# A Survey on Network Attacks and Defence-in-Depth Mechanism by Intrusion Detection System

**A thesis**

**Submitted to the Department of Computer Science & Engineering**

**Of**

**BRAC University**

**By**

**Shakil Akhter Khan**

**ID: 02201004**

**MD. NIZAM UDDIN**

**ID: 02201001**

**In Partial Fulfillment of the**

**Requirements for the Degree**

**Of**

**Bachelor of Computer Science and Engineering**

**December 2006**

**BRAC University, Dhaka, Bangladesh**

# DECLARATION

I, Shakil Akhter Khan, University ID: - 02201004 have completed the thesis on topic, *A Survey on Network Attacks and Defence-in-Depth Mechanism by Intrusion Detection System*, Under CSE400 course regarding the partial fulfillment of my undergraduate degree of Bachelor in Computer Science and Engineering.

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

| | |
|---|---|
| Signature of Supervisor | Signature of Author |

## ACKNOWLEDGEMENT

I am grateful to Allah for giving me the strength and energy to start this project and finally finish it successfully. I am really grateful to my parents who always guide me. Without their guidance it is impossible for me to do anything.

I am really very grateful and take the honor to express my special thanks to my supervisor Risat Mahmud Pathan, M. Sc. for all sorts of supportive suggestions and opinions. Without his support, co-operation and resources it would be a day dream to complete my research in this due time. He is a very hard worker and tries his best to help me to complete my thesis work.

My special thanks go to Mushiqur Rouf Nasa for his useful and important suggestions. I would also like to thank my class mates who gave me feedback whenever I needed. Without their supportive consultancies my research will never have fulfilled the requirements.

I would also like to thank the senior brothers of University and friends who helped me in every possible way.

I want to specially thank my teachers and friends who always supported and helped me during my thesis work and treated me as a family member by always giving me moral support.

Lastly, I give my special thanks to my department for giving me the honor to perform the thesis, a partial fulfillment of the requirement for the Degree of Bachelor of Computer Science and Engineering.

## ABSTRACT

Intrusion Detection System (IDS) is almost new technology in the area of computer and Network security system. It has been experimented all over the world for recent years. In this thesis we have proposed Intrusion detection system by identify different attacks at different layers in the Internet protocol stack. Using this system computer and network system can be made more secure and robust. Our goal in this thesis is to develop an Intrusion Detection system, which will be able help to secure the network, give more information to the administrator to take necessary action when the computer or network system is attacked. Prevention and detection measures help us to stop unauthorized users from accessing any part of computer and network system. The system must be implemented in a way so that the network admin can see and take appropriate action by monitoring all logging information gathered by IDS. Such an implementation can provide computer or network security system. In my thesis, I have done a survey about different attacks that can occur in different layers of the network protocol stack to attack host machines. Some part of these proposed IDS has been implemented by us.

# TABLE OF CONTENTS

# CHAPTER 1

# INTRODUCTION

We use computers for everything from banking and investing to shopping and communicating with others through email or chat programs. Although we may not consider our communications "top secret", we probably do not want strangers reading our email, using our computer to attack other systems, sending forged email from our computer, or examining personal information stored on our computer (such as financial statements).Computer and Network security is the process of preventing and detecting unauthorized use of your computer resources. Prevention and detection measures help us to stop unauthorized users (also known as "intruders") from accessing any part of computer and network system.

Now, hackers are trying to fool the firewall. They are trying to attack different ways that the firewall doesn’t understand. As a result computer and network are not secured anymore. So researcher tries to find out an artificial intelligence based security system. That can monitor the computer/network system and after that take necessary action. In the end of this chapter there is a survey how security hampered in the world.

Network and computer security is an area of ever increasing importance. Computer criminals like hackers, crackers, and fraudsters are trying to attack different systems because we rely on computerized services for various purposes starting from financial transactions to secure communication and storage. Many systems have been developed to deal with the security problem. One such system is Intrusion Detection System (IDS) that works together with other security systems like firewall to provide a defense-in-depth.

## 1.1 Motivation of study

In my network course I found that the network security chapter is an interesting area. So I read more about this particular area. I research and talk with my teachers. After that I found that intrusion detection system is almost new area to research. I talk with my teacher and after that I made up my mind to do this research.

## 1.2 What is Intrusion Detection System?

An intrusion detection system (IDS) inspects all inbound and outbound network activity and identifies suspicious patterns that may indicate a network or system attack from someone attempting to break into or compromise a system.

An Intrusion Detection System (or IDS) generally detects unwanted manipulations to computer systems, mainly through the internet. There are many different types of IDS; some of them are described in next section. The manipulations may take the form of attacks by skilled malicious hackers, or script kiddies using automated tools.

An Intrusion Detection System is used to detect all types of malicious network traffic and computer usage that can't be detected by a conventional firewall. This includes network attacks against vulnerable services, 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).

An IDS is composed of several components: Sensors which generate security events, a Console to monitor events and alerts and control the sensors, and a central Engine that records events logged by the sensors in a database and uses a system of rules to generate alerts from security events received. There are several ways to categorize IDS depending on the type and location of the sensors and the methodology used by the engine to generate alerts. In many simple IDS implementations all three components are combined in a single device or appliance.

## 1.3 Types of Intrusion Detection System

There are many types of IDS. Before going to the types it will better to know where we put the IDS in different network. And how can necessary action taken by IDS when the security of the computer/Network in trouble.

### 1.3.1 Passive system vs. reactive system

In a passive system, the IDS sensor detects a potential security breach, logs the information and signals an alert on the console. In a reactive system, also known as an Intrusion Prevention System (IPS), the IDS responds to the suspicious activity by resetting the connection or by reprogramming the firewall to block network traffic from the suspected malicious source. This can happen automatically or at the command of an operator.

Though they both relate to network security, an IDS differs from a firewall in that a firewall looks outwardly for intrusions in order to stop them from happening. Firewalls limit access between networks to prevent intrusion and do not signal an attack from inside the network. An IDS evaluates a suspected intrusion once it has taken place and signals an alarm. An IDS also watches for attacks that originate from within a system.

This is traditionally achieved by examining network communications, identifying heuristics and patterns (often known as signatures) of common computer attacks, and taking action to alert operators. A system which terminates connections is called an intrusion-prevention system, and is another form of an application layer firewall.

In a network-based system, or NIDS, the sensors are located at choke points in the network to be monitored, often in the DMZ or at network borders. A network intrusion detection system (NIDS) is a system that tries to detect malicious activity such as denial of service attacks, port-scans or even attempts to crack into computers by monitoring network traffic. The NIDS does this by reading all the incoming packets and trying to find suspicious patterns. If, for example, a large number of TCP connection requests to a very large number of different ports are observed, one could assume that there is someone committing a "port scan" at some of the computer(s) in the network. It also (mostly) tries to detect incoming shellcodes in the same manner that an ordinary intrusion detection systems does.

The sensor captures all network traffic flows and analyzes the content of individual packets for malicious traffic. In systems, PIDS (A PIDS will monitor the dynamic behavior and state of the protocol and will typically consists of a system or agent that would typically sit at the front end of a server) and APIDS (monitor the dynamic behavior and state of the protocol and will typically consists of a system or agent that would typically sit between a process, or group of servers, monitoring and analyzing the application protocol between two connected devices) are used to monitor the transport and protocols illegal or inappropriate traffic or constricts of language (say SQL). In a host-based system, the sensor usually consists of a software agent which monitors all activity of the host on which it is installed. Hybrids of these two types of system also exist. There are different types of IDS which is categorized by the researcher.

### 1.3.2 Types of IDS

- A Network Intrusion Detection System is an independent platform which identifies intrusions by examining network traffic and monitors multiple hosts. Network Intrusion Detection Systems gain access to network traffic by connecting to a hub, network switch configured for port mirroring, or network tap. An example of a NIDS is Snort.

- A Protocol-based Intrusion Detection System consists of a system or agent that would typically sit at the front end of a server, monitoring and analyzing the communication protocol between a connected device (a user/PC or system). For a web server this would typically monitor the HTTPS protocol stream and understand the HTTP protocol relative to the web server/system it is trying to protect. Where HTTPS is in use then this system would need to reside in the "shim" or interface between where HTTPS is un-encrypted and immediately prior to it entering the Web presentation layer.

- An Application Protocol-based Intrusion Detection System consists of a system or agent that would typically sit within a group of servers, monitoring and analyzing the communication on application specific protocols. For example; in a web server with database this would monitor the SQL protocol specific to the middleware/business-login as it transacts with the database.

- A Host-based Intrusion Detection System consists of an agent on a host which identifies intrusions by analyzing system calls, application logs, file-system modifications (binaries, password files, capability/acl databases) and other host activities and state.

- A Hybrid Intrusion Detection System combines one or more approaches. Host agent data is combined with network information to form a comprehensive view of the network. An example of a Hybrid IDS is Prelude.

- Misuse detection vs. anomaly detection: in misuse detection, the IDS analyze the information it gathers and compares it to large databases of attack signatures. Essentially, the IDS look for a specific attack that has already been documented. Like a virus detection system, misuse detection software is only as good as the database of attack signatures that it uses to compare packets against. In anomaly detection, the system administrator defines the baseline, or normal, state of the network's traffic load, breakdown, protocol, and typical packet size. The anomaly detector monitors network segments to compare their state to the normal baseline and look for anomalies.
- Network-based vs. host-based systems: in a network-based system, or NIDS, the individual packets flowing through a network are analyzed. The NIDS can detect malicious packets that are designed to be overlooked by a firewall's simplistic filtering rules. In a host-based system, the IDS examines at the activity on each individual computer or host.

Though they both relate to network security, an IDS differs from a firewall in that a firewall looks out for intrusions in order to stop them from happening. The firewall limits the access between networks in order to prevent intrusion and does not signal an attack from inside the network. An IDS evaluates a suspected intrusion once it has taken place and signals an alarm. An IDS also watches for attacks that originate from within a system.

## 1.4 Current Survey

Computer security is an area of ever increasing importance. CSI made surveys from 1997 to 2002, how the computer and network security hampered. It will also include the loss status of the recent years. According there survey it can be easily conclude that the security threat is increasing.

In the following survey, by CSI/FBI which shows different kinds of attacks are increasing day by day.

| **Threat** | **Percent Reporting an Incident 1997** | **Percent Reporting an Incident 2002** | **Average Annual Loss per Firm (x1000) 1997** | **Average Annual Loss per Firm (x1000) 2002** |
|---|---|---|---|---|
| Viruses | 82% | 85% | $76 | $283 |
| Laptop Theft | 58% | 65% | $38 | $89 |
| Denial of Service | 24% | 40% | $77 | $297 |
| System Penetration | 20% | 40% | $132 | $226 |
| Unauthorized Access by Insiders | 40% | 38% | NA | NA |
| Theft of Intellectual Property | 20% | 20% | $954 | $6,571 |
| Financial Fraud | 12% | 12% | $958 | $4,632 |
| Sabotage | 14% | 8% | $164 | $541 |
| Telecom Fraud | 27% | 9% | NA | NA |
| Telecom Eaves-dropping | 11% | 6% | NA | NA |
| Active Wiretap | 3% | 1% | NA | NA |

Figure: CSI/FBI Computer Crime and Security Survey (1997-2002)

# CHAPTER 2
# SURVEY

Sometimes we have a vague idea about IDS. We may think sometimes IDS as a firewall. But firewall and the IDS is not the same thing. Before we talk about the difference between firewall and IDS it is better to know what IDS is and what IDS is not.

## 2.1 Difference between firewall and IDS

Contrary to popular market belief and terminology employed in the literature on intrusion detection systems, not everything falls into this category. In particular, the following security devices are not IDS:

- Network logging systems used, for example, to detect complete vulnerability to any Denial of Service (DoS) attack across a congested network. These are network traffic monitoring systems.
- Vulnerability assessment tools that check for bugs and flaws in operating systems and network services (security scanners), for example Cyber Cop Scanner.
- Anti-virus products designed to detect malicious software such as viruses, Trojan horses, worms, bacteria, logic bombs. Although feature by feature these are very similar to intrusion detection systems and often provide an effective security breach detection tool.

Firewall and IDS both are network security mechanism. Firewall technology emerged in the late 1980s when the Internet was a fairly new technology in terms of its global use and connectivity. The original idea was formed in response to a number of major internet security breaches, which occurred in the late 1980s. It can work in network level, in circuit level or in application level. What firewall does is it block the intruder from accessing the target system. So it is just a Yes/No process. Either a user can pass or it will be blocked. However, IDS is more intelligent than firewall. IDS do not block anything. What it does, it raise an alarm in any intrusion occurs. A user (or intruder) can fell the presence of a firewall which is not always true for IDS.

The firewall is an intrusion blocking system that opens a certain port only and blocks all the others. The IDS is an intrusion detection system that detects all packets on the network in real-time and blocks illegal sessions.

Intrusion detection systems (IDSs) are often mistaken for firewalls when, in many cases, this kind of software or hardware simply monitors and logs suspicious activities. However, in some cases, an IDS provides basic firewall benefits. In the entry-level desktop security realm, the differences between what an IDS does and what a basic firewall does are blurred. Therefore, it's important to recognize the different types of IDSs available so you can compare them effectively to firewalls and, ultimately, your network needs.

Firewalls in general are much like an NIDS or HIDS in that they can either be installed on a desktop, server, or in products like Check Point used as a central firewall for all inbound and outbound network traffic. Firewalls are generally configured to allow certain traffic in and out, and everything else is dropped and preferably logged for later review.

A firewall performs inspection of the packet headers. If a packet type is not allowed (invalid destination port), it is dropped or RST. Non-home use firewalls use a state full inspection, which means that they maintain a table of all of the currently established sessions, so if someone sends an acceptable packet, like SRC port 80, the firewall will still reject it.

IDSs have historically been installed as a "tap" to the circuit, and they passively identify any suspect acceptable traffic (like a buffer overflow targeting your web server) and notify the Admin. Today, some IDSs are recommending that they be installed in series with the firewall so that they can "trap" these allowed but dangerous packets.

The reason that they are complimentary devices is that one protects against invalid connections, while the other protects against hazardous, valid connections. Both can be implemented in the same box if you have low traffic requirements, but many medium and all large businesses find that the performance of their Internet connection suffers if both applications run on the same platform. And many manufacturers require different configurations for their firewalls and IDSs, so they need to have separate boxes.

Firewall cannot detect security breaches associated with traffic that does not pass through it. Only IDS is aware of traffic in the internal network. Not all access to the Internet occurs through the firewall. Firewall does not inspect the content of the permitted traffic. Firewall is more likely to be attacked more often than IDS. Firewall is usually helpless against tunneling attacks. IDS is capable of monitoring messages from other pieces of security infrastructure

## 2.2 Positioning of IDS

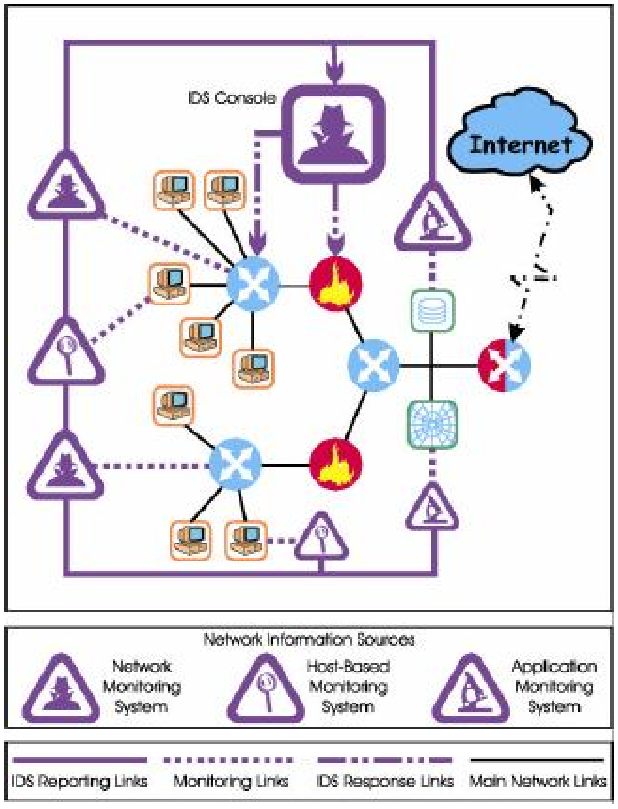
We can position the IDS in different position within the network and computer. We can place the IDS in three different ways.

**Control Strategy**

Control Strategy describes how the elements of an IDS is controlled, and furthermore, how the input and output of the IDS is managed.

Here the IDS is placed before the firewall and at the router. So whenever any packet goes through the network it should be monitor. In according the logging information it will take necessary action or alert the administrator. When any packet wants to go outside from any computer through the network it will also monitor at the point when it reach to router.

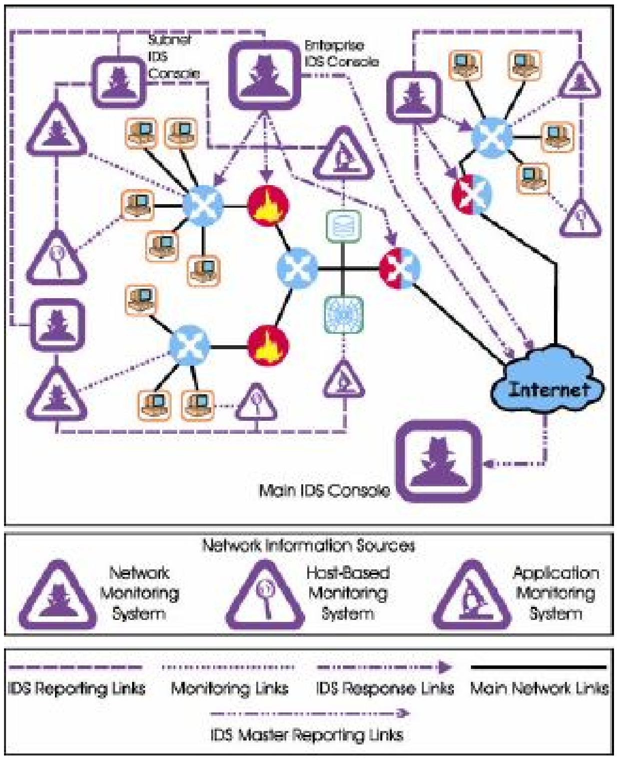
When information goes from the computer, it first checks by the router's IDS. At that time IDS monitor that information.  After some time it will take necessary information if the information malicious. Or it will inform admin to take necessary action.


IDS Console
Internet
Network Information Sources
Network Monitoring System
Host-Based Monitoring System
Application Monitoring System
IDS Reporting Links
Monitoring Links
IDS Response Links
Main Network Links

**Centralized (Figure 1)**
Under centralized control strategies, all monitoring, detection and Reporting is controlled directly from a central location

In partial distribution the IDS is placed before the network and in the server. And every computer has their own IDS. When there is any security problem it will give report to the local IDS which is within the network. It is one of the best ways to place the IDS as the packet is monitored twice. If Central IDS fails to log the information then it is not a problem as the local IDS will find and take the necessary action if any.

This type of IDS is used by universities and research center. Admin for the different computer departments have their own IDS. Only they have to do is to communicate with the admin to inform that which type of logging information should be blocked. It is very helpful for the researcher.



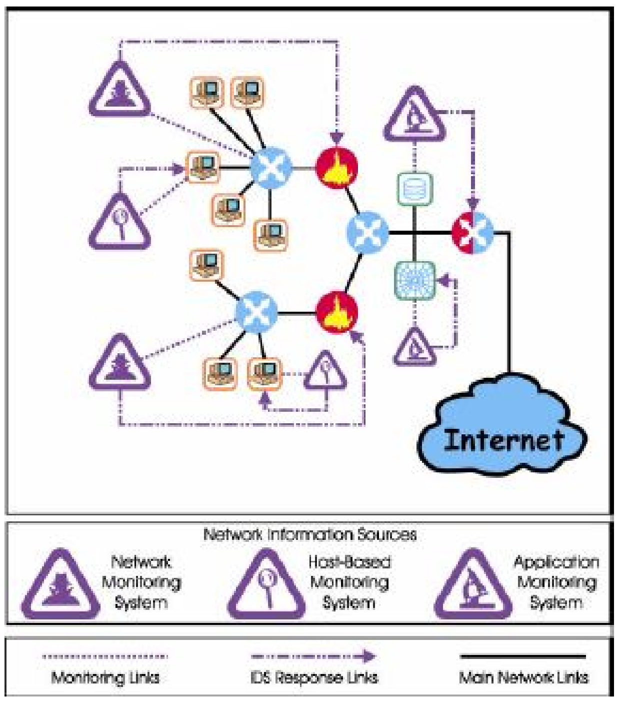
Subnet IDS Console
Enterprise IDS Console
Internet
Main IDS Console
Network Information Sources
Network Monitoring System
Host-Based Monitoring System
Application Monitoring System
IDS Reporting Links
Monitoring Links
IDS Response Links
Main Network Links
IDS Master Reporting Links

**Partially Distributed (Figure 2)**
Monitoring and detection is controlled from a local control node, with hierarchical reporting to one or more central location(s).

In Fully distribution IDS, it is placed every computer, router and gateway. It is expensive and proposed. Sometimes it is not possible to maintain all IDS within the network and computer, IDS collects audit data from multiple hosts and analyzes them centrally to detect intrusions.

It is the best IDS that researcher agree. In every single computer, internet, intranet there placed IDS. So it is called the fully distributed IDS.

Here response decision taken by the individual IDS. Sometimes an artificial Intelligence within the IDS takes necessary action rather then administrator for a single computer or router or computer network.


Internet
Network Information Sources
Network Monitoring System
Host-Based Monitoring System
Application Monitoring System
Monitoring Links
IDS Response Links
Main Network Links

**Fully Distributed (Figure 3)**
Monitoring and detection is done using an agent-based approach, where Response decisions are made at the point of analysis.

## 2.3 Frame work of attack

Computer/network attacks can be defined in different ways. In the following figure the attacks are categorize:

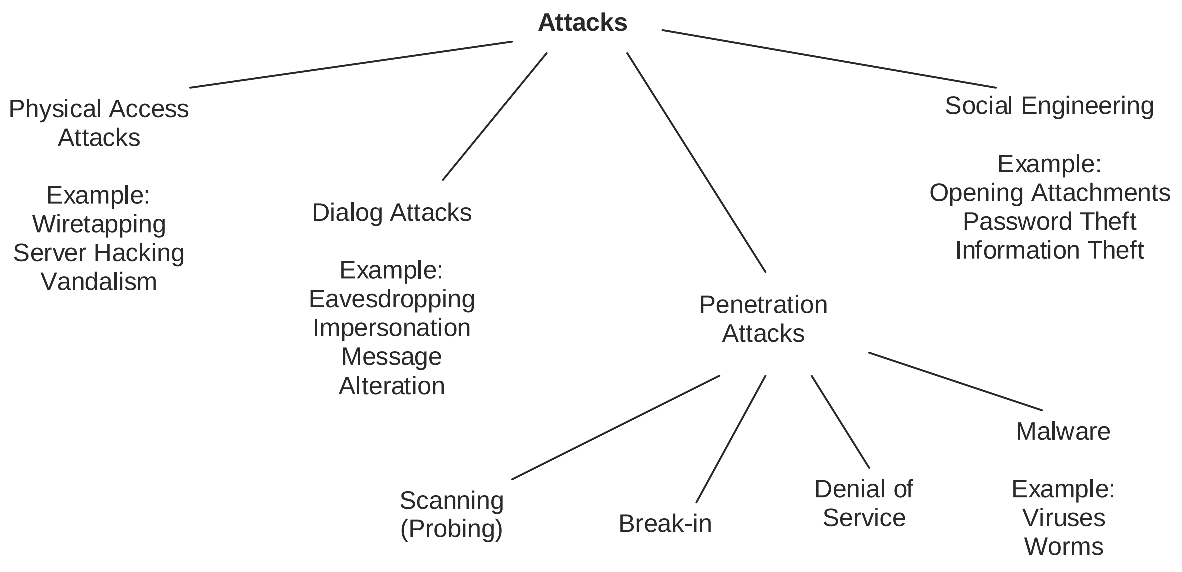


Figure 1-Framework for Attacks

Generally, attacks can be categorized in two areas:

- Passive (aimed at gaining access to penetrate the system without compromising IT resources),
- Active (results in an unauthorized state change of IT resources).

Attacks are also identified by the source category, namely those performed from internal systems (local network), the Internet or from remote dial-in sources). Now, let us see what types of attacks and abuses are detectable (sometimes hardly detectable) by IDS tools to put them in the ad-hoc categorization. The following types of attacks can be identified:

- Those related to unauthorized access to the resources (often as introductory steps toward more sophisticated actions):
    - Password cracking and access violation,
    - Trojan horses,
    - Interceptions; most frequently associated with TCP/IP stealing and interceptions that often employ additional mechanisms to compromise operation of attacked systems (for example by flooding); man in the middle attacks),
    - Spoofing (deliberately misleading by impersonating or masquerading the host identity by placing forged data in the cache of the named server i.e. DNS spoofing)
    - Scanning ports and services, including ICMP scanning (Ping), UDP, TCP Stealth Scanning TCP that takes advantage of a partial TCP connection establishment protocol.) Etc.
    - Remote OS Fingerprinting, for example by testing typical responses on specific packets, addresses of open ports, standard application responses (banner checks), IP stack parameters etc.,
    - Network packet listening (a passive attack that is difficult to detect but sometimes possible),
    - Stealing information, for example disclosure of proprietary information,
    - Authority abuse; a kind of internal attack, for example, suspicious access of authorized users having odd attributes (at unexpected times, coming from unexpected addresses),

    - Unauthorized network connections,
    - Usage of IT resources for private purposes, for example to access pornography sites,
    - Taking advantage of system weaknesses to gain access to resources or privileges,
- Unauthorized alteration of resources (after gaining unauthorized access):
    - Falsification of identity, for example to get system administrator rights,
    - Information altering and deletion,
    - Unauthorized transmission and creation of data (sets), for example arranging a database of stolen credit card numbers on a government computer (e.g. the spectacular theft of several thousand numbers of credit cards in 1999),
    - Unauthorized configuration changes to systems and network services (servers).
- Denial of Service (DoS):
    - Flooding – compromising a system by sending huge amounts of useless information to lock out legitimate traffic and deny services:
        - Ping flood (Smurf) – a large number of ICMP packets sent to a broadcast address,
        - Send mail flood - flooding with hundreds of thousands of messages in a short period of time; also POP and SMTP relaying,
        - SYN flood – initiating huge amounts of TCP requests and not completing handshakes as required by the protocol,
        - Distributed Denial of Service (DDoS); coming from a multiple source,
    - Compromising the systems by taking advantage of their vulnerabilities:
        - Buffer Overflow, for example Ping of Death — sending a very large ICMP (exceeding 64 KB),
        - Remote System Shutdown,
- Web Application attacks; attacks that take advantage of application bugs may cause the same problems as described above.

It is important to remember, that most attacks are not a single action, rather a series of individual events developed in a coordinated manner. Structure and architecture of intrusion detection systems

The main task of intrusion detection systems is defense of a computer system by detecting an attack and possibly repelling it. Detecting hostile attacks depends on the number and type of appropriate actions (Fig.2). Intrusion prevention requires a well-selected combination of "baiting and trapping" aimed at both investigations of threats. Diverting the intruder's attention from protected resources is another task. Both the real system and a possible trap system are constantly monitored. Data generated by intrusion detection systems is carefully examined (this is the main task of each IDS) for detection of possible attacks (intrusions).

Here we can see how IDS work in the computer or network system.

| |
|---|
| Prevention |
| *Simulation* |
| Intrusion Monitoring |
| *Analysis* |
| Intrusion Detection |
| *Notification* |
| Response |

Figure 2: Intrusion detection system activities

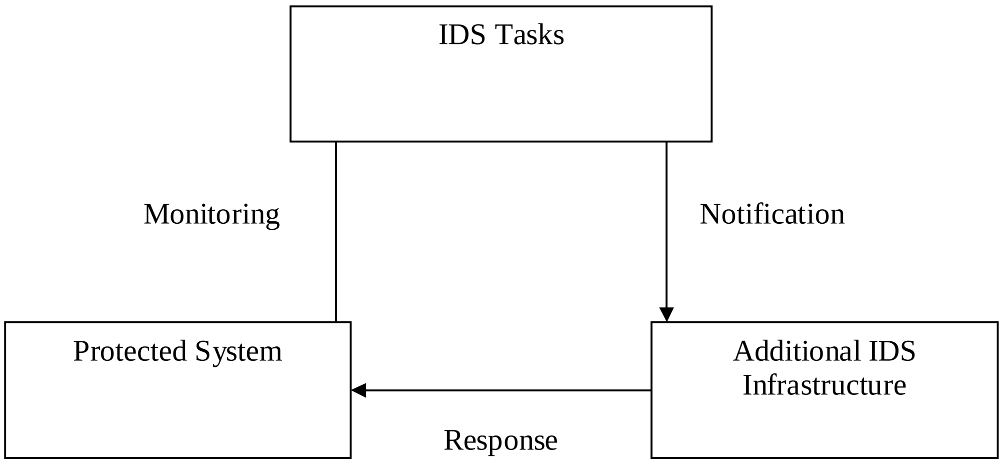


Figure 3: Intrusion detection system infrastructure

An intrusion detection system always has its core element - a sensor (an analysis engine) that is responsible for detecting intrusions. This sensor contains decision-making mechanisms regarding intrusions. Sensors receive raw data from three major information sources (Figure.4): own IDS knowledge base, syslog and audit trails. The syslog may include, for example, configuration of file system, user authorizations etc. This information creates the basis for a further decision-making process.

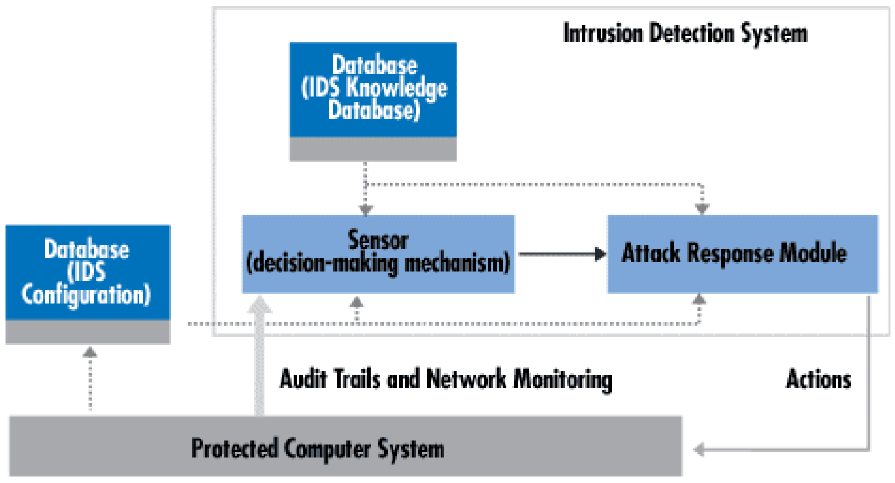

Intrusion Detection System
Database (IDS Knowledge Database)
Database (IDS Configuration)
Sensor (decision-making mechanism)
Attack Response Module
Audit Trails and Network Monitoring
Actions
Protected Computer System

Figure 4: A sample IDS. The arrow width is proportional to the amount of information flowing between system components

The sensor is integrated with the component responsible for data collection (Figure: 5) — an event generator. The collection manner is determined by the event generator policy that defines the filtering mode of event notification information. The event generator (operating system, network, application) produces a policy-consistent set of events that may be a log (or audit) of system events, or network packets. This, set along with the policy information can be stored either in the protected system or outside. In certain cases, no data storage is employed for example, when event data streams are transferred directly to the analyzer. This concerns the network packets in particular.

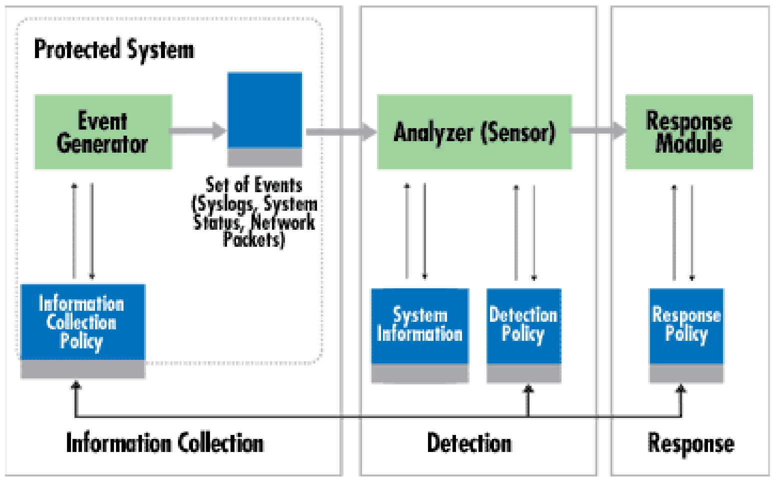


Fig 5: IDS components

The role of the sensor is to filter information and discard any irrelevant data obtained from the event set associated with the protected system, thereby detecting suspicious activities. The analyzer uses the detection policy database for this purpose. The latter comprises the following elements: attack signatures, normal behavior profiles, necessary parameters (for example, thresholds). In addition, the database holds IDS configuration parameters, including modes of communication with the response module. The sensor also has its own database containing the dynamic history of potential complex intrusions (composed from multiple actions).

Intrusion detection systems can be arranged as either centralized (for example, physically integrated within a firewall) or distributed. A distributed IDS consists of multiple Intrusion Detection Systems (IDS) over a large network, all of which communicate with each other. More sophisticated systems follow an agent structure principle where small autonomous modules are organized on a per-host basis across the protected network. The role of the agent is to monitor and filter all activities within the protected area and — depending on the approach adopted — make an initial analysis and even undertake a response action. The cooperative agent network that reports to the central analysis server is one of the most important components of intrusion detection systems. DIDS can employ more sophisticated analysis tools, particularly connected with the detection of distributed attacks. Another separate role of the agent is associated with its mobility and roaming across multiple physical locations. In addition, agents can be specifically devoted to detect certain known attack signatures. This is a decisive factor when introducing protection means associated with new types of attacks. IDS agent-based solutions also use less sophisticated mechanisms for response policy updating.

## 2.4 How Can IDS Protect Against Insider Attacks?

Configuring IDS to detect internal attacks can be difficult. Part of the challenge lies in creating a good rule set for the internal IDS. The reason the rule set needs to be different is due to the fact that different network users require a different amount of access to different services, servers, and systems for their work. The rule set of the internal IDS system should be created so that all the static of employees' day-to-day work activities, such as accessing various services and servers, does not trigger attack warnings, and only the important information is reported. This important information would include detected activities that users do not require for their daily work, as well as any other glaringly obvious attacks such as an nmap, or quest scan.

The logging and reporting of attacks by the internal IDS systems can be used to do much more than detect specific, isolated, and unrelated attacks. By combining the data from all internal IDS systems, system administrators can identify attack trends and patterns. Once attack trends and patterns are identified, the admins will be more able to identify any network users who pose a threat to network security, have been exhibiting any malicious network behavior, or who are doing anything that is against company policy in general. Once these users have been identified, the proper action can be taken to prevent any successful intrusions or the continuance of the activity.

Further, the logs provided by IDS systems can allow the system administrators an audit trail in case there are in fact any successful intrusions. Identified attack trends and patterns can also allow system administrators to see where people are trying to attack against the most. This would allow them to identify any possible security holes, or policy oversights, as well as any servers on the network that have a higher risk of being attacked, and thus allow them to know which systems to keep security tighter on.

A combination of IDS systems should be used to detect insider attacks. The systems that can be deployed to assist in combating against insider attacks include network intrusion detection systems (NIDS), network node intrusion detection systems (NNIDS), host-based intrusion detection systems (HIDS), anomaly-based intrusion detection systems, and the analytical powers of the distributed intrusion detection system (dIDS). These systems each have their uses within the network, along with certain advantages and disadvantages, all of which shall be discussed. The use of network taps to allow some of these systems to operate will also be covered, as well as general security guidelines to follow with regards to deploying the various IDS systems.

## 2.5 Layer wise Attack

Attack can be happened in different layers. On the part of survey we can find out the different attack that can happen in different layer. It is describing here in briefly.

### 2.5.1 Physical Layer

Cable cut

Cable cuts can impact our information infrastructure in a variety of ways. Cable cuts can cause a widespread denial of service to communications customers.

Spectrum

It is the way to change the pattern of the waveform of information. As a result desired data could not be found.

Jamming/fade

There are many ways to jam the signal and data may be destroyed in this way.

### 2.5.2 Data Link Layer

Flooding Attacks

These attacks may be better characterized as brute-force flooding attacks. Sending a steady flood of bogus BPDUs, force continuous spanning tree recalculation, thereby creating a DoS condition on the computational power of the switches. The bigger the attack bandwidth the more chances it stands to succeed.

Topology Engagement Attacks

In this category of attacks, a station being serviced by bridges maliciously claims an active role in the tree topology.

### 2.5.3 Network Layer

Denial of service (Single host or distributed host):

A denial-of-service attack (DoS attack) is an attempt to make a computer resource unavailable to its intended users. Typically the targets are high-profile web servers, and the attack attempts to make the hosted web pages unavailable on the Internet. It is a computer crime that violates the Internet proper use policy as indicated by the Internet Architecture Board (IAB).

DoS attacks have two general forms:

- Force the victim computer(s) to reset or consume its resources such that it can no longer provide its intended service.
- Obstruct the communication media between the intended users and the victim so that they can no longer communicate adequately.

Not all service outages, even those that result from malicious activity, are necessarily denial-of-service attacks. Other types of attack may include a denial of service as a component, but the denial of service may be part of a larger attack. Illegitimate use of resources may also result in denial of service. For example, an intruder may use one's anonymous FTP area as a place to store illegal copies of commercial software, consuming disk space and generating network traffic.

Spoofing:

In the context of network security, a spoofing attack is a situation in which one person or program successfully masquerades as another by falsifying data and thereby gains an illegitimate advantage. An example from cryptography is the man-in-the-middle attack, in which an attacker spoofs Alice into believing he's Bob, and spoofs Bob into believing he's Alice, thus gaining access to all messages in both directions without the trouble of any cryptanalytic effort. The attacker must monitor the packets sent from Alice to Bob and then guess the sequence number of the packets. Then the attacker knocks out Alice with a SYN attack and injects his own packets, claiming to have the address of Alice. Alice's firewall can defend against some spoof attacks when it has been configured with knowledge of all the IP addresses connected to each of its interfaces. It can then detect a spoofed packet if it arrives at an interface that is not known to be connected to the IP address. Another kind of spoofing is "webpage spoofing," also known as phishing. In this attack, a legitimate web page such as a bank's site is reproduced in "look and feel" on another server under control of the attacker. The intent is to fool the users into thinking that they are connected to a trusted site, for instance to harvest user names and passwords. This attack is often performed with the aid of URL spoofing, which exploits web browser bugs in order to display incorrect URLs in the browsers location bar; or with DNS cache poisoning in order to direct the user away from the legitimate site and to the fake one. Once the user puts in their password, the attack-code reports a password error and then redirects the user back to the legitimate site. Some websites, especially pornographic pay sites, allow access to their materials only from certain approved (login-) pages. This is enforced by checking the referring header of the HTTP request. This referrer header however can be changed (known as "Referrer spoofing" or "Ref-tar spoofing"), allowing users to gain unauthorized access

Smurfing:

Smurfing, in relation to computer networks, is a type of denial of service attack that causes a network to become flooded with responses to a fake ICMP “echo request” packets. One typical method would be for an attacker to ping the broadcast address of a network with a spoofed source address. All hosts on the network will respond to the ICMP echo request, but they will respond to the spoofed address. In this case the target would be the host with the spoofed address.

Sequence number guessing (A part of spoofing):

In this way attacker at first find the port no that is using to interconnect to the client to server. Then it will try to find those port which is opened to communicate with server but is not used. When the attacker gets it, they attack those computer/network.

### 2.5.4 TCP layer

ACK Denial-of-Service Attack:

A denial of service attack that sends a large number of TCP packets with the ACK flag set to a target. The goal of the attack is to use all of the target system's network resources causing the target's performance to degrade and possible even cause a system crash. This attack can also be very effective against stateful firewalls & IDS/IPS product if they have a low amount of memory dedicated to state table management.

Sniffing:

Packet sniffing is a technique of monitoring network traffic. It is effective on both switched and non-switched networks. In a non-switched network environment packet sniffing is an easy thing to do. This is because network traffic is sent to a hub which broadcasts it to everyone. Switched networks are completely different in the way they operate. Switches work by sending traffic to the destination host only. This happens because switches have CAM tables. These tables store information like MAC addresses, switch ports, and VLAN information. Before sending traffic from one host to another on the same local area network, the host ARP cache is first checked. The ARP cache is a table that stores both Layer 2 (MAC) addresses and Layer 3 (IP) addresses of hosts on the local network. If the destination host isn't in the ARP cache, the source host sends a broadcast ARP request looking for the host. When the host replies, the traffic can be sent to it. The traffic goes from the source host to the switch, and then directly to the destination host. This description shows that traffic isn't broadcast out to every host, but only to the destination host, therefore it's harder to sniff traffic.

SYN Attack:

A SYN flood sends a flood of TCP/SYN packets, often with a forged sender address. Each of these packets are handled like a connection request, causing the server to spawn a half-open connection, by sending back a TCP/SYN-ACK packet, and waiting for an TCP/ACK packet in response from the sender address. However, because the sender address is forged, the response never comes. These half-open connections consume resources on the server and limit the number of connections the server is able to make, reducing the server's ability to respond to legitimate requests until after the attack ends.

When a computer wants to make a TCP/IP connection (the most common internet connection) to another computer, usually a server, an exchange of TCP/SYN and TCP/ACK packets of information occur. The computer, requesting the connection, usually the client's or user's computer, sends a TCP/SYN packet which asks the server if it can connect. If the server will allow connections, it sends a TCP/SYN-ACK packet back to the client to say "Yes, you may connect" and reserves a space for the connection, waiting for the client to respond with a TCP/ACK packet detailing the specifics of its connection.

In a SYN flood the address of the client is often forged so that when the server sends the go-ahead back to the client, the message is never received because the client either doesn't exist or wasn't expecting the packet and subsequently ignores it. This leaves the server with a dead connection, reserved for a client that will never respond. Usually this is done to one server (targeted server or victim server) many times in order to reserve all the connections for unresolved clients, which keeps legitimate clients from making connections.

Tear Drop Attack:

A teardrop exploits the vulnerability of the TCP/IP protocol at the time of reassembling the fragmented IP packets at the receiving end. In section 4 we described a field in the IP packet called Fragment Offset which is used to identify where each of the fragments belong at the time of reassembly. A teardrop attack is carried by manipulating the fragment offset field to overlap. This confuses the server at the time of reassembly causing it to crash.

Session Hijacking Attack:

TCP session hijacking is when a hacker takes over a TCP session between two machines. Since most authentication only occurs at the start of a TCP session, this allows the hacker to gain access to a machine. A popular method is using source-routed IP packets. This allows a hacker at point A on the network to participate in a conversation between B and C by encouraging the IP packets to pass through its machine. If source-routing is turned off, the hacker can use "blind" hijacking, whereby it guesses the responses of the two machines. Thus, the hacker can send a command, but can never see the response. However, a common command would be to set a password allowing access from somewhere else on the net. A hacker can also be "inline" between B and C using a sniffing program to watch the conversation. This is known as a "man-in-the-middle attack". A common component of such an attack is to execute a denial-of-service (DoS) attack against one end-point to stop it from responding. This attack can be either against the machine to force it to crash, or against the network connection to force heavy packet loss.

Port Scan:

A cracker can use scanning software to determine which hosts are active and which are down to avoid wasting time on inactive hosts. A port scan can gather data about a single host or hosts within a subnet (256 adjacent network addresses). A scan can be implemented using the Ping utility. After determining which hosts and associated ports are active, the cracker will initiate different types of probes on the active ports.

Examples of probes are as follows:

. Gathering information from the Domain Name System (DNS)
. Determining the available network services, such as e-mail, FTP, and remote logon
. Determining the type and release of the operating system

Reset (RST) Attack:

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.

FIN Attack:

A FIN attack is similar to the RST attack and 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.

### 2.5.5 Application Layer

Vulnerable CGI programs:

Most web servers support Common Gateway Interface (CGI) programs to provide interactivity in web pages enabling functions such as data collection and verification. Default installation of most web servers come with sample CGI scripts - in many cases these sample scripts and frequently used scripts contain vulnerabilities that allow a user to execute code on the local system.

Spam (for Email):

Spamming is the abuse of electronic messaging systems to send unsolicited, undesired bulk messages, while the most widely recognized form of spam is e-mail spam, the term is applied to similar abuses in other media such as instant messaging spam, Usenet newsgroup spam, web search engine spam, spam in blogs, and mobile phone messaging spam.

Nimda worm & Mutations:

The name of this virus came from the reversed spelling of "admin". The worm sends itself out by email, searches for open network shares, attempts to copy itself to unpatched or already compromised Microsoft IIS web servers, and is a virus infecting both local files and files on remote network shares.

Malicious URLs:

A web site may inadvertently include malicious HTML tags or script in a dynamically generated page based on unvalidated input from untrustworthy sources. This can be a problem when a web server does not adequately ensure that generated pages are properly encoded to prevent unintended execution of scripts, and when input is not validated to prevent malicious HTML from being presented to the user.

Most web browsers have the capability to interpret scripts embedded in web pages downloaded from a web server. Such scripts may be written in a variety of scripting languages and are run by the client's browser. Most browsers are installed with the capability to run scripts enabled by default. Malicious code provided by one client for another client. Sites that host discussion groups with web interfaces have long guarded against vulnerability where one client embeds malicious HTML tags in a message intended for another client. For example, an attacker might post a message like ‘Hello message board’. This is a message.
<SCRIPT>malicious code</SCRIPT>
This is the end of my message.

When a victim with scripts enabled in their browser reads this message, the malicious code may be executed unexpectedly. Scripting tags that can be embedded in this way include <SCRIPT>, <OBJECT>, <APPLET>, and <EMBED>.
When client-to-client communications are mediated by a server, site developers explicitly recognize that data input is untrustworthy when it is presented to other users. Most discussion group servers either will not accept such input or will encode/filter it before sending anything to other readers.

However, this situation may occur when the client relies on an untrustworthy source of information when submitting a request.

Spyware and Ad ware Attacks:
Spyware is computer software that collects personal information about users without their informed consent. Personal information is secretly recorded with a variety of techniques, including logging keystrokes, recording Internet web browsing history, and scanning documents on the computer's hard disk. Purposes range from overtly criminal (theft of passwords and financial details) to the merely annoying (recording Internet search history for targeted advertising, while consuming computer resources). Spyware may collect different types of information. Some variants attempt to track the websites a user visits and then send this information to an advertising agency. More malicious

variants attempt to intercept passwords or credit card numbers as a user enters them into a web form or other application.

Back door:

A back door attack takes place using dial-up modems or asynchronous external connections. The strategy is to gain access to a network through bypassing of control mechanisms by getting in through a back door such as a modem.

Trojan horses:

A Trojan horse is a program that hides in a useful program and usually has a malicious function. A major difference between viruses and Trojan horses is that Trojan horses do not self-replicate. In addition to launching attacks on a system, a Trojan horse can establish a back door that can be exploited by attackers. For example, a Trojan horse can be programmed to open a high-numbered port, which could be scanned and make the system vulnerable to attackers.

FTP Bounce Attack:

In the field of computer networking and security, the **FTP** bounce attack is an exploit of the FTP protocol whereby an attacker is able to use the **PORT** command to request access to ports indirectly through the use of the victim machine as a middle man for the request. This technique can be used to port scan hosts discreetly, and to access specific ports that the attacker cannot access through a direct connection.

## 2.6 Some IDS: (operation/experiment)

Now, we already understand that there are many IDS for different systems. On the basis of our work we found some IDS (Snort, Deep Sight Analyzer, and IDS Policy Manager) that will help us to guide our own IDS structure. These IDS are discussed briefly here.

### 2.6.1 Snort

This lightweight network intrusion detection and prevention system excels at traffic analysis and packet logging on IP networks. Through protocol analysis, content searching, and various pre-processors, Snort detects thousands of worms, vulnerability exploit attempts, port scans, and other suspicious behavior. Snort uses a flexible rule-based language to describe traffic that it should collect or pass, and a modular detection engine. Also check out the free Basic Analysis and Security Engine (BASE), a web interface for analyzing Snort alerts. Snort handles IP de-fragmentation. Snort perform TCP stream reassembly. Snort perform stateful protocol analysis.

#### 2.6.2 Deep Sight Analyzer

A secure and personalized incident management system, gives IT professionals the ability to track and manage incidents and attacks on their own networks. DeepSight Analyzer automatically correlates attacks from a multitude of IDS and Firewall products, giving IT professionals a comprehensive view of their environment. DeepSight Analyzer manages threats by comparing incidents on their network against the world's largest vulnerability database, tracking attacks to resolution, and generating statistical incident reports. Users anonymously and automatically submit suspicious network traffic and intrusion attempts to the Symantec Event Database. This information is used to identify patterns in attacks that help serve as a threat-gauging system for the Internet community. In return, participants receive access to DeepSight Analyzer, a secure, personalized, Web-based incident console, which provides local incident tracking, personalized incident reports, and the ability to generate attacker notification messages.

### 2.6.3 IDS Policy Manager

IDS Policy Manager was designed to manage Snort IDS sensors in a distributed environment. This is done by having the ability to take the text. Configuration and rule files allow modifying them with an easy-to-use graphical interface. With the added ability to merge new rule sets, managing preprocessors, control output modules and SCP rules to sensors, this tool makes managing snort easy for most security professionals. It also allows merging new snort rules into existing rule files, making quick changes to snort rules, updating rules via the web, manage multiple sensors with multiple policy files, uploading policies files via SFTP, FTP, and File-Copy, restart sensors after uploading.

### 2.6.4 Analysis Console for Intrusion Databases (ACID)

The Analysis Console for Intrusion Databases (ACID) is a PHP-based analysis engine to search and process a database of security events generated by various IDSes, firewalls, and network monitoring tools. The features currently include:

- Query-builder and search interface for finding alerts matching on alert meta information (e.g. signature, detection time) as well as the underlying network evidence (e.g. source/destination address, ports, payload, or flags).
- Packet viewer (decoder) will graphically display the layer-3 and layer-4 packet information of logged alerts
- Alert management by providing constructs to logically group alerts to create incidents (alert groups), deleting the handled alerts or false positives, exporting to email for collaboration, or archiving of alerts to transfer them between alert databases.
- Chart and statistics generation based on time, sensor, signature, protocol, IP address, TCP/UDP ports, or classification

ACID has the ability to analyze a wide variety of events which are post-processed into its database.

### 2.6.5 OSSEC HIDS

OSSEC HIDS performs log analysis, integrity checking, root kit detection, time-based alerting and active response. In addition to its IDS functionality, it is commonly used as a SEM/SIM solution. Because of its powerful log analysis engine, ISPs, universities and data centers are running OSSEC HIDS to monitor and analyze their firewalls, IDSs, web servers and authentication logs.

# CHAPTER 3

# SYSTEM DEVELOPMENT

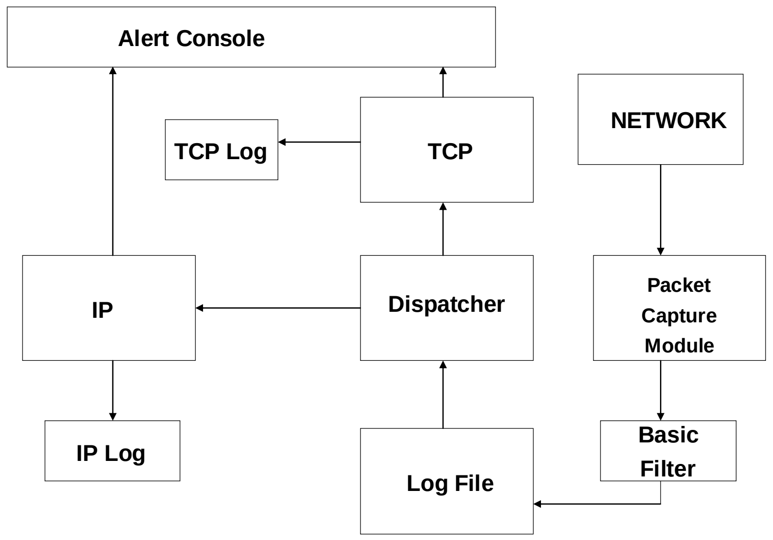
## 3.1 Project initialization and planning

The work done by us was a previously worked out research project and we have continued the project further more. The previously completed tasks were implementing an standard IDS system. This IDS is proposed to work with single host technology. Our major project requirement was to make sure that the logging information observe and take necessary action by our IDS. For this reason, we needed to study network structure. To do that, we studied a lot. We took help from Internet and also from many books.

## 3.2 Architecture

The structure of our IDS showing below:

Figure: Architecture of the proposed IDS



Here is the description of each part of the architecture:

**Alert Console:** The interface between an IDS and computer system. The IDS generate alert through Alert Console.
**Network:** The source of all information both raw data and attack.

Packet capture module: This module captures packets from network.

**Basic Filter:** The filter which discard unnecessary information.

**Log file:** File which contains the logging information.

**Dispatcher:** Data provider to TCP defense and IP defense.

**TCP defense:** Defense system which detect attacks on TCP layer.

**IP defense:** Defense system which detect attacks on IP layer.

**TCP log:** Log file which contains attack record on TCP layer.

**IP log:** Log file which contains attack record on IP layer.

**Working procedure:** The Packet capture module captures data from the network. The data is stored in a log file after being filtered by a basic filter which discards the unnecessary information. Now a dispatcher provides this information to a TCP defense system and an IP defense system. The TCP defense detects and generates alarm if any attack occurs on TCP layer. On the other hand, the IP defense detects and generates alarm if any attack occurs on IP layer. Both defense systems log the attack information in its log file.

## 3.3 Building the Prototype

To do the prototype successfully, we needed to be familiar with the Linux (Red Hat Linux Fedora Core 4).We are using C language to build our prototype. We also need packet capture library to capture the network packets. After getting acquainted with the environment we started the software module that actually runs in Linux operating system environment. This part was really tricky for us to implement as the algorithm we intended to use wasn't implemented earlier. So we maintained a constant touch with our supervisor who helped us in many ways.

## 3.4 Defense Mechanism

We consider few attacks on network layer and TCP layer. They are described bellow:

**<u>Network Layer:</u>**

Ping DoS attack:

Data to log:

1. source IP address,
2. Type field (if type is 8 then ping).

Defense mechanism:

The log data need to be stored in a file. After a certain period the file will be checked. If the number of ping crosses a limit then IDS will generate alarm to block all sources IPs. This mechanism prevents both single and distributed DOS attack.

Spoofing:

Data to log:

Source IP address

Defense mechanism:

In the case of spoofing the main problem is to find that either it is an attack or not. What can IDS do it can send a return request to the source that either it send the packet. If the answer is not yes then IDS will generate alarm. Now the source IP can not be blocked because it does not do anything wrong. What the system can do, it can shout down its web service for some moment. But it always depends on the situation. Another approach can be: two trusted host can maintain common signature so that some intruder can not spoof.

Timestamp DoS attack:

Data to log:

3. source IP address,
4. Type field (if type is 8 then ping).

Defense mechanism:

We need to store the log data in a file. After a certain period the file will be checked. If the number of Timestamp crosses a limit then IDS will generate alarm to block all sources IPs. This mechanism prevents both single and distributed DOS attack.

Smurfing:

Defense mechanism:

The IDS must run on the main router of the LAN. For any ICPM ping packet, if the destination address is a broadcast address then the packet needs to be drooped.

Ping of death attack:

Data to log:

1. Source IP address,
2. Type field (if type is 8 then ping).
3. Size field

Defense mechanism:

If the type of the packet is ping, then we need to check the size of the ICMP packet. If the size cross maximum limit then the packet need to be drooped. So the IDS will generate alarm.

**TCP layer:**

SYN flooding attack:

Data to log:

1. source IP address,
2. A group of flags which indicate the number of SYN.

Defense mechanism:

A flag will be maintained for each source IP. The value of each flag will increase by one for SYN from corresponding source IP. If the value of any flag cross a limit (may be 20 or 30) within a certain time, then IDS will generate alarm.

ACK DoS attack:

Data to log:

3. source IP address,
4. A group of flags which indicate the number of SYN-ACK.

Defense mechanism:

A flag will be maintained for each source IP. The value of each flag will increase by one for SYN-ACK from corresponding source IP. If the value of any flag cross a limit (may be 20 or 30) within a certain time, then IDS will generate alarm.

RST attacks:

Data to log:

Source IP address

Defense mechanism:

A table will be maintained which contain all source IP. It means that this IPs has a TCP connection with this destination. Now if any RST comes from any source then the table will be checked. If the source IP is not in the table then it is an attack and IDS comes to an action.

FIN attacks:

Data to log:

Source IP address

Defense mechanism:

A table will be maintained which contain all source IP. It means that this IPs has a TCP connection with this destination. Now if any FIN comes from any source then the table will be checked. If the source IP is not in the table then it is an attack and IDS comes to an action.

# CHAPTER 4
# CONCLUSION

We know we cannot secure the network or computer system hundred percent. Al we have to do is level up the security. As our topics is security defense-in-depth. IDS works in depth. In our system we found out some major flaws. As our proposed system is not easy to implement but it can logged some major attacks from computer networks. When we implement the full design it is difficult to track which information should logged for different network packet. But we can successfully manage it. Analysis of logged data is another different area. Researchers are still researching. Scalable IDS is also proposed. With our propped IDS it will help the security system more secured. Network/computer security can not be attack by intruders easily.

# APPENDIX

# Appendix A

**Defence Mechanism Code for FIN attack:**

```
#include <pcap.h>
#include <stdio.h>
#include <stdlib.h>
#include <errno.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netinet/ip.h>
#include <netinet/ip_icmp.h>
#include <netinet/tcp.h>
#include <netinet/udp.h>
#include <arpa/inet.h>
#include <netinet/if_ether.h>
#include <net/ethernet.h>
#include <netinet/ether.h>
#include <sys/types.h>
#include <time.h>

                        int numberOfTCP,pingFlag,handle_icmpFlag,type;
                        char *dev;  //device name
                        char a[1000][20],inp[20];
                        time_t tval1,tval2;
                        struct tm *now1,*now2;
                        int min1,min2,ti,size;
//sizes
                        int size_ethernet = sizeof(struct ether_header);
                        int size_ip = sizeof(struct ip);
                        int size_tcp = sizeof(struct tcphdr);
                        int size_icmp=sizeof(struct icmphdr);
                        int size_udp=sizeof(struct udphdr);
int setVal(char inp[])
{
                                        int i;
                                        for(i=0;i<size;++i)
                                        {
                                         if(!strcmp(a[i],inp))
                                         {
                                            return 1;
                                         }
                                        }
                                        return 0;
}
/* function to print payload data */


void                                    handle_Tcp(const u_char * packet)
{
     struct tcphdr * tcp;
const char *payload;
tcp=(struct tcphdr*)(packet+size_ethernet);
int ff;
if(tcp->fin)
{
   ff=setVal(inp);
   if(ff==0)
                                        printf("\n........... Attack.............\n");
}
}
void                                    handle_Udp(const u_char * packet)
{
}
void                                    handle_Icmp(const u_char * packet)
{
handle_icmpFlag=1;
```

```
struct icmphdr * icmphdr;
const char *payload;
icmphdr=(struct icmphdr*)(packet+size_ethernet);
type=icmphdr->type;
if(icmphdr->type==69)
{
                                        ++numberOfICMP;
                                        pingFlag=1;
}
}

void handleIP( const u_char * packet)
{
    int ff;
                                        struct ip *ip;
                                        ip = (struct ip*)(packet + size_ethernet);
                                        strcpy(inp,inet_ntoa(ip->ip_src));
                                        ff=setVal(inp);
                                        if (ip->ip_p == IPPROTO_TCP)
                                        {
                                            handle_Tcp(packet);
                                        }
                                        else
                                        if (ip->ip_p == IPPROTO_UDP)
                                        {
                                            handle_Udp(packet);
                                        }
                                        else
                                        if (ip->ip_p == IPPROTO_ICMP)
                                        {
                                            //++numberOfICMP;
                                            handle_Icmp(packet);
                                        }
                                        else
                                        if (ip->ip_p == IPPROTO_IP)
                                        {
                                           printf("\n\tProtocol: IP\n");
                                        }
}

void Process_Packet(u_char * a, const struct pcap_pkthdr *pk_header, const u_char *
packet)
                                         {
                                           static int Count=1;
                                           struct ether_header *ethHeader;
                                           ethHeader = (struct ether_header *) packet;
                                           if(ntohs (ethHeader-
>ether_type)==ETHERTYPE_IP)
                                           {
                                           printf("IP");    /*IP*/
                                           handleIP(packet);
                                           }

                                          }

int main(int argc,char **argv)
{
                                        //freopen("a.txt","w",stdout);
                                        tval1 = time(NULL);
                                        numberOfICMP=0;
                                        pingFlag=0;
                                        handle_icmpFlag=0;
                                        /*
                                        Declarations
                                        */
                                        char errbuf[PCAP_ERRBUF_SIZE];  //256 defined in
pcap.h
                                        pcap_t* pd;
```

```
                                int snaplen=200;        //len of packet to capture
                                const u_char *packet;//packet
                                bpf_u_int32 maskp;        //mask
                                bpf_u_int32 netp;       //net address
                                char
localnet[INET_ADDRSTRLEN],umask[INET_ADDRSTRLEN];
                                struct pcap_pkthdr hdr;
                                u_char* args = NULL;          // dont know why??????

                                if(dev==NULL)           //ethernet or wlan card   get the
device name to dev
                                {
                                 if((dev=pcap_lookupdev(errbuf))==NULL)
                                 {
                                    perror("Device Lookup :");
                                 }
                                }

                                if(dev == NULL)
                                {
                                 printf("%s\n",errbuf); exit(1);
                                }
                                else
                                {
                                 printf("\tThe Device Found is :%s\n",dev);            // the
device found is
                                }

                                if((pd=pcap_open_live(dev,snaplen,0,500,errbuf))==NU
LL)  //200bytes of capture ,0 promiscous,500 time out pd ??????
                                 perror("Error Open live ");

                                if(pcap_lookupnet(dev,&netp,&maskp,errbuf)<0)  //net
is network address , maskp is the umask value
```

```
                                    perror("Error lookup:");

                                    printf("\tlocal net %s and umask %s
\n",inet_ntop(AF_INET,&netp,localnet,sizeof(localnet)),inet_ntop
                                    (AF_INET,&maskp,umask,sizeof(umask)));



                                    if(packet == NULL)
                                    {
                                    printf("Didn't grab packet\n");
                                    exit(1);
                                    }

                                    now1 = localtime(&tval1);
                                    min1=now1->tm_min;
                                    /* loop function*/
                                    pcap_loop(pd,-1,Process_Packet,args);
                                    printf("\n\t-----------------------------------------------------------
---------------\n");

}
```

**Defence Mechanism Code for ping DoS attack:**

```
#include <pcap.h>
#include <stdio.h>
#include <stdlib.h>
#include <errno.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netinet/ip.h>
#include <netinet/ip_icmp.h>
#include <netinet/tcp.h>
#include <netinet/udp.h>
```

```
#include <arpa/inet.h>
#include <netinet/if_ether.h>
#include <net/ethernet.h>
#include <netinet/ether.h>
#include <sys/types.h>
#include <time.h>

                int numberOfICMP,pingFlag,handle_icmpFlag,type;
                char *dev;  //device name
                time_t tval1,tval2;
                struct tm *now1,*now2;
                int min1,min2,ti;
//sizes
                int size_ethernet = sizeof(struct ether_header);
                int size_ip = sizeof(struct ip);
                int size_tcp = sizeof(struct tcphdr);
                int size_icmp=sizeof(struct icmphdr);
                int size_udp=sizeof(struct udphdr);

/* function to print payload data */


void            handle_Tcp(const u_char * packet)
{
}
void            handle_Udp(const u_char * packet)
{
}
void            handle_Icmp(const u_char * packet)
{
handle_icmpFlag=1;
struct icmphdr * icmphdr;
const char *payload;
icmphdr=(struct icmphdr*)(packet+size_ethernet);
type=icmphdr->type;
if(icmphdr->type==69)
{
                                        ++numberOfICMP;
                                        pingFlag=1;
}
}

void handleIP( const u_char * packet)
{
                                        struct ip *ip;
                                        ip = (struct ip*)(packet + size_ethernet);
                                        if (ip->ip_p == IPPROTO_TCP)
                                        {
                                           handle_Tcp(packet);
                                        }
                                        else
                                        if (ip->ip_p == IPPROTO_UDP)
                                        {
                                           handle_Udp(packet);
                                        }
                                        else
                                        if (ip->ip_p == IPPROTO_ICMP)
                                        {
                                           //++numberOfICMP;
                                           handle_Icmp(packet);
                                        }
                                        else
                                        if (ip->ip_p == IPPROTO_IP)
                                        {
                                           printf("\n\tProtocol: IP\n");
                                        }
}
```

```
void Process_Packet(u_char * a, const struct pcap_pkthdr *pk_header, const u_char *
packet)
                                    {
                                      static int Count=1;
                                      struct ether_header *ethHeader;
                                      ethHeader = (struct ether_header *) packet;
                                      if(ntohs (ethHeader-
>ether_type)==ETHERTYPE_IP)
                                      {
                                      printf("IP");    /*IP*/
                                      handleIP(packet);
                                      }
                                      if(numberOfICMP>0)
                                      {
                                         //(void) time(&t2);
                                         //if(t2)
                                         tval2 = time(NULL);
                                         now2 = localtime(&tval2);
                                         min2=now2->tm_min;
                                         printf("min1=%d\n",min1);
                                         printf("min2=%d\n",min2);
                                         if(min2<min1)
                                         {
                                            ti=60-min1;   //time before 60
                                            ti=ti+min2; //time in new hour
                                            if(ti>=1)
                                            {
                                               if(numberOfICMP>=5)
                                               {

                                                  printf("\nAttack.............\n");
                                                  exit(1);
                                                  numberOfICMP=0;
                                               }
```

```
                                        min1=min2;
                                    }
                                }
                                else
                                {
                                    ti=min2-min1;
                                    if(ti>=1)
                                    {
                                        if(numberOfICMP>=5)
                                        {

                                            printf("\nAttack.............\n");
                                            exit(1);
                                            numberOfICMP=0;
                                        }
                                        min1=min2;
                                    }
                                }
                            }
                          }

int main(int argc,char **argv)
{
                          //freopen("a.txt","w",stdout);
                          tval1 = time(NULL);
                          numberOfICMP=0;
                          pingFlag=0;
                          handle_icmpFlag=0;
                          /*
                          Declarations
                          */
                          char errbuf[PCAP_ERRBUF_SIZE];  //256 defined in
pcap.h
                          pcap_t* pd;
```

```
                              int snaplen=200;        //len of packet to capture
                              const u_char *packet;//packet
                              bpf_u_int32 maskp;        //mask
                              bpf_u_int32 netp;      //net address
                              char
localnet[INET_ADDRSTRLEN],umask[INET_ADDRSTRLEN];
                              struct pcap_pkthdr hdr;
                              u_char* args = NULL;          // dont know why??????

                              if(dev==NULL)          //ethernet or wlan card   get the
device name to dev
                              {
                              if((dev=pcap_lookupdev(errbuf))==NULL)
                              {
                                  perror("Device Lookup :");
                              }
                              }

                              if(dev == NULL)
                              {
                              printf("%s\n",errbuf); exit(1);
                              }
                              else
                              {
                              printf("\tThe Device Found is :%s\n",dev);           // the
device found is
                              }

                              if((pd=pcap_open_live(dev,snaplen,0,500,errbuf))==NU
LL)  //200bytes of capture ,0 promiscous,500 time out pd ??????
                              perror("Error Open live ");

                              if(pcap_lookupnet(dev,&netp,&maskp,errbuf)<0)  //net
is network address , maskp is the umask value
```

```
                                perror("Error lookup:");

                                printf("\tlocal net %s and umask %s
\n",inet_ntop(AF_INET,&netp,localnet,sizeof(localnet)),inet_ntop
                                (AF_INET,&maskp,umask,sizeof(umask)));



                                if(packet == NULL)
                                {
                                printf("Didn't grab packet\n");
                                exit(1);
                                }

                                now1 = localtime(&tval1);
                                min1=now1->tm_min;
                                /* loop function*/
                                pcap_loop(pd,-1,Process_Packet,args);
                                printf("\n\t-------------------------------------------------------------
---------------\n");

}
```

**Defence Mechanism Code for RST attack:**

```
#include <pcap.h>
#include <stdio.h>
#include <stdlib.h>
#include <errno.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netinet/ip.h>
#include <netinet/ip_icmp.h>
#include <netinet/tcp.h>
#include <netinet/udp.h>
#include <arpa/inet.h>
```

```
#include <netinet/if_ether.h>
#include <net/ethernet.h>
#include <netinet/ether.h>
#include <sys/types.h>
#include <time.h>

                        int numberOfTCP,pingFlag,handle_icmpFlag,type;
                        char *dev;  //device name
                        char a[1000][20],inp[20];
                        time_t tval1,tval2;
                        struct tm *now1,*now2;
                        int min1,min2,ti,size;
//sizes
                        int size_ethernet = sizeof(struct ether_header);
                        int size_ip = sizeof(struct ip);
                        int size_tcp = sizeof(struct tcphdr);
                        int size_icmp=sizeof(struct icmphdr);
                        int size_udp=sizeof(struct udphdr);

int setVal(char inp[])
{
                        int i;
                        for(i=0;i<size;++i)
                        {
                         if(!strcmp(a[i],inp))
                         {
                           return 1;
                         }
                        }
                        return 0;
}
/* function to print payload data */
```

```
void                                    handle_Tcp(const u_char * packet)
{
     struct tcphdr * tcp;
const char *payload;
tcp=(struct tcphdr*)(packet+size_ethernet);
int ff;
if(tcp->rst)
{
   ff=setVal(inp);
   if(ff==0)
                                        printf("\n........... Attack.............\n");
}
}
void                                    handle_Udp(const u_char * packet)
{
}
void                                    handle_Icmp(const u_char * packet)
{
handle_icmpFlag=1;
struct icmphdr * icmphdr;
const char *payload;
icmphdr=(struct icmphdr*)(packet+size_ethernet);
type=icmphdr->type;
if(icmphdr->type==69)
{
                                        ++numberOfICMP;
                                        pingFlag=1;
}
}

void handleIP( const u_char * packet)
{
    int ff;
                                        struct ip *ip;
```

```
                                          ip = (struct ip*)(packet + size_ethernet);
                                          strcpy(inp,inet_ntoa(ip->ip_src));
                                          ff=setVal(inp);
                                           if (ip->ip_p == IPPROTO_TCP)
                                           {
                                               handle_Tcp(packet);
                                           }
                                           else
                                           if (ip->ip_p == IPPROTO_UDP)
                                           {
                                               handle_Udp(packet);
                                           }
                                           else
                                           if (ip->ip_p == IPPROTO_ICMP)
                                           {
                                               //++numberOfICMP;
                                               handle_Icmp(packet);
                                           }
                                           else
                                           if (ip->ip_p == IPPROTO_IP)
                                           {
                                               printf("\n\tProtocol: IP\n");
                                           }
}

void Process_Packet(u_char * a, const struct pcap_pkthdr *pk_header, const u_char *
packet)
                                            {
                                               static int Count=1;
                                               struct ether_header *ethHeader;
                                               ethHeader = (struct ether_header *) packet;
                                               if(ntohs (ethHeader-
>ether_type)==ETHERTYPE_IP)
                                               {
```

```
                                printf("IP");    /*IP*/
                                handleIP(packet);
                                }

                              }

int main(int argc,char **argv)
{
                              //freopen("a.txt","w",stdout);
                              tval1 = time(NULL);
                              numberOfICMP=0;
                              pingFlag=0;
                              handle_icmpFlag=0;
                              /*
                              Declarations
                              */
                              char errbuf[PCAP_ERRBUF_SIZE];  //256 defined in
pcap.h
                              pcap_t* pd;
                              int snaplen=200;        //len of packet to capture
                              const u_char *packet;//packet
                              bpf_u_int32 maskp;        //mask
                              bpf_u_int32 netp;       //net address
                              char
localnet[INET_ADDRSTRLEN],umask[INET_ADDRSTRLEN];
                              struct pcap_pkthdr hdr;
                              u_char* args = NULL;         // dont know why??????

                              if(dev==NULL)           //ethernet or wlan card   get the
device name to dev
                              {
                              if((dev=pcap_lookupdev(errbuf))==NULL)
                              {
                                perror("Device Lookup :");
```

```
                                             }
                                            }

                                            if(dev == NULL)
                                            {
                                             printf("%s\n",errbuf); exit(1);
                                            }
                                            else
                                            {
                                             printf("\tThe Device Found is :%s\n",dev);          // the
device found is
                                            }

                                            if((pd=pcap_open_live(dev,snaplen,0,500,errbuf))==NU
LL)  //200bytes of capture ,0 promiscous,500 time out pd ??????
                                             perror("Error Open live ");

                                            if(pcap_lookupnet(dev,&netp,&maskp,errbuf)<0)  //net
is network address , maskp is the umask value
                                             perror("Error lookup:");

                                            printf("\tlocal net %s and umask %s
\n",inet_ntop(AF_INET,&netp,localnet,sizeof(localnet)),inet_ntop
                                            (AF_INET,&maskp,umask,sizeof(umask)));



                                            if(packet == NULL)
                                            {
                                            printf("Didn't grab packet\n");
                                            exit(1);
                                            }

                                            now1 = localtime(&tval1);
```

```
                                                min1=now1->tm_min;
                                                /* loop function*/
                                                pcap_loop(pd,-1,Process_Packet,args);
                                                printf("\n\t-----------------------------------------------------------
---------------\n");

}
```

**Defence Mechanism Code for Sumrfing attack:**

```
#include <pcap.h>
#include <stdio.h>
#include <stdlib.h>
#include <errno.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netinet/ip.h>
#include <netinet/ip_icmp.h>
#include <netinet/tcp.h>
#include <netinet/udp.h>
#include <arpa/inet.h>
#include <netinet/if_ether.h>
#include <net/ethernet.h>
#include <netinet/ether.h>
#include <sys/types.h>
#include <time.h>

                                                int numberOfICMP,pingFlag,handle_icmpFlag,type;
                                                char *dev;  //device name
                                                time_t tval1,tval2;
                                                struct tm *now1,*now2;
                                                int min1,min2,ti;
//sizes
                                                int size_ethernet = sizeof(struct ether_header);
                                                int size_ip = sizeof(struct ip);
                                                int size_tcp = sizeof(struct tcphdr);
```

```
                                        int size_icmp=sizeof(struct icmphdr);
                                        int size_udp=sizeof(struct udphdr);

/* function to print payload data */


void                                    handle_Tcp(const u_char * packet)
{
}
void                                    handle_Udp(const u_char * packet)
{
}
void                                    handle_Icmp(const u_char * packet)
{
handle_icmpFlag=1;
struct icmphdr * icmphdr;
const char *payload;
icmphdr=(struct icmphdr*)(packet+size_ethernet);
type=icmphdr->type;
if(icmphdr->type==69)
{
                                        //++numberOfICMP;
                                        //pingFlag=1;
                                        if(!strcmp("192.168.1.255",inet_ntoa(ip->ip_dst)))
                                        {
          printf("\n......Attack...........\n");
   }
}
}

void handleIP( const u_char * packet)
{
                                        struct ip *ip;
                                        ip = (struct ip*)(packet + size_ethernet);
                                        if (ip->ip_p == IPPROTO_TCP)
                                        {
                                           handle_Tcp(packet);
                                        }
                                        else
                                        if (ip->ip_p == IPPROTO_UDP)
                                        {
                                           handle_Udp(packet);
                                        }
                                        else
                                        if (ip->ip_p == IPPROTO_ICMP)
                                        {
                                           //++numberOfICMP;
                                           handle_Icmp(packet);
                                        }
                                        else
                                        if (ip->ip_p == IPPROTO_IP)
                                        {
                                           printf("\n\tProtocol: IP\n");
                                        }
}

void Process_Packet(u_char * a, const struct pcap_pkthdr *pk_header, const u_char *
packet)
                                         {
                                           static int Count=1;
                                           struct ether_header *ethHeader;
                                           ethHeader = (struct ether_header *) packet;
                                           if(ntohs (ethHeader-
>ether_type)==ETHERTYPE_IP)
                                           {
                                           printf("IP");    /*IP*/
                                           handleIP(packet);
                                           }
```

```
    }

int main(int argc,char **argv)
{
                                        //freopen("a.txt","w",stdout);
                                        tval1 = time(NULL);
                                        numberOfICMP=0;
                                        pingFlag=0;
                                        handle_icmpFlag=0;
                                        /*
                                        Declarations
                                        */
                                        char errbuf[PCAP_ERRBUF_SIZE];  //256 defined in
pcap.h
                                        pcap_t* pd;
                                        int snaplen=200;        //len of packet to capture
                                        const u_char *packet;//packet
                                        bpf_u_int32 maskp;        //mask
                                        bpf_u_int32 netp;       //net address
                                        char
localnet[INET_ADDRSTRLEN],umask[INET_ADDRSTRLEN];
                                        struct pcap_pkthdr hdr;
                                        u_char* args = NULL;          // dont know why??????

                                        if(dev==NULL)           //ethernet or wlan card   get the
device name to dev
                                        {
                                        if((dev=pcap_lookupdev(errbuf))==NULL)
                                        {
                                            perror("Device Lookup :");
                                        }
                                        }

                                        if(dev == NULL)
                                        {
                                        printf("%s\n",errbuf); exit(1);
                                        }
                                        else
                                        {
                                        printf("\tThe Device Found is :%s\n",dev);          // the
device found is
                                        }

                                        if((pd=pcap_open_live(dev,snaplen,0,500,errbuf))==NU
LL)  //200bytes of capture ,0 promiscous,500 time out pd ??????
                                        perror("Error Open live ");

                                        if(pcap_lookupnet(dev,&netp,&maskp,errbuf)<0)  //net
is network address , maskp is the umask value
                                        perror("Error lookup:");

                                        printf("\tlocal net %s and umask %s
\n",inet_ntop(AF_INET,&netp,localnet,sizeof(localnet)),inet_ntop
                                        (AF_INET,&maskp,umask,sizeof(umask)));



                                        if(packet == NULL)
                                        {
                                        printf("Didn't grab packet\n");
                                        exit(1);
                                        }

                                        now1 = localtime(&tval1);
                                        min1=now1->tm_min;
                                        /* loop function*/
                                        pcap_loop(pd,-1,Process_Packet,args);
                                        printf("\n\t------------------------------------------------------------
---------------\n");

}
```

**Defence Mechanism Code for SYN-ACK attack:**

```
#include <pcap.h>
#include <stdio.h>
#include <stdlib.h>
#include <errno.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netinet/ip.h>
#include <netinet/ip_icmp.h>
#include <netinet/tcp.h>
#include <netinet/udp.h>
#include <arpa/inet.h>
#include <netinet/if_ether.h>
#include <net/ethernet.h>
#include <netinet/ether.h>
#include <sys/types.h>
#include <time.h>

                                        int numberOfTCP,pingFlag,handle_icmpFlag,type;
                                        char *dev;  //device name
                                        time_t tval1,tval2;
                                        struct tm *now1,*now2;
                                        int min1,min2,ti;
//sizes
                                        int size_ethernet = sizeof(struct ether_header);
                                        int size_ip = sizeof(struct ip);
                                        int size_tcp = sizeof(struct tcphdr);
                                        int size_icmp=sizeof(struct icmphdr);
                                        int size_udp=sizeof(struct udphdr);
```

```
/* function to print payload data */


void                                    handle_Tcp(const u_char * packet)
{
     struct tcphdr * tcp;
const char *payload;
tcp=(struct tcphdr*)(packet+size_ethernet);
if(tcp->syn-ack)
{
                                        numberOfTCP++;
}
}
void                                    handle_Udp(const u_char * packet)
{
}
void                                    handle_Icmp(const u_char * packet)
{
handle_icmpFlag=1;
struct icmphdr * icmphdr;
const char *payload;
icmphdr=(struct icmphdr*)(packet+size_ethernet);
type=icmphdr->type;
if(icmphdr->type==69)
{
                                        ++numberOfICMP;
                                        pingFlag=1;
}
}

void handleIP( const u_char * packet)
{
                                        struct ip *ip;
```

```
                                ip = (struct ip*)(packet + size_ethernet);
                                 if (ip->ip_p == IPPROTO_TCP)
                                 {
                                     handle_Tcp(packet);
                                 }
                                 else
                                 if (ip->ip_p == IPPROTO_UDP)
                                 {
                                     handle_Udp(packet);
                                 }
                                 else
                                 if (ip->ip_p == IPPROTO_ICMP)
                                 {
                                     //++numberOfICMP;
                                     handle_Icmp(packet);
                                 }
                                 else
                                 if (ip->ip_p == IPPROTO_IP)
                                 {
                                     printf("\n\tProtocol: IP\n");
                                 }
}

void Process_Packet(u_char * a, const struct pcap_pkthdr *pk_header, const u_char *
packet)
                                  {
                                     static int Count=1;
                                     struct ether_header *ethHeader;
                                     ethHeader = (struct ether_header *) packet;
                                     if(ntohs (ethHeader-
>ether_type)==ETHERTYPE_IP)
                                     {
                                     printf("IP");    /*IP*/
                                     handleIP(packet);
```

```
}
if(numberOfTCP>0)
{
    //(void) time(&t2);
    //if(t2)
    tval2 = time(NULL);
    now2 = localtime(&tval2);
    min2=now2->tm_min;
    printf("min1=%d\n",min1);
    printf("min2=%d\n",min2);
    if(min2<min1)
    {
        ti=60-min1;   //time before 60
        ti=ti+min2; //time in new hour
        if(ti>=1)
        {
            if(numberOfTCP>=5)
            {

                printf("\nAttack.............\n");
                exit(1);
                numberOfTCP=0;
            }
            min1=min2;
        }
    }
    else
    {
        ti=min2-min1;
        if(ti>=1)
        {
            if(numberOfTCP>=5)
            {
```

```
                                                printf("\nAttack.............\n");
                                                exit(1);
                                                numberOfTCP=0;
                                            }
                                            min1=min2;
                                        }
                                    }
                                }
                              }

int main(int argc,char **argv)
{
                              //freopen("a.txt","w",stdout);
                              tval1 = time(NULL);
                              numberOfICMP=0;
                              pingFlag=0;
                              handle_icmpFlag=0;
                              /*
                              Declarations
                              */
                              char errbuf[PCAP_ERRBUF_SIZE];  //256 defined in
pcap.h
                              pcap_t* pd;
                              int snaplen=200;        //len of packet to capture
                              const u_char *packet;//packet
                              bpf_u_int32 maskp;       //mask
                              bpf_u_int32 netp;       //net address
                              char
localnet[INET_ADDRSTRLEN],umask[INET_ADDRSTRLEN];
                              struct pcap_pkthdr hdr;
                              u_char* args = NULL;          // dont know why??????

                              if(dev==NULL)           //ethernet or wlan card   get the
device name to dev
```

```
                                    {
                                     if((dev=pcap_lookupdev(errbuf))==NULL)
                                     {
                                         perror("Device Lookup :");
                                     }
                                    }

                                    if(dev == NULL)
                                    {
                                     printf("%s\n",errbuf); exit(1);
                                    }
                                    else
                                    {
                                     printf("\tThe Device Found is :%s\n",dev);           // the
device found is
                                    }

                                    if((pd=pcap_open_live(dev,snaplen,0,500,errbuf))==NU
LL)  //200bytes of capture ,0 promiscous,500 time out pd ??????
                                     perror("Error Open live ");

                                    if(pcap_lookupnet(dev,&netp,&maskp,errbuf)<0)  //net
is network address , maskp is the umask value
                                     perror("Error lookup:");

                                    printf("\tlocal net %s and umask %s
\n",inet_ntop(AF_INET,&netp,localnet,sizeof(localnet)),inet_ntop
                                    (AF_INET,&maskp,umask,sizeof(umask)));



                                    if(packet == NULL)
                                    {
                                    printf("Didn't grab packet\n");
```

```
                                        exit(1);
                                        }

                                        now1 = localtime(&tval1);
                                        min1=now1->tm_min;
                                        /* loop function*/
                                        pcap_loop(pd,-1,Process_Packet,args);
                                        printf("\n\t-----------------------------------------------------------
---------------\n");

}
```

**Defence Mechanism Code for SYN Flood attack:**

```
#include <pcap.h>
#include <stdio.h>
#include <stdlib.h>
#include <errno.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netinet/ip.h>
#include <netinet/ip_icmp.h>
#include <netinet/tcp.h>
#include <netinet/udp.h>
#include <arpa/inet.h>
#include <netinet/if_ether.h>
#include <net/ethernet.h>
#include <netinet/ether.h>
#include <sys/types.h>
#include <time.h>

                                        int numberOfTCP,pingFlag,handle_icmpFlag,type;
                                        char *dev;  //device name
                                        time_t tval1,tval2;
                                        struct tm *now1,*now2;
                                        int min1,min2,ti;
```

```
//sizes
                                    int size_ethernet = sizeof(struct ether_header);
                                    int size_ip = sizeof(struct ip);
                                    int size_tcp = sizeof(struct tcphdr);
                                    int size_icmp=sizeof(struct icmphdr);
                                    int size_udp=sizeof(struct udphdr);

/* function to print payload data */



void                                handle_Tcp(const u_char * packet)
{
    struct tcphdr * tcp;
const char *payload;
tcp=(struct tcphdr*)(packet+size_ethernet);
if(tcp->syn)
{
                                    numberOfTCP++;
}
}
void                                handle_Udp(const u_char * packet)
{
}
void                                handle_Icmp(const u_char * packet)
{
handle_icmpFlag=1;
struct icmphdr * icmphdr;
const char *payload;
icmphdr=(struct icmphdr*)(packet+size_ethernet);
type=icmphdr->type;
if(icmphdr->type==69)
{
                                    ++numberOfICMP;
                                    pingFlag=1;
}
}

void handleIP( const u_char * packet)
{
                                        struct ip *ip;
                                        ip = (struct ip*)(packet + size_ethernet);
                                        if (ip->ip_p == IPPROTO_TCP)
                                        {
                                            handle_Tcp(packet);
                                        }
                                        else
                                        if (ip->ip_p == IPPROTO_UDP)
                                        {
                                            handle_Udp(packet);
                                        }
                                        else
                                        if (ip->ip_p == IPPROTO_ICMP)
                                        {
                                            //++numberOfICMP;
                                            handle_Icmp(packet);
                                        }
                                        else
                                        if (ip->ip_p == IPPROTO_IP)
                                        {
                                            printf("\n\tProtocol: IP\n");
                                        }
}

void Process_Packet(u_char * a, const struct pcap_pkthdr *pk_header, const u_char *
packet)
                                         {
                                           static int Count=1;
                                           struct ether_header *ethHeader;
```

```
ethHeader = (struct ether_header *) packet;
if(ntohs (ethHeader-
>ether_type)==ETHERTYPE_IP)
{
printf("IP");    /*IP*/
handleIP(packet);
}
if(numberOfTCP>0)
{
    //(void) time(&t2);
    //if(t2)
    tval2 = time(NULL);
    now2 = localtime(&tval2);
    min2=now2->tm_min;
    printf("min1=%d\n",min1);
    printf("min2=%d\n",min2);
    if(min2<min1)
    {
        ti=60-min1;   //time before 60
        ti=ti+min2; //time in new hour
        if(ti>=1)
        {
            if(numberOfTCP>=5)
            {

                printf("\nAttack.............\n");
                exit(1);
                numberOfTCP=0;
            }
            min1=min2;
        }
    }
    else
    {
```

```
                                        ti=min2-min1;
                                        if(ti>=1)
                                        {
                                            if(numberOfTCP>=5)
                                            {

                                                printf("\nAttack.............\n");
                                                exit(1);
                                                numberOfTCP=0;
                                            }
                                            min1=min2;
                                        }
                                    }
                                }
                              }

int main(int argc,char **argv)
{
                              //freopen("a.txt","w",stdout);
                              tval1 = time(NULL);
                              numberOfICMP=0;
                              pingFlag=0;
                              handle_icmpFlag=0;
                              /*
                              Declarations
                              */
                              char errbuf[PCAP_ERRBUF_SIZE];  //256 defined in
pcap.h
                              pcap_t* pd;
                              int snaplen=200;        //len of packet to capture
                              const u_char *packet;//packet
                              bpf_u_int32 maskp;        //mask
                              bpf_u_int32 netp;       //net address
```

```
                        char
localnet[INET_ADDRSTRLEN],umask[INET_ADDRSTRLEN];
                        struct pcap_pkthdr hdr;
                        u_char* args = NULL;        // dont know why??????

                        if(dev==NULL)          //ethernet or wlan card   get the
device name to dev
                        {
                         if((dev=pcap_lookupdev(errbuf))==NULL)
                         {
                           perror("Device Lookup :");
                         }
                        }

                        if(dev == NULL)
                        {
                         printf("%s\n",errbuf); exit(1);
                        }
                        else
                        {
                         printf("\tThe Device Found is :%s\n",dev);          // the
device found is
                        }

                        if((pd=pcap_open_live(dev,snaplen,0,500,errbuf))==NU
LL)  //200bytes of capture ,0 promiscous,500 time out pd ??????
                         perror("Error Open live ");

                        if(pcap_lookupnet(dev,&netp,&maskp,errbuf)<0)  //net
is network address , maskp is the umask value
                         perror("Error lookup:");
```

```
                    printf("\tlocal net %s and umask %s
\n",inet_ntop(AF_INET,&netp,localnet,sizeof(localnet)),inet_ntop
                    (AF_INET,&maskp,umask,sizeof(umask)));



                    if(packet == NULL)
                    {
                    printf("Didn't grab packet\n");
                    exit(1);
                    }

                    now1 = localtime(&tval1);
                    min1=now1->tm_min;
                    /* loop function*/
                    pcap_loop(pd,-1,Process_Packet,args);
                    printf("\n\t-----------------------------------------------------------
---------------\n");

}
```

## Appendix B

**Terms:**

script kiddies: - In computing, a script kiddie (occasionally script bunny, script kitty, script kiddo, "skidiot", skiddie or Victor Skill Deficiency (VSD)) is a derogatory term for inexperienced crackers who use scripts and programs developed by others, without knowing what they are or how they work, for the purpose of compromising computer accounts and files, and for launching attacks on whole computer systems (see DoS). In general, they do not have the ability to write these kinds of programs on their own. Such programs have included WinNuke applications, Back Orifice, NetBus, Sub7, and Metasploit.

It is a common belief that many script kiddies also enjoy cracking any website they can, just to prove their "superiority" in the underground cracker community. Script kiddies, instead of attacking an individual system, often scan thousands of computers looking for vulnerable targets before initiating an attack. This is similar to wardialing and wardriving in which the attacker isn't looking at one specific system, but instead anything that is open and looks interesting. The term is also often used as a derogatory moniker for individuals who do not contribute to the development of new security-related programs, especially exploits, but rather benefit from the work of others.

Malware- Malware or Malicious Software is software designed to infiltrate or damage a computer system without the owner's informed consent.

NIDS-A network intrusion detection system (NIDS) is a system that tries to detect malicious activity such as denial of service attacks, port-scans or even attempts to crack into computers by monitoring network traffic. The NIDS does this by reading all the incoming packets and trying to find suspicious patterns. If, for example, a large number of TCP connection requests to a very large number of different ports are observed, one could assume that there is someone committing a "port scan" at some of the computer(s) in the network. It also (mostly) tries to detect incoming shellcodes in the same manner that an ordinary intrusion detection systems does.

DMZ-(Demilitarized Zone) DMZ is a network area that sits between an organization's internal network and an external network, usually the Internet.

Port mirroring: Port mirroring is used on a network switch to send a copy of all network packets seen on one switch port to a network monitoring connection on another switch port. This is commonly used for network appliances that require monitoring of network traffic, such as an intrusion-detection system. Port mirroring on a Cisco Systems switch is generally referred to as Switched Port Analyzer (SPAN); some other vendors have other names for it, such as Roving Analysis Port (RAP) on 3Com switches.

Network tap-: A network tap is a hardware device which provides a way to access the data flowing across a computer network. Computer networks, including the Internet, are collections of devices, such as computers, routers and switches, which are connected to each other. The connections can utilize different technologies, such as Ethernet, 802.11, FDDI, and ATM. In many cases, it is desirable for a third party to monitor the network traffic between two points in the network, point A and point B. If the network between points A and B consists of a physical cable, a network tap may be the best way to accomplish this monitoring. The network tap has at least three ports -- an A port, a B port, and a monitor port. To place a tap between points A and B, the network cable between point A and point B is replaced with a pair of cables, one going to the tap's A port, one going to the tap's B port. The tap passes through all traffic between A and B, so A and B still think they are connected to each other, but the tap also copies the traffic between A and B to its monitor port, enabling a third party to listen.

https: https is a URI scheme which is syntactically identical to the http: scheme normally used for accessing resources using HTTP. URL indicates that HTTP is to be used, but with a different default port (443) and an additional encryption/authentication layer between HTTP and TCP. This system was invented by Netscape Communications Corporation to provide authentication and encrypted communication and is widely used on the World Wide Web for security-sensitive communication such as payment transactions and corporate logons.

Malicious Code: Code is intended to harm, disrupt, or circumvent Computer and network functions. This code can be mobile, Such as Java applets or code in the Active

X environment. It can also attach itself to legitimate code and propagate; it can lurk in Useful applications or replicate itself across the Internet. The Following sections describe these different types of malware.

Viruses

A virus is code that attaches to a host program and propagates when the infected program is executed. Thus, a virus is Self-replicating and self-executing. Viruses are transmitted in a variety of ways, including as part of files downloaded from the Internet or as e-mail attachments and closely related types of code fall into the following categories:

Macro viruses

These viruses are one of the most common types found and Infect applications such as Microsoft Word or Excel. Recall that a macro is a Set of low-level instructions within an application that is useful in performing Repetitive operations, including modifying and deleting files. In operation, Macro viruses attach to an application's initialization sequence. When the Application is opened; the virus executes instructions before transferring control to the application. Following this activity, the virus replicates itself and Attaches to other code in the computer system.

File infectors

File infector viruses usually attach themselves to executable code, such as .com or .exe files. The virus is then installed when the code is loaded. Another version of a file infector associates itself with a file by creating a virus file with the same name, but with an .exe extension. Therefore, when the file is opened, the virus file will execute.

System or boot-record infectors

Boot-record viruses attach to the master boot record on hard disks or the boot sector on diskettes. When the system is started, it will look at the boot sector and load the virus into memory, where it can propagate to other disks and computers.

Stealth viruses

Stealth viruses take over system functions to conceal themselves. They do this by compromising virus-scanning software so that the software will report an infected area as being uninfected. These viruses conceal any increase in the size of an infected file or changes to the file’s date and time of last modification.

Worms

Worms differ from viruses in that they do not attach to a host file, but are self-contained programs that propagate across networks and computers. Worms are commonly spread through e-mail attachments, which, when opened, activate the worm program. A typical worm exploit would involve the worm sending a copy of itself to everyone in an infected computer’s e-mail address book and/or replicate itself in the infected computer. Also it has the capability to encrypt itself, making the searching for this worm more difficult for antivirus software.

Distributed denial-of-service attacks

A denial-of-service (DoS) attack hogs or overwhelms a system’s resources so that it cannot respond to service requests. A DoS attack can be implemented by flooding a server with so many simultaneous connection requests that it cannot respond. Another approach would be to transfer huge files to a system’s hard drive, exhausting all its storage space. A related attack is the distributed denial-of-service (DDoS)

Examples of DoS attacks include the following:

Buffer overflow — A process receives much more data than expected. If the process has no programmed routine to deal with this excessive amount of data, it acts in an unexpected way that the intruder can exploit. For example, a ping-of-death attack exploits the Internet Control Message Protocol (ICMP) by sending an illegal ECHO packet of more than 65K octets of data, which can cause an overflow of system variables and lead to a system crash.

Smurf — This attack involves using IP spoofing and the ICMP to saturate a target network with traffic, thereby launching a DoS attack. It consists of three elements: the source site, the bounce site, and the target site. The attacker (the source site) sends a spoofed ping packet to the broadcast address of a large network (the bounce site). This modified packet contains the address of the target site. This causes the bounce site to broadcast the misinformation to all of the devices on its local network. All of these devices now reply to the target system, which is then saturated with those replies.

Man-in-the-middle

A man-in-the-middle attack involves an attacker, A, substituting his or her public key for that of another person, P. Then, anyone wanting to send an encrypted message to P using P’s public key is unknowingly using A’s public key. Therefore, a can read the message intended for P. A can then send the message on to P, encrypted in P’s real public key, and P will never be the wiser. Obviously, A could modify the message before resending it to P.

TCP/Hijacking

An attacker hijacks a session between a trusted client and network server. The attacking computer substitutes its IP address for that of the trusted client and the server continues the dialog believing it is communicating with the trusted client.

Simply stated, the steps in this attack are as follows:

1. A trusted client connects to a network server.
2. The attack computer gains control of the trusted client.
43. The attack computer disconnects the trusted client from the network server.
4. The attack computer replaces the trusted client’s IP address with its own IP address and spoofs the client’s sequence numbers.
5. The attack computer continues dialog with the network server believes it is still communicating with trusted client).

Social engineering

This attack uses social skills to obtain information such as passwords or PIN numbers to be used against information systems. For example, an attacker may impersonate someone in an organization and make phone calls to employees of that organization requesting passwords for use in maintenance operations. The following are additional examples of social engineering attacks:

. E-mails to employees from a cracker requesting their passwords to validate the organizational database after a network intrusion has occurred

. E-mails to employees from a cracker requesting their passwords because work has to be done over the weekend on the system

. E-mails or phone calls from a cracker impersonating an official who is conducting an investigation for the organization and requires passwords for the investigation

. Improper release of medical information to individuals posing as doctors and requesting data from patients' records

. A computer repair technician convincing a user that the hard disk on his or her PC is damaged and not repairable and installing a new hard disk for the user, the technician then taking the original hard disk to extract information and sell the information to a competitor or foreign government The best defense against social engineering attacks is an information security policy addressing social of attacks.

<u>Birthday attacks</u>

Birthday attacks are made against hash algorithms that are used to verify the integrity of a message and for digital signatures. A message processed by a hash function produces an output message digest (MD) of fixed length, independent of the length of the input message. The MD uniquely characterizes the message. For a strong hash algorithm, H, and message M, the following is true:

. It should be computationally infeasible to find two messages that produce a common message digest (that is, H(M1) . H(M2)).

. If there exist a message and its corresponding message digest, it should be computationally infeasible to find another message that generates that specific message digest.

<u>Password guessing</u>

Because passwords are the most commonly used mechanism to authenticate users to an information system, obtaining passwords is a common and effective attack approach. Access to a person's password can be obtained by looking around their desk for notes with the password, "sniffing" the connection to the network to acquire unencrypted passwords, through social engineering, gaining access to a password database, or outright guessing. The last approach can be done in a random or systematic manner.

Brute force

Brute-force password guessing means using a random approach by trying different passwords and hoping that one works. Some logic can be applied by trying passwords related to the person’s name, job title, hobbies, or other similar items.

Dictionary attack

A dictionary attack is one in which a dictionary of common passwords is used in an attempt to gain access to a user’s computer and network. One approach is to copy an encrypted file that contains the passwords and, applying the same encryption to a dictionary of commonly can be automated.

Honeypots

A different approach to intrusion detection and response is the use of a honeypot. A honeypot is a monitored decoy mechanism that is used to entice a hacker away from valuable network resources and provide an early indication of an attack. It also provides for detailed examination of an attacker during and following honeypot exploitation. A definition of a honeypot provided by Lance Spitzner, President of the Honeynet Project, is, “an information system resource whose value lies in unauthorized or illicit use of that resource.” The Honeynet Project is a non-profit research organization of volunteer security the state of the art in information system security.

Honeypots are employed primarily for either research or production purposes. In The research mode, a honeypot collects information on new and emerging threats, attack trends, motivations, and, essentially, characterizes the attacker community. In the production category, honeypots are applied to preventing attacks, detecting attacks and responding to attacks. The methods for accomplishing these tasks are summarized in the following sections.

Honeypots are effective in preventing attacks by doing the following:

. Slowing or impeding scans initiated by worms or automated attacks by monitoring unused IP space and detecting scanning activity

. Consuming an attacker’s energy through interaction with a honeypot while the attack is detected, analyzed, and handled

. Deterring an attack by a cracker who suspects a network employs honeypots and is concerned about getting caught

Detecting attacks

Network security, no matter how conscientiously and effectively applied, cannot prevent all attacks all of the time. Therefore, honey pots offer means to detect an attack that is taking place or has occurred. Honeypots have the following advantages in detecting attacks:

. The ability to capture new and unknown attacks

. The ability to capture polymorphic code

# REFERENCE

1. Raymond Panko, *Corporate Computer and Network Security*, Prentice Hall, 2004,Third Edition
2. Emilie Lundin Barse. Logging for Intrusion and Fraud Detection; PhD Thesis, Chalmers University of Technology, Goteborg, Sweden
3. William Stallings, *Cryptography and Network Security*, 3rd Edition.
4. Andrew S. Tanenbaum, Computer Networks*,* 5th edition, Prentice Hall PTR.
5. Network Perimeter Defense- CERT in Department of Information Technology Ministry of Communication & Information Technology, Electronics Niketon, 6 C.G.O. Complex, New Delhi-110003, India
6. Assessing Network Infrastructure Vulnerabilities to Physical Layer Attacks-T. H. Shake., B. HazzardÁ, D. Marquis. Distributed Systems Group, ÁAdvanced Networks Group, Lincoln Laboratory, Massachusetts Institute of Technology, Lexington, Massachusetts, USA
7. Adaptive Multi-Layer Network Survivability: A Unified Framework for Countering Cyber-Terrorism, William Yurcik, Telecommunications Program, University of Pittsburgh
8. Intrusion Detection Systems (IDS) Part I - (network intrusions; attack symptoms; IDS tasks; and IDS architecture); Author: Przemyslaw Kazienko & Piotr Dorosz
http://www.windowsecurity.com/articles/Intrusion_Detection_Systems_IDS_Part_I__net work_intrusions_attack_symptoms_IDS_tasks_and_IDS_architecture.html?printversion
9. Cable Cuts by Matt Hoffman:http://all.net/CID/Attack/papers/CableCuts.html
10. Consolidation and Evaluation of IDS Taxonomies, Magnus Almgren Emilie Lundin Barse Erland Jonsson, Department of Computer Engineering, Chalmers University of Technology, SE-412 96 G¨oteborg, Sweden
11.RoutingAttack:
http://www3.cc.gatech.edu/classes/AY2003/cs6262_spring/routing_attack.ppt
12. www.wikimediafoundation.org
13. Packet Sniffing on Layer 2 Switched Local Area Networks, Ryan Spangler Packet watch Research: http://www.packetwatch.net
14. Distributed Reflection Denial of Service, by Steve Gibson, Gibson Research Corporation: www.grc.com
15. ACK Dos Attack: http://www.checkpoint.com/Attack.htm

16. Network layer Attack by Pathak: http://www.phatak.com/Network-Layer-DoS.php
17. Attacks at the Data Link Layer By GUILLERMO MARIO MARRO
Electronic Engineer (Universidad Nacional de Rosario)
http://seclab.cs.ucdavis.edu/papers/Marro_masters_thesis.pdf
18. IDS information by security focus: http://www.securityfocus.com/static/ids