO Auto-generated catch block
                  e.printStackTrace();
            }
      }



      public String getAckFlag() {
            return ackFlag;
      }

      public void setAckFlag(String ackFlag) {
            this.ackFlag = ackFlag;
      }

      public String getRstflag() {
            return rstflag;
      }

      public void setRstflag(String rstflag) {
            this.rstflag = rstflag;
      }


}
```

REFERENCE

[1] Stefan Axelsson. Aspects of the Modelling and Performance of Intrusion Detection. Technical Report no. 319L. Department of Computer Engineering, Chalmers University of Technology, SE-412 96 G¨oteborg, Sweden.

[2] Janne Anttila . Intrusion detection in critical e-business environment. Thesis subbmitted in partial fulfillment of the requirements for the degree of Master of Science of Engineering, Espoo 6.3.2004. Helsinki University of Technology, Department of Computer Science and Engineering.

.

[3] Hans Hedbom, Stefan Lindskog, and Erland Jonsson. Risks and Dangers of Security Extensions. Depar tment of Computer Engineering, Chalmers University of Technology, SE-412 96 G¨oteborg, Sweden.

[4] Hans Hedbom. On the Self-Protection of Firewalls and Distributed Intrusion Detection systems. Technical Report 398L. Department of Computer Engineering, Chalmers University of Technology, SE-41296 Göteborg, Sweden.

[5] Ulf Lindqvis, Ulf Gustafson, Erland Jonsson. Analysis of Selected Computer Security Intrusions: In Search of the Vulnerability. Technical Report No. 275. Department of Computer Engineering, Chalmers University of Technology, SE-412 96 Göteborg, Sweden.

[6] Hans Hedbom, Håkan Kvarnstöm and Erland Jonsson. Security Implications of Distributed Intrusion Detection Architectures. In *Proceedings of Fourth Nordic Workshop on Secure IT Systems, NordSec'99*, Stockholm, Sweden, November1–2 1999.

[7] Emilie Lundin Barse and Erland Jonsson, Setting the scene for intrusion detection, Technical report 04-05, August 2004, Department of Computer Engineering, Chalmers University of Technology, G¨oteborg, Sweden.

[8] Behrouz A. Forouzan, Data Communications and Networking, Fourth Edition, pg 724, 727.

[9] http://www.phatak.com/Network-Layer-DoS.php

[10] Raymond R. Panko, Corporate Computer and Network Security, Chapter 5.

[11] Raymond R. Panko, Corporate Computer and Network Security, Chapter 5.

[12] www.wikipedia.org/wiki/ids

[13] Lawrence A. Gordon, Martin P. Loeb, William Lucyshyn and Robert Richardson, 11th Annual CSI/FBI COMPUTER CRIME AND SECURITY SURVEY, 2006

[14] http://en.wikipedia.org/wiki/Tcp/ip_model

[15] http://images.google.com.bd/imgres?imgurl=http://whatis.techtarget.com/digitalguide/images/Misc/fsimage2a.jpg&imgrefurl=http://whatis.techtarget.com/definition/0,,sid9_gci989915,00.html&h=300&w=503&sz=37&hl=en&start=10&um=1&tbnid=FYy7tn30WtCKHM:&tbnh=78&tbnw=130&prev=/images%3Fq%3Dtcp/ip%2Bmodel%26svnum%3D10%26um%3D1%26hl%3Den%26sa%3DG

[16] Understanding Intrusion Detection Through Visualisation by STEFAN AXELSSON, Thesis for the degree of Doctor of Philosophy, Department of Computer Science and Engineering , Chalmers University of Technology, Sweden

[17] A Framework for Effective Alert Visualization by Uday Banerjee Jon Ramsey, SecureWorks 11 Executive Park Dr Atlanta, GA 30329

[18] http://www.necommunications.com/images/diagram_security.jpg

[19] http://student.bu.ac.bd/~shadid/summer06/cse470/index.html

[20] http://www.codeproject.com/library/WinSNMPWrapper/class_diagram.png