NETWORK SECURITY AND INTRUSION DETECTION SYSTEM

Rajib Rahman

Student ID 02101080

Department of Computer Science and Engineering

January 2007

BRAC University, Dhaka, Bangladesh

Network Security And Intrusion Detection System

Thesis report prepared by

Rajib Rahman ID 02101080

Under the supervision of

Mushfiqur Rouf
Lecturer
Department of Computer Science and Engineering, BRAC University

Department of Computer Science and Engineering
BRAC University
Fall 2006

Foreword

The thesis report is submitted in partial fulfillment of the academic requirements for the degree of Bachelor of Science in Computer Science and Engineering to the Computer Science and Engineering department at BRAC University, 66 Mohakhali Dhaka-1212, Bangladesh.

Rajib Rahman ID 02101080

As the supervisor of the candidates I have approved this dissertation for submission.

…….

Mushfiqur Rouf
Lecturer
Department of Computer Science and Engineering
BRAC University

Declaration

The whole dissertation, unless specifically indicated to the contrary in the text, is our original work, and has not been submitted in part, or in whole for the degree or diploma to any other university.

…….

Rajib Rahman
ID 02101080

Approval Sheet

The thesis report titled **"Network Security And Intrusion Detection System"** has been submitted to the following respected members of the Board of Examiners from the Faculty of Computer Science and Engineering in partial fulfillment of the academic requirements for the degree of Bachelor of Science in Computer Science and Engineering on January 07, 2007 by the following students and has been accepted as satisfactory.

Rajib Rahman ID 02101080

…….....

Mushfiqur Rouf
Lecturer
Department of Computer Science
and Engineering
BRAC University

Acknowledgments

At first my heartiest gratitude goes to Almighty ALLAH, without His divine blessing it would not be possible for me to complete this project successfully. Then I am eternally grateful to my parents for their constant support throughout our academic work.

This was a group project and I am give my special thanks to my group members Sarwar Alam and Salman Zaman for working together with me and giving great support.

It's been a pleasure and an honor to show our respect to Mushfiqur Rouf, Lecturer, Department of Computer Science and Engineering, BRAC University, who has rendered continuous and encouraging guidance throughout the entire thesis period with his kind advice. His generosity and zeal for this thesis is exemplary.

We want to thank all the honorable faculty members especially Risat Mahmud Pathan, Lecturer, Department of Computer Science and Engineering, BRAC University, for their feedback and inspiration that was really instrumental in shaping this thesis.

Finally we like to offer thanks to all of our friends and well–wishers for helping us by rigorous reviews of this work and inspiring suggestion.

Table of Contents

Abstract

Objective of this project is to study various ways network security could be challenged, analyze the events that lead to vulnerabilities and hacking in remote network manipulation method, learn methodologies used to compromise remote systems, and developing a software that will detect this remote intrusion. It will be oriented towards the study of network security as a whole, and the development of a working Network Based Intrusion Detection System.

Chapter 1 Introduction

In the growing use of Internet in todays business, corporate institutions and almost everywhere in everyday life people are getting connected in the huge global internetwork. Hence the risk from network related attacks are growing and becoming a serious concern. Considering the importance of security, it is unwise to consider Internet as not so dangerous means of vulnerability, instead the risk it poses should be looked at thoroughly and security measures should be undertaken.

Network security is a very challenging work in todays world. Attackers are trying to break the security protocol using various malicious means. Security for a network, being similar in maintenance as Institution security, has many forms of solution to tackle misuse of network protocol which includes gateway security, authentication, encryption, monitoring routers, and firewalls. But most of them can be bypassed despite their strength in protection. Only firewall and others are not enough to protect a network, in addition to them we need some extra protection to protect our network. We need a mechanism which can capture packets, analyze them and decide on the behavior of a particular communicating host whether it could be considered malicious or fair, and give alarm to the firewall or network system administrator as necessary. Intrusion Detection System is the best technique for this purpose.

Intrusion Detection System or IDS is a software or hardware based protection systems that monitor the events occurring or threats in a network, analyzing them for signatures of security problems. The goal of IDS is to identify, preferably in real time, unauthorized use, misuse, and abuse of computer systems from both internal and external means from a networks point of view. As network attacks have increased day

by day in a alarming rate, Intrusion Detection Systems is becoming a necessary addition to the security infrastructure (firewall) of most organizations. Developing an Intrusion Detection System is a challenging work due to complex and immense nature of computer internetwork which increases its connectivity day by day and gives greater access of internal systems to the outsiders and makes it easier to attack and compromise a computer of an institution. The principle behavior of an IDS will be to distinguish an intruder from that of a legitimate user.

Chapter 2 Intrusion Detection System (IDS)

Our deepest acknowledgment goes to Charles P. and Shari Lawrence Pfleeger for their book on "Security in Computing" from which major quotations were made when writing most of the initial Introductory texts below which was part of our research phase.

2.1 Definition of IDS

Intrusion detection and response is the task of monitoring for evidence occurring in a computer system or network and analyzing them for signs of intrusions, defined as attempts to compromise the confidentiality, integrity, availability, or to bypass the security mechanisms of a computer or network and responding to this evidence. Response includes notifying the appropriate parties to take action against intruders accessing the systems from the Internet, authorized users of the systems who attempt to gain additional privileges for which they are not authorized, and authorized users who misuse the privileges given them. IDS, therefore, is the detection of inappropriate, incorrect, or anomalous activity.

IDS have two primary components:

- Creation and maintenance of intrusion detection systems (IDSs) and processes for host and network monitoring and event notification
- Creation of a computer incident response team (CIRT) for the following tasks:
    - Analysis of an event notification
    - Response to an incident if the analysis warrants it
    - Escalation path procedures
    - Resolution, post-incident follow-up, and reporting to appropriate parties

IDS is a software or hardware product that monitors, analyze, and tracks network traffic or host audit logs to determine whether any violations of an organization's security policy have taken place. IDS can detect intrusions that have circumvented or passed through a firewall or that are occurring within the local area network (LAN) behind the firewall.

IDSs perform a variety of functions:

- Monitoring users and system activity
- Auditing system configuration for vulnerabilities and misconfiguration
- Assessing the integrity of critical system and data files
- Recognizing known attack patterns in system activity
- Identifying abnormal activity through statistical analysis
- Managing audit trails and highlighting user violation of policy or normal activity
- Correcting system configuration errors
- Installing and operating traps to record information about intruders

2.2 Necessity of IDS

Intrusion detection system allows organizations to protect their systems from the threats that come with increasing network connectivity and reliance on information systems. IDSs have gained acceptance as a necessary addition to every organization's security infrastructure.

A fundamental goal of computer security management is to affect the behavior of individual users in a way that protects information systems from security problems. Intrusion detection systems help organizations accomplish this goal by increasing the

perceived risk of discovery and punishment of attackers. This serves as a significant deterrent to those who would violate security policy.

Attackers, using widely publicized techniques, can gain unauthorized access to many, if not most systems, especially those connected to public networks. This often happens when known vulnerabilities in the systems are not corrected.

In an ideal world, commercial software vendors would minimize vulnerabilities in their products, and user organizations would correct all reported vulnerabilities quickly and reliably. However, in the real world, this seldom happens thanks to our reliance on commercial software where new flaws and vulnerabilities are discovered on a daily basis.

Intrusion detection can represent an excellent approach to protecting a system. IDS can detect when an attacker has penetrated a system by exploiting an uncorrected or uncorrectable flaw. Furthermore, it can serve an important function in system protection, by bringing the fact that the system has been attacked to the attention of the administrators who can contain and recover any damage that results. This is far preferable to simply ignoring network security threats where one allows the attackers continued access to systems and the information on them.

When adversaries attack a system, they typically do so in predictable stages. The first stage of an attack is usually probing or examining a system or network, searching for an optimal point of entry. In systems with no IDS, the attacker is free to thoroughly examine the system with little risk of discovery or retribution. Given this unfettered access, a determined attacker will eventually find vulnerability in such a network and exploit it to gain entry to various systems.

The same network with an IDS monitoring its operations presents a much more difficult challenge to that attacker. Although the attacker may probe the network for weaknesses, the IDS will observe the probes, will identify them as suspicious, may actively block the attacker's access to the target system, and will alert security personnel who can then take appropriate actions to block subsequent access by the attacker. Even the presence of a reaction to the attacker's probing of the network will elevate the level of risk the attacker perceives, discouraging further attempts to target the network.

IDSs verify, itemize, and characterize the threat from both outside and inside organization's network, assisting you in making sound decisions regarding your allocation of computer security resources. Using IDSs in this manner is important, as many people mistakenly deny that anyone (outsider or insider) would be interested in breaking into their networks. Furthermore, the information that IDSs give you regarding the source and nature of attacks allows you to make decisions regarding security strategy driven by demonstrated need, not guesswork or folklore.

When IDSs run over a period of time, patterns of system usage and detected problems can become apparent. These can highlight flaws in the design and management of security for the system, in a fashion that supports security management correcting those deficiencies before they cause an incident.

2.3 Types of IDS

There are several types of IDSs available today, characterized by different monitoring and analysis approaches. Each approach has distinct advantages and disadvantages. The most common approaches of IDS are statistical anomaly detection (also known as behavior-based) and signature-based (also known as knowledge-based

or pattern-matching) detection. IDSs that operate on a specific host and detect malicious activity on that host are called host-based IDSs. IDSs that operate on network segments and analyze that segment's traffic are called network-based IDSs. Because there are pros and cons of each, an effective IDS should use a combination of both network- and host-based IDSs. A truly effective IDS will detect common attacks as they occur, which includes distributed attacks.

2.3.1 Network Based IDS

The majority of commercial IDSs are network based. Network-based IDSs reside on a discrete network segment and monitor the traffic on that segment. They usually consist of a network appliance with a network interface card (NIC) that is operating in promiscuous mode and is intercepting and analyzing the network packets in real time. Listening on a network segment or switch, one network-based IDS can monitor the network traffic affecting multiple hosts that are connected to the network segment, thereby protecting those hosts.

Network-based IDSs often consist of a set of single-purpose sensors or hosts placed at various points in a network. These sensors can only see the packets that happen to be carried on that particular network segment, monitor network traffic, performing local analysis of that traffic and reporting attacks to a central management console. As the sensors are limited to running the IDS, they can be more easily secured against attack. Many of these sensors are designed to run in "stealth" mode, in order to make it more difficult for an attacker to determine their presence and location.

Packets are identified to be of interest if they match a signature. Three primary types of signatures are as follows:

- **String signatures**—String signatures look for a text string that indicates a possible attack.
- **Port signatures**—Port signatures watch for connection attempts to well known, frequently attacked ports.
- **Header condition signatures**—Header signatures watch for dangerous or illogical combinations in packet headers.

2.3.1.1 Advantages of Network-Based IDS

- A few well-placed network-based IDSs can monitor a large network.
- The deployment of network-based IDSs has little impact upon an existing network. Network-based IDSs are usually passive devices that acquire data and review packets and headers on a network without interfering with the normal operation of a network. Thus, it is usually easy to retrofit a network to include network-based IDSs with minimal effort.
- Network-based IDSs can be made very secure against attack and even made invisible to many attackers.
- It can also detect denial of service attacks.

2.3.1.2 Disadvantages of Network-Based IDS

- Network-based IDSs may have difficulty processing all packets in a large or busy network and, therefore, may fail to recognize an attack launched during periods of high traffic. Some vendors are attempting to solve this problem by implementing IDSs completely in hardware, which is much faster. The need to

analyze packets quickly also forces vendors to both detect fewer attacks and also detect attacks with as little computing resource as possible which can reduce detection effectiveness.

- Many of the advantages of network-based IDSs don't apply to more modern switch-based networks. Switches subdivide networks into many small segments (usually one fast Ethernet wire per host) and provide dedicated links between hosts serviced by the same switch. Most switches do not provide universal monitoring ports and this limits the monitoring range of a network-based IDS sensor to a single host. Even when switches provide such monitoring ports, often the single port cannot mirror all traffic traversing the switch.
- Network-based IDSs cannot analyze encrypted information. This problem is increasing as more organizations (and attackers) use virtual private networks.
- Most network-based IDSs cannot tell whether or not an attack was successful; they can only discern that an attack was initiated. This means that after a network-based IDS detects an attack, administrators must manually investigate each attacked host to determine whether it was indeed penetrated.
- Some network-based IDSs have problems dealing with network-based attacks that involve fragmenting packets. These malformed packets cause the IDSs to become unstable and crash.

2.3.2 Host Based IDS

Host-based IDSs use small programs (intelligent agents) that reside on a host computer. These IDSs operate on information collected from within an individual computer system. This vantage point allows host-based IDSs to analyze activities with great reliability and precision, determining exactly which processes and users are

involved in a particular attack on the operating system, writing to log files, and triggering alarms. Furthermore, unlike network-based IDSs, host-based IDSs can “see” the outcome of an attempted attack, as they can directly access and monitor the data files and system processes usually targeted by attacks. Host-based systems look for activity only on the host computer; they do not monitor the entire network segment.

Host-based IDSs normally utilize information sources of two types, operating system audit trails, and system logs. Operating system audit trails are usually generated at the innermost (kernel) level of the operating system, and are therefore more detailed and better protected than system logs. However, system logs are much less obtuse and much smaller than audit trails, and are furthermore far easier to comprehend. Some host-based IDSs are designed to support a centralized IDS management and reporting infrastructure that can allow a single management console to track many hosts. Others generate messages in formats that are compatible with network management systems. In particular, host-based IDSs have the following characteristics:

- They monitor accesses and changes to critical system files and changes in user privileges.
- They detect trusted insider attacks better than network-based IDS.
- They are relatively effective for detecting attacks from the outside.
- They can be configured to look at all network packets, connection attempts, or login attempts to the monitored machine, including dial-in attempts or other non–network-related communication ports

2.3.2.1 Advantages of Host Based IDS

- Host-based IDSs, with their ability to monitor events local to a host, can detect attacks that cannot be seen by network-based IDS.
- Host-based IDSs can often operate in an environment in which network traffic is encrypted, when the host-based information sources are generated before data is encrypted and/or after the data is decrypted at the destination host.
- Host-based IDSs are unaffected by switched networks.
- When these IDSs operate on OS audit trails, they can help to detect Trojan Horse or other attacks that involve software integrity breaches. These appear as inconsistencies in process execution.

2.3.2.2 Disadvantages of Host Based IDS

- Host-based IDSs are harder to manage, as information must be configured and managed for every host monitored.
- Since at least the information sources (and sometimes part of the analysis engines) for host-based IDSs reside on the host targeted by attacks, the IDS may be attacked and disabled as part of the attack.
- Host-based IDSs are not well suited for detecting network scans or other such surveillance that targets an entire network, because the IDS only sees those network packets received by its host.
- Host-based IDSs can be disabled by certain denial-of-service attacks.
- When host-based IDSs use operating system audit trails as an information source, the amount of information can be immense, requiring additional local storage on the system.

- Host-based IDSs use the computing resources of the hosts they are monitoring, therefore inflicting a performance cost on the monitored systems.

2.3.3 Application-Based IDS

Application-based IDSs are a special subset of host-based IDSs that analyze the events transpiring within a software application. The most common information sources used by application-based IDSs are the application's transaction log files. The ability to interface with the application directly, with significant domain or application-specific knowledge included in the analysis engine, allows application-based IDSs to detect suspicious behavior due to authorized users exceeding their authorization. This is because such problems are more likely to appear in the interaction between the user, the data, and the application.

2.3.3.1 Advantages of Application-Based IDS

- Application-based IDSs can monitor the interaction between user and application, which often allows them to trace unauthorized activity to individual users.
- Application-based IDSs can often work in encrypted environments, since they interface with the application at transaction endpoints, where information is presented to users in unencrypted form.

2.3.3.2 Disadvantages of Application-Based IDS

- Application-based IDSs may be more vulnerable than host-based IDSs to attacks as the applications logs are not as well-protected as the operating system audit trails used for host-based IDSs.

- As Application-based IDSs often monitor events at the user level of abstraction, they usually cannot detect Trojan Horse or other such software tampering attacks. Therefore, it is advisable to use Application-based IDS in combination with Host-based and/or Network-based IDSs.

2.3.4 Signature-Based IDS

In a signature-based IDS or knowledge-based IDS, signatures or attributes that characterize an attack are stored for reference. Then, when data about events is acquired from host audit logs or from network packet monitoring, this data is compared with the attack signature database. If there is a match, a response is initiated. This method is more common than using behavior-based IDSs. Signature-based IDSs are characterized by low false alarm rates (or positives) and, generally, are standardized and understandable by security personnel.

2.3.4.1 Disadvantage of Signature-Based IDS

- A weakness of the signature-based IDS approach is the failure to characterize slow attacks that extend over a long period of time. To identify these types of attacks, large amounts of information must be held for extended time periods. Another issue with signature-based IDSs is that only attack signatures that are stored in their database are detected.
- The IDS is resource-intensive. The knowledge database continually needs maintenance and updating with new vulnerabilities and environments to remain accurate.

- Because knowledge about attacks is much focused (dependent on the operating system, version, platform, and application), new, unique, or original attacks often go unnoticed.

2.3.5 Statistical Anomaly Based IDS

Statistical anomaly or behavior-based IDSs dynamically detect deviations from the learned patterns of “normal” user behavior and trigger an alarm when an intrusive activity occurs. Behavior-based IDSs learn normal or expected behavior of the system or the users and assume that an intrusion can be detected by observing deviations from this norm.

With this method, an IDS acquires data and defines a “normal” usage profile for the network or host that is being monitored. This characterization is accomplished by taking statistical samples of the system over a period of normal use. Typical characterization information used to establish a normal profile includes memory usage, CPU utilization, and network packet types. With this approach, new attacks can be detected because they produce abnormal system statistics.

2.3.5.1 Advantages of Statistical Anomaly Based IDS

- The system can dynamically adapt to new, unique, or original vulnerabilities.
- This IDS is not as dependent upon specific operating systems as a knowledge-based IDS.
- They help detect abuse-of-privileges types of attacks that do not actually involve exploiting any security vulnerability.

2.3.5.2 Disadvantage of Statistical Anomaly Based IDS

- Statistical anomaly based IDSs are not detect an attack that does not significantly change the system-operating characteristics, and it might falsely detect a non-attack event that caused a momentary anomaly in the system.
- High false alarm rates. High positives are the most common failure of behavior-based ID systems and can create data noise that can make the system unusable or difficult to use.
- Activity and behavior of the users of a networked system might not be static enough to effectively implement a behavior-based ID system.
- Network may experience an attack at the same time IDS learning the behavior.

Chapter 3 Threats in Networks

Most computer attacks only corrupt a system's security in very specific ways. Attacks against network resources are common in today's Internet-dependent world. Attacks are launched for a variety of reasons, including monetary gain, maliciousness (as a challenge), recognition for their activities, fraud, warfare, ideology, and to gain an economic advantage. Despite the varied capabilities of computer attacks, they usually result in violation of only four different security properties: availability, confidentiality, integrity, and control. These violations are described below.

- **Confidentiality:** An attack causes a confidentiality violation if it allows attackers to access data without authorization (either implicit or explicit) from the owner of the information.
- **Integrity:** An attack causes an integrity violation if it allows the (unauthorized) attacker to change the system state or any data residing on or passing through a system
- **Availability:** An attack causes an availability violation if it keeps an authorized user (human or machine) from accessing a particular system resource when, where, and in the form that they need it.
- **Control:** An attack causes a control violation if it grants an (unauthorized) attacker privilege in violation of the access control policy of the system. This privilege enables a subsequent confidentiality, integrity, or availability violation.

Attacks are directed at compromising the confidentiality, integrity, and availability of networks and their resources and fall into the following four general categories:

- **Modification attack**—Unauthorized alteration of information
- **Repudiation attack**—Denial that an event or transaction ever occurred
- **Denial-of-service attack**—Actions resulting in the unavailability of network resources and services, when required
- **Access attack**—Unauthorized access to network resources and information

3.1 Port Scan

An attacker 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. By port scanning an attacker can know about three features:

- Which standard ports or services (such as e-mail, FTP, and remote logon) are running and responding on the target system
- What operating system is installed on the target system
- What applications and versions of applications are present

This information is readily available for the asking from a networked system; it can be obtained quietly, anonymously, without identification or authentication, drawing little or no attention to the scan.

Types of port scans include:

- Vanilla - An attempt to connect to all ports (there are 65,536)
- Strobe - An attempt to connect to only selected ports (typically, under 20)
- Stealth scan - Several techniques for scanning that attempt to prevent the request for connection being logged
- FTP Bounce Scan - Attempts that are directed through an File Transfer Protocol server to disguise the cracker's location
- Fragmented Packets - Scans by sending packet fragments that can get through simple packet filters in a firewall
- UDP - Scans for open User Datagram Protocol ports
- Sweep - Scans the same port on a number of computers

3.2 Social Engineering

Social engineering 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 to 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 unrepairable 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

Because the victim has helped the attacker, the victim will think nothing is wrong and not report the incident. Thus, the damage may not be known for some time. An attacker has little to lose in trying a social engineering attack. At worst it will raise awareness of a possible target. But if the social engineering is directed against someone who is not skeptical, especially someone not involved in security management, it may well succeed. We as humans like to help others when asked politely.

3.3 Reconnaissance

Reconnaissance is the general term for collecting information. In security it often refers to gathering discrete bits of information from various sources and then putting them together like the pieces of a puzzle. Commonly used reconnaissance techniques are dumpster diving & eavesdropping.

3.3.1 Dumpster Diving

Dumpster diving involves the acquisition of information that is discarded by an individual or organization. In many cases, information found in trash can be very valuable to a cracker. Discarded information may include network diagrams, technical manuals, security device configurations, password lists, system designs and source

code, telephone numbers, and organization charts. It is important to note that one requirement for information to be treated as a trade secret is that the information be protected and not revealed to any unauthorized individuals. If a document containing an organization's trade secret information is inadvertently discarded and found in the trash by another person, the other person can use that information because it was not adequately protected by the organization.

3.3.2 Eavesdropping

Eavesdropping attacks occur through the interception of network traffic. This situation is particularly prevalent when a network includes wireless components and remote access devices. By eavesdropping, an attacker can obtain passwords, credit card numbers, and other confidential information that a user might be sending over the network. Examples of the various manners of eavesdropping include the following:

- **Passive eavesdropping**—Unauthorized, covert monitoring of transmissions
- **Active eavesdropping**—Probing, scanning, or tampering with a transmission channel to access the transmitted information
- **Inductance**—By this process an intruder can tap a wire and read radiated signals without making physical contact with the cable. A cable's signals travel only short distances, and they can be blocked by other conductive materials. The equipment needed to pick up signals is inexpensive and easy to obtain, so inductance threats are a serious concern for cable-based networks. For the attack to work the intruder must be fairly close to the cable; this form of attack is thus limited to situations with reasonable physical access.

3.3.3 Operating System and Application Fingerprinting

The port scan supplies the attacker with very specific information. For instance, an attacker can use one to find out that port 80 is open and supports HTTP, the protocol for transmitting web pages. But the attacker is likely to have many related questions, such as which commercial server application is running, what version, and what the underlying operating system and version are. Once armed with this additional information, the attacker can consult a list of specific software's known vulnerabilities to determine which particular weaknesses to try to exploit.

How can the attacker answer these questions? The network protocols are standard and vendor independent. Still, each vendor's code is implemented independently, so there may be minor variations in interpretation and behavior. The variations do not make the software non compliant with the standard, but they are different enough to make each version distinctive. For example, each version may have different sequence numbers, TCP flags, and new options. To see why, consider that sender and receiver must coordinate with sequence numbers to implement the connection of a TCP session. Some implementations respond with a given sequence number, others respond with the number one greater, and others respond with an unrelated number. Likewise, certain flags in one version are undefined or incompatible with others. How a system responds to a prompt (for instance, by acknowledging it, requesting retransmission, or ignoring it) can also reveal the system and version. Finally, new features offer a strong clue: A new version will implement a new feature but an old version will reject the request. All these peculiarities, sometimes called the operating system or application fingerprint, can mark the manufacturer and version.

3.3.4 Bulletin Boards and Chats

The Internet is probably the greatest tool for sharing knowledge since the invention of the printing press. It is probably also the most dangerous tool for sharing knowledge. Numerous underground bulletin boards and chat rooms support exchange of information. Attackers can post their latest exploits and techniques, read what others have done, and search for additional information on systems, applications, or sites. Remember that, as with everything on the Internet, anyone can post anything, so there is no guarantee that the information is reliable or accurate.

3.3.5 Availability of Documentation

The vendors themselves sometimes distribute information that is useful to an attacker. For example, Microsoft produces a resource kit by which application vendors can investigate a Microsoft product in order to develop compatible, complementary applications. This toolkit also gives attackers tools to use in investigating a product that can subsequently be the target of an attack.

3.3.6 Theft of Service

In a wireless network many hosts run the Dynamic Host Configuration Protocol (DHCP), by which a client negotiates a one-time IP address and connectivity with a host. A small number of IP addresses can be shared among users. Essentially the addresses are available in a pool. A new client requests a connection and an IP address through DHCP, and the server assigns one from the pool. This scheme admits a big problem with authentication. Unless the host authenticates users before assigning a connection, any requesting client is assigned an IP address and network access. (Typically, this assignment occurs before the user on the client workstation actually

identifies and authenticates to a server, so there may not be an authenticable identity that the DHCP server can demand.) The situation is so serious that in some metropolitan areas a map is available, showing many accepting wireless connections. A user wanting free Internet access can often get it simply by finding a wireless LAN offering DHCP service.

3.4 Protocol Flaws

Internet protocols are publicly posted for scrutiny by the entire Internet community. Each accepted protocol is known by its Request for Comment (RFC) number. Many problems with protocols have been identified by sharp reviewers and corrected before the protocol was established as a standard.

But protocol definitions are made and reviewed by fallible humans. Likewise, protocols are implemented by fallible humans. For example, TCP connections are established through sequence numbers. The client (initiator) sends a sequence number to open a connection, the server responds with that number and a sequence number of its own, and the client responds with the server's sequence number. Suppose someone can guess a client's next sequence number. That person could impersonate the client in an interchange. Sequence numbers are incremented regularly, so it can be easy to predict the next number.

3.4.1 Fragmentation Attacks

A fragmentation attack is used as a method of getting packets around a packet filtering firewall. In a basic fragmentation attack, packets are broken into fragments with the first packet containing the complete header data. The remaining packets do not

contain any header information. Because some routers filter packets based on this header information, the remaining packets without header data are not filtered and pass through the firewall.

Two examples of fragmentation attacks follow:

- A tiny fragment attack occurs when the intruder sends a very small fragment that forces some of the TCP header field into a second fragment. If the target's filtering device does not enforce minimum fragment size, this illegal packet can then be passed on through the target's network.
- An overlapping fragment attack is another variation on a datagram's zero-offset modification (similar to the teardrop attack). Subsequent packets overwrite the initial packet's destination address information, and then the second packet is passed by the target's filtering device. This action can happen if the target's filtering device does not enforce a minimum fragment offset for fragments with non-zero offsets.

3.5 Impersonation

Impersonation is a more significant threat in a wide area network than in a local one. Local individuals often have better ways to obtain access as another user; they can, for example, simply sit at an unattended workstation. Still, impersonation attacks should not be ignored even on local area networks, because local area networks are sometimes attached to wider area networks without anyone's first thinking through the security implications. In an impersonation, an attacker has several choices:

- Guess the identity and authentication details of the target.
- Pick up the identity and authentication details of the target from a previous communication or from wiretapping.

- Circumvent or disable the authentication mechanism at the target computer.
- Use a target that will not be authenticated.
- Use a target whose authentication data are known.

3.5.1 Authentication Foiled by Guessing

Many users choose easy-to-guess passwords. Many worms tried to impersonate each user on a target machine by trying, in order, a handful of variations of the user name, a list of common passwords and, finally, the words in a dictionary. Sadly, many users' accounts are still open to these easy attacks.

A second source of password guesses is default passwords. Many systems are initially configured with default accounts having GUEST or ADMIN as login IDs; accompanying these IDs are well-known passwords such as "guest" or "null" or "password" to enable the administrator to set up the system. Administrators often forget to delete or disable these accounts, or at least to change the passwords.

In a trustworthy environment, such as an office LAN, a password may simply be a signal that the user does not want others to use the workstation or account. Sometimes the password-protected workstation contains sensitive data, such as employee salaries or information about new products. Users may think that the password is enough to keep out a curious colleague; they see no reason to protect against concerted attacks. However, if that trustworthy environment is connected to an untrustworthy wider-area network, all users with simple passwords become easy targets. Indeed, some systems are not originally connected to a wider network, so their users begin in a less exposed situation that clearly changes when the connection occurs.

Dead accounts offer a final source of guessable passwords. Now the attacker uses social engineering on the system administration. Alternatively, the attacker can try several passwords until the password guessing limit is exceeded. The system then locks the account administratively, and the attacker uses a social engineering attack. In all these ways the attacker may succeed in resetting or discovering a password.

3.5.2 Authentication Thwarted by Eavesdropping or Wiretapping

Because of the rise in distributed and client-server computing, some users have access privileges on several connected machines. To protect against arbitrary outsiders using these accesses, authentication is required between hosts. This access can involve the user directly, or it can be done automatically on behalf of the user through a host-to-host authentication protocol. In either case, the account and authentication details of the subject are passed to the destination host. When these details are passed on the network, they are exposed to anyone observing the communication on the network. An impersonator can reuse these same authentication details until they are changed.

3.5.3 Authentication Foiled by Avoidance

Obviously, authentication is effective only when it works. A weak or flawed authentication allows access to any system or person who can circumvent the authentication.

In a classic operating system flaw, the buffer for typed characters in a password was of fixed size, counting all characters typed, including backspaces for correction. If a user typed more characters than the buffer would hold, the overflow caused the

operating system to bypass password comparison and act as if a correct authentication had been supplied. These flaws can be exploited by anyone seeking access.

Many network hosts, especially those that connect to wide area networks, run variants of Unix System V or BSD Unix. In a local environment, many users are not aware of which networked operating system is in use; still fewer would know of, be capable of, or be interested in exploiting flaws. However, some hackers regularly scan wide area networks for hosts running weak or flawed operating systems. Thus, connection to a wide area network, especially the Internet, exposes these flaws to a wide audience intent on exploiting them.

3.5.4 Nonexistent Authentication

If two computers are used by the same users to store data and run processes and if each has authenticated its users on first access, one may assume that computer-to-computer or local user-to-remote process authentication is unnecessary. These two computers and their users are a trustworthy environment in which the added complexity of repeated authentication seems excessive.

However, this assumption is not valid. In Unix, the file .rhosts lists trusted hosts and .rlogin lists trusted users who are allowed access without authentication. The files are intended to support computer-to-computer connection by users who have already been authenticated at their primary hosts. These "trusted hosts" can also be exploited by outsiders who obtain access to one system through an authentication weakness (such as a password guessing) and then transfer to another system that accepts the authenticity of a user who comes from a system on its trusted list.

An attacker may also realize that a system has some identities requiring no authentication. Some systems have "guest" or "anonymous" accounts to allow outsiders to access things the systems want to release to anyone. A user can log in as "guest" and retrieve publicly available items. Typically, no password is required, or the user is shown a message requesting that the user type "GUEST" when asked for a password. Each of these accounts allows access to unauthenticated users.

3.5.5 Well-Known Authentication

Authentication data should be unique and difficult to guess. But unfortunately, the convenience of one, well-known authentication scheme sometimes usurps the protection. The system network management protocol (SNMP) is widely used for remote management of network devices, such as routers and switches that support no ordinary users. SNMP uses a "community string," essentially a password for the community of devices that can interact with one another. But network devices are designed especially for quick installation with minimal configuration, and many network administrators do not change the default community string installed on a router or switch. This laxity makes these devices on the network perimeter open to many SNMP attacks.

Some vendors still ship computers with one system administration account installed, having a default password. Or the systems come with a demonstration or test account, with no required password. Some administrators fail to change the passwords or delete these accounts.

3.5.6 Trusted Authentication

Finally, authentication can become a problem when identification is delegated to other trusted sources. For instance, a file may indicate who can be trusted on a particular host. Or the authentication mechanism for one system can "vouch for" a user. Files indicate hosts or users that are trusted on other hosts are useful to users who have accounts on multiple machines or for network management, maintenance, and operation, they must be used very carefully. Each of them represents a potential hole through which a remote user—or a remote attacker—can achieve access.

3.6 Spoofing

Guessing or otherwise obtaining the network authentication credentials of an entity (a user, an account, a process, a node, a device) permits an attacker to create a full communication under the entity's identity. Impersonation falsely represents a valid entity in a communication. Closely related is spoofing, when an attacker falsely carries on one end of a networked interchange. Examples of spoofing are masquerading, session hijacking, man-in-the-middle attacks, and IP spoofing.

3.6.1 Masquerade

In a masquerade one host pretends to be another. A common example is URL confusion. Domain names can easily be confused, or someone can easily mistype certain names. Thus xyz.com, xyz.org, and xyz.net might be three different organizations, or one bona fide organization (for example, xyz.com) and two masquerade attempts from someone who registered the similar domain names. Names with or without hyphens (coca-cola.com versus cocacola.com) and easily mistyped

names (l0pht.com versus lopht.com, or citibank.com versus citybank.com) are candidates for masquerading.

In another version of a masquerade, the attacker exploits a flaw in the victim's web server and is able to overwrite the victim's web pages. Although there is some public humiliation at having one's site replaced, perhaps with obscenities or strong messages opposing the nature of the site (for example, a plea for vegetarianism on a slaughterhouse web site), most people would not be fooled by a site displaying a message absolutely contrary to its aims. However, a clever attacker can be more subtle. Instead of differentiating from the real site, the attacker can try to build a false site that resembles the real one, perhaps to obtain sensitive information (names, authentication numbers, credit card numbers) or to induce the user to enter into a real transaction. For example, if one bookseller's site, call it Books-R-Us, were overtaken subtly by another, called Books Depot, the orders may actually be processed, filled, and billed to the naive users by Books Depot.

3.6.2 Session Hijacking

Session hijacking is intercepting and carrying on a session begun by another entity. Suppose two entities have entered into a session but then a third entity intercepts the traffic and carries on the session in the name of the other. 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:

- A trusted client connects to a network server.
- The attack computer gains control of the trusted client.

- The attack computer disconnects the trusted client from the network server.
- The attack computer replaces the trusted client's IP address with its own IP address and spoofs the client's sequence numbers.
- The attack computer continues dialog with the network server (and the network server believes it is still communicating with trusted client).

3.6.3 Man-in-the-Middle Attack

A man-in-the-middle attack is a similar form of session hijacking, in which one entity intrudes between two others. The difference between man-in-the-middle and hijacking is that a man-in-the-middle usually participates from the start of the session, whereas a session hijacking occurs after a session has been established. The difference is largely semantic and not too significant.

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.

3.6.4 IP Spoofing

IP spoofing is used by an intruder to convince a system that it is communicating with a known, trusted entity to provide the intruder with access to the system. IP spoofing involves an alteration of a packet at the TCP level, which is used to attack Internet-connected systems that provide various TCP/IP services. In this exploit, the attacker sends a packet with an IP source address of a known, trusted host instead of

its own IP source address to a target host. The target host may accept the packet and act upon it.

3.7 Message Confidentiality Threats

An attacker can easily violate message confidentiality (and perhaps integrity) because of the public nature of networks. Eavesdropping and impersonation attacks can lead to a confidentiality or integrity failure. There are several other vulnerabilities that can affect confidentiality.

3.7.1 Misdelivery

Sometimes messages are misdelivered because of some flaw in the network hardware or software. Most frequently, messages are lost entirely, which is an integrity or availability issue. Occasionally, however, a destination address will be modified or some handler will malfunction, causing a message to be delivered to someone other than the intended recipient. All of these "random" events are quite uncommon.

More frequent than network flaws are human errors. It is far too easy to mistype an address such as 100064,30652 as 10064,30652 or 100065,30642, or to type "idw" or "iw" instead of "diw" for David Ian Walker, who is called Ian by his friends. There is simply no justification for a computer network administrator to identify people by meaningless long numbers or cryptic initials when "iwalker" would be far less prone to human error.

3.7.2 Exposure

To protect the confidentiality of a message, one must track it all the way from its creation to its disposal. Along the way, the content of a message may be exposed in temporary buffers; at switches, routers, gateways, and intermediate hosts throughout the network; and in the workspaces of processes that build, format, and present the message. All of these exposures apply to networked environments as well. Furthermore, a malicious attacker can use any of these exposures as part of a general or focused attack on message confidentiality.

Passive wiretapping is one source of message exposure. So also is subversion of the structure by which a communication is routed to its destination. Finally, intercepting the message at it source, destination, or at any intermediate node can lead to its exposure.

3.7.3 Traffic Flow Analysis

Sometimes not only is the message itself sensitive but the fact that a message exists is also sensitive. For example, if the enemy during wartime sees a large amount of network traffic between headquarters and a particular unit, the enemy may be able to infer that significant action is being planned involving that unit. In a commercial setting, messages sent from the president of one company to the president of a competitor could lead to speculation about a takeover or conspiracy to fix prices. Or communications from the prime minister of one country to another with whom diplomatic relations were suspended could lead to inferences about a rapprochement between the countries.

3.8 Message Integrity Threats

In many cases, the integrity or correctness of a communication is at least as important as its confidentiality. In fact for some situations, such as passing authentication data, the integrity of the communication is paramount. In other cases, the need for integrity is less obvious.

3.8.1 Falsification of Messages

Increasingly, people depend on electronic messages to justify and direct actions. An attacker can take advantage of users trust in messages to mislead users. In particular, an attacker may

- Change some or all of the content of a message
- Replace a message entirely, including the date, time, and sender/receiver identification
- Reuse (replay) an old message
- Combine pieces of different messages into one
- Change the apparent source of a message
- Redirect a message
- Destroy or delete a message

These attacks can be perpetrated in the ways we have already examined, including:

- Active wiretap
- Trojan horse
- Impersonation
- Preempted host
- Preempted workstation

3.8.2 Noise

Signals sent over communications media are subject to interference from other traffic on the same media, as well as from natural sources, such as lightning, electric motors, and animals. Such unintentional interference is called noise. These forms of noise are inevitable, and they can threaten the integrity of data in a message.

Fortunately, communications protocols have been intentionally designed to overcome the negative effects of noise. For example, the TCP/IP protocol suite ensures detection of almost all transmission errors. Processes in the communications stack detect errors and arrange for retransmission, all invisible to the higher-level applications. Thus, noise is scarcely a consideration for users in security-critical applications.

3.8.3 Web Site Defacement

One of the most widely known attacks is the web site defacement attack. Because of the large number of sites that have been defaced and the visibility of the result, the attacks are often reported in the popular press.

Defacement is common not only because of its visibility but also because of the ease with which one can be done. Web sites are designed so that their code is downloaded, enabling an attacker to obtain the full hypertext document and all programs directed to the client in the loading process. An attacker can even view programmers' comments left in as they built or maintained the code. The download process essentially gives the attacker the blueprints to the web site.

The web site vulnerabilities enable attacks known as buffer overflows, dot-dot problems, application code errors, and server-side include problems.

3.8.3.1 Buffer Overflows

Buffer overflow is on web pages. The attacker simply feeds a program far more data than it expects to receive. A buffer size is exceeded, and the excess data spill over into adjoining code and data locations.

Perhaps the best-known web server buffer overflow is the file name problem known as iishack. This attack is so well known that is has been written into a procedure. To execute the procedure, an attacker supplies as parameters the site to be attacked and the URL of a program the attacker wants that server to execute.

Other web servers are vulnerable to extremely long parameter fields, such as passwords of length 10,000 or a long URL padded with space or null characters.

3.8.3.2 Dot-Dot and Address Problems

Web server code should always run in a constrained environment. Ideally, the web server should never have editors, xterm and Telnet programs, or even most system utilities loaded. By constraining the environment in this way, even if an attacker escapes from the web server application, no other executable programs will help the attacker use the web server's computer and operating system to extend the attack. The code and data for web applications can be transferred manually to a web server or pushed as a raw image.

But many web applications programmers are naive. They expect to need to edit a web application in place, so they expect to need editors and system utilities to give them a complete environment in which to program.

A second, less desirable, condition for preventing an attack is to create a fence confining the web server application. With such a fence, the server application cannot escape from its area and access other potentially dangerous system areas (such as

editors and utilities). The server begins in a particular directory subtree, and everything the server needs is in that same subtree.

3.8.3.3 Application Code Errors

A user's browser carries on an intricate, undocumented protocol interchange with the web server. To make its job easier, the web server passes context strings to the user, making the user's browser reply with full context. A problem arises when the user can modify that context.

Consider a online CD store, selling compact disks. At any given time, a server at that site may have a thousand or more transactions in various states of completion. The site displays a page of goods to order, the user selects one, the site displays more items, the user selects another, the site displays more items, the user selects two more, and so on until the user is finished selecting. Many people go on to complete the order by specifying payment and shipping information. But other people use web sites like this one as an online catalog or guide, with no real intention of ordering. If the user is a bona fide customer, sometimes web connections fail, leaving the transaction incomplete. For these reasons, the web server often keeps track of the status of an incomplete order in parameter fields appended to the URL. These fields travel from the server to the browser and back to the server with each user selection or page request.

Assume one has selected a CD and are looking at a second web page. The web server has passed the customer a URL similar to http://www.CDs-r-us.com/page4&i1=459012&p1=1599. This URL means the customer has chosen CD number 459012, and its price is $15.99. He now selects a second and the URL becomes http://www.CDs-r-us.com/page7&i1=459012&p1=1599&i2=365217&p2=1499. But if the customer is a clever attacker, he realizes that he can edit the URL in the address

window of his browser. Consequently, he changes each of 1599 and 1499 to 199. And when the server totals up his order, lo and behold, his two CDs cost only $1.99 each.

This failure is an example of the time-of-check to time-of-use flaw. The server sets (checks) the price of the item when you first display the price, but then it loses control of the checked data item and never checks it again. This situation arises frequently in server application code because application programmers are generally not aware of security and typically do not anticipate malicious behavior.

3.8.3.4 Server-Side Include

A potentially more serious problem is called a server-side include. The problem takes advantage of the fact that web pages can be organized to invoke a particular function automatically. For example, many pages use web commands to send an e-mail message in the "contact us" part of the displayed page. The commands, such as e-mail, if, go to, and include, are placed in a field that is interpreted in HTML.

3.9 Denial of Service

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 effected 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) attack, which is also an attack on a network's resources, but is launched from a large number of other host machines. Attack software is installed on these host computers, unknown to their owners, and then activated simultaneously to launch communications to the target machine of such magnitude as to overwhelm the target machine.

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.

3.9.1 Transmission Failure

Communications fail for many reasons. For instance, a line is cut. Or network noise makes a packet unrecognizable or undeliverable. A machine along the transmission path fails for hardware or software reasons. A device is removed from service for repair or testing. A device is saturated and rejects incoming data until it can clear its overload. Many of these problems are temporary or automatically fixed (circumvented) in major networks, including the Internet.

However, some failures cannot be easily repaired. A break in the single communications line to your computer (for example, from the network to your network interface card or the telephone line to your modem) can be fixed only by establishment of an alternative link or repair of the damaged one. The network administrator will say "service to the rest of the network was unaffected," but that is of little consolation to you.

3.9.2 Connection Flooding

The most primitive denial-of-service attack is flooding a connection. If an attacker sends as much data as one's communications system can handle, user is prevented from receiving any other data. Even if an occasional packet reaches to user from someone else, communication with user will be seriously degraded.

More sophisticated attacks use elements of Internet protocols. In addition to TCP and UDP, there is a third class of protocols, called ICMP or Internet Control Message Protocols. Normally used for system diagnostics, these protocols do not have associated user applications. ICMP protocols include:

- **Ping**, which requests a destination to return a reply, intended to show that the destination system is reachable and functioning
- **Echo**, which requests a destination to return the data sent to it, intended to show that the connection link is reliable (ping is actually a version of echo)
- **Destination unreachable**, which indicates that a destination address cannot be accessed
- **Source quench**, which means that the destination is becoming saturated and the source should suspend sending packets for a while

These protocols have important uses for network management. But they can also be used to attack a system. The protocols are handled within the network stack, so that acts may be difficult to detect or block on the receiving host.

3.9.2.1 Echo-Chargen

This attack works between two hosts. Chargen is a protocol that generates a stream of packets; it is used to test the network's capacity. The attacker sets up a chargen process on host A that generates its packets as echo packets with a destination of host B. Then, host A produces a stream of packets to which host B replies by echoing them back to host A. This series puts the network infrastructures of A and B into an endless loop. If the attacker makes B both the source and destination address of the first packet, B hangs in a loop, constantly creating and replying to its own messages.

3.9.2.2 Ping of Death

A ping of death is a simple attack. Since ping requires the recipient to respond to the ping request, all the attacker needs to do is send a flood of pings to the intended victim. The attack is limited by the smallest bandwidth on the attack route. If the attacker is on a 10-megabyte (MB) connection and the path to the victim is 100 MB or more, the attacker cannot mathematically flood the victim alone. But the attack succeeds if the numbers are reversed: The attacker on a 100-MB connection can easily flood a 10-MB victim. The ping packets will saturate the victim's bandwidth.

3.9.3 Smurf

The smurf attack is a variation of a ping attack. It uses the same vehicle, a ping packet, with two extra twists. First, the attacker chooses a network of unwitting victims. The attacker spoofs the source address in the ping packet so that it appears to come from the victim. Then, the attacker sends this request to the network in broadcast mode

by setting the last byte of the address to all 1s; broadcast mode packets are distributed to all hosts on the network.

3.9.4 Script Kiddies

Attacks can be scripted. A simple smurf denial-of-service attack is not hard to implement. But an underground establishment has written scripts for many of the popular attacks. With a script, attackers need not understand the nature of the attack not even the concept of a network. The attackers merely download the attack script (no more difficult than downloading a newspaper story from a list of headlines) and execute it. The script takes care of selecting an appropriate (that is, vulnerable) victim and launching the attack.

People who download and run attack scripts are called script kiddies. As the rather derogatory name implies, script kiddies are not well respected in the attacker community because the damage they do requires almost no creativity or innovation. Nevertheless, script kiddies can cause serious damage, sometimes without even knowing what they do.

3.9.5 Syn Flood

Another popular denial-of-service attack is the syn flood. This attack uses the TCP protocol suite, making the session-oriented nature of these protocols work against the victim.

For a protocol such as Telnet, the protocol peers establish a virtual connection, called a session, to synchronize the back-and-forth, command-response nature of the Telnet terminal emulation. A session is established with a three-way TCP handshake. Each TCP packet has flag bits, two of which are denoted SYN and ACK. To initiate a

TCP connection, the originator sends a packet with the SYN bit on. If the recipient is ready to establish a connection, it replies with a packet with both the SYN and ACK bits on. The first party then completes the exchange to demonstrate a clear and complete communication channel by sending a packet with the ACK bit on, as shown in Figure.

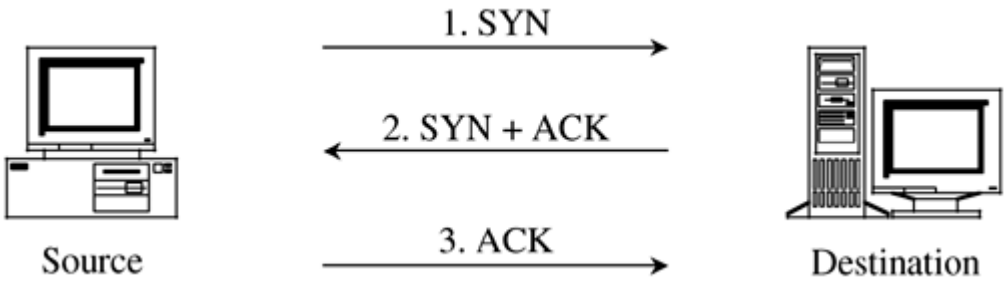


Figure1: Three-Way Connection Handshake (from "Security in Computing")

Occasionally packets get lost or damaged in transmission. The destination maintains a queue called the SYN_RECV connections, tracking those items for which a SYN–ACK has been sent but no corresponding ACK has yet been received. Normally, these connections are completed in a short time. If the SYN–ACK (2) or the ACK (3) packet is lost, eventually the destination host will time out the incomplete connection and discard it from its waiting queue.

The attacker can deny service to the target by sending many SYN requests and never responding with ACKs, thereby filling the victim's SYN_RECV queue. Typically, the SYN_RECV queue is quite small, such as 10 or 20 entries. Because of potential routing delays in the Internet, typical holding times for the SYN_RECV queue can be minutes. So the attacker needs only to send a new SYN request every few seconds and it will fill the queue.

Attackers using this approach usually do one more thing: they spoof the nonexistent return address in the initial SYN packet for two reasons.

- First, the attacker does not want to disclose the real source address in case someone should inspect the packets in the SYN_RECV queue to try to identify the attacker.
- Second, the attacker wants to make the SYN packets indistinguishable from legitimate SYN packets to establish real connections.

Choosing a different (spoofed) source address for each one makes them unique. A SYN–ACK packet to a nonexistent address will result in an ICMP Destination Unreachable result, but this will not be the ACK for which the TCP connection is waiting.

3.9.5 Traffic Redirection

A router is a device that forwards traffic on its way through intermediate networks between a source host's network and a destination's. So if an attacker can corrupt the routing, traffic can disappear.

Routers use complex algorithms to decide how to route traffic. No matter the algorithm, they essentially seek the best path (where "best" is measured in some combination of distance, time, cost, quality, and the like). Routers are aware only of the routers with which they share a direct network connection, and they use gateway protocols to share information about their capabilities. Each router advises its neighbors about how well it can reach other network addresses. This characteristic allows an attacker to disrupt the network.

In spite of its sophistication, a router is simply a computer with two or more network interfaces. Suppose a router advertises to its neighbors that it has the best

path to every other address in the whole network. Soon all routers will direct all traffic to that one router. The one router may become flooded, or it may simply drop much of its traffic. In either case, a lot of traffic never makes it to the intended destination.

3.9.6 DNS Attack

A domain name server (DNS) is a table that converts domain names like ATT.COM into network addresses like 211.217.74.130; this process is called resolving the domain name. A domain name server queries other name servers to resolve domain names it does not know. For efficiency, it caches the answers it receives so it can resolve that name more rapidly in the future.

In the most common implementations of Unix, name servers run software called Berkeley Internet Name Domain or BIND or named (a shorthand for "name daemon"). There have been numerous flaws in BIND, including the now-familiar buffer overflow.

By overtaking a name server or causing it to cache spurious entries, an attacker can redirect the routing of any traffic, with an obvious implication for denial of service.

3.9.7 Distributed Denial of Service

To perpetrate a distributed denial-of-service (or DDoS) attack, an attacker does two things. In the first stage, the attacker uses any convenient attack (such as exploiting a buffer overflow or tricking the victim to open and install unknown code from an e-mail attachment) to plant a Trojan horse on a target machine. That Trojan horse does not necessarily cause any harm to the target machine, so it may not be noticed. The Trojan horse file may be named for a popular editor or utility, bound to a standard operating system service, or entered into the list of processes (daemons) activated at

startup. No matter how it is situated within the system, it will probably not attract any attention.

The attacker repeats this process with many targets. Each of these target systems then becomes what is known as a zombie. The target systems carry out their normal work, unaware of the resident zombie; these zombies can then be remotely controlled by a master. Each zombie could generate thousands of requests of a server, with hundreds of zombies; millions of packets can be generated. With enough zombies, even the biggest web sites or Internet pipes can be filled.

At some point the attacker chooses a victim and sends a signal to all the zombies to launch the attack. Then, instead of the victim's trying to defend against one denial-of-service attack from one malicious host, the victim must try to counter n attacks from the n zombies all acting at once. Not all of the zombies need to use the same attack; for instance, some can use smurf attacks and others syn floods to address different potential weaknesses.

In addition to their tremendous multiplying effect, distributed denial-of-service attacks are a serious problem because they are easily launched from scripts. Given a collection of denial-of-service attacks and a Trojan horse propagation method, one can easily write a procedure to plant a Trojan horse that can launch any or all of the denial-of-service attacks.

DDOS attacks work by using remotely controlled computers to generate more requests of a device than it can serve. The attackers gain access to machines and install a zombie client upon them

3.9.8 Land Attack

A land attack involves sending a spoofing TCP SYN packet (connection initiation) with the target host's IP address with an open port as both source and destination. The attack causes the targeted machine to reply to itself continuously and eventually crash.

3.9.9 ICMP Flood

A smurf attack is one particular variant of a flooding DoS attack on the public Internet. It relies on mis-configured network devices that allow packets to be sent to all computer hosts on a particular network via the broadcast address of the network, rather than a specific machine. The network then serves as a smurf amplifier. In such an attack, the perpetrators will send large numbers of IP packets with the source address faked to appear to be the address of the victim. To combat Denial of Service attacks on the Internet, services like the Smurf Amplifier Registry have given network service providers the ability to identify misconfigured networks and to take appropriate action such as filtering.

Ping flood is based on sending the victim an overwhelming number of ping packets, usually using the "ping -f" command. It is very simple to launch, the primary requirement being access to greater bandwidth than the victim.

3.9.10 UDP Flood

UDP floods include "Fraggle attacks". In a fraggle attack an attacker sends a large amount of UDP echo traffic to IP broadcast addresses, all of it having a fake source address. It is a simple rewrite of the smurf attack code.

3.9.11 Teardrop Attack

The Teardrop attack involves sending IP fragments with overlapping oversized payloads to the target machine. A bug in the TCP/IP fragmentation re-assembly code caused the fragments to be improperly handled, crashing the operating system as a result. Windows 3.1x, Windows 95 and Windows NT operating systems, as well as versions of Linux prior to 2.0.32 and 2.1.63 are vulnerable to this attack.

3.10 Threats to Active or Mobile Code

Active code or mobile code is a general name for code that is pushed to the client for execution. For example, suppose an internet user wants his web site to have bears dancing across the top of the page. To download the dancing bears, he could download a new image for each movement the bears take: one bit forward, two bits forward, and so forth. However, this approach uses far too much server time and bandwidth to compute the positions and download new images. A more efficient use of (server) resources is to download a program that runs on the client's machine and implements the movement of the bears.

This mean a site user doesn’t control, which could easily be hacked by attackers or crackers, is going to push code to his machine that will execute without his knowledge, permission, or oversight. In fact, there are many different kinds of active code.

3.10.1 Cookies

Cookies are not active code. They are data files that can be stored and fetched by a remote server. However, cookies can be used to cause unexpected data transfer from a client to a server, so they have a role in a loss of confidentiality.

A cookie is a data object that can be held in memory (a per-session cookie) or stored on disk for future access (a persistent cookie). Cookies can store anything about a client that the browser can determine: keystrokes the user types, the machine name, connection details (such as IP address), date and type, and so forth. On command a browser will send to a server the cookies saved for it. Per-session cookies are deleted when the browser is closed, but persistent cookies are retained until a set expiration date, which can be years in the future.

Cookies provide context to a server. Using cookies, certain web pages can greet one with "Welcome back, Sarwar Zaman" or reflect one's preferences, as in "Shall I ship this order to you at 135 Elm Street?" But as these two examples demonstrate, anyone possessing someone's cookie becomes that person in some contexts. Thus, anyone intercepting or retrieving a cookie can impersonate the cookie's owner.

3.10.2 Scripts

Clients can invoke services by executing scripts on servers. Typically, a web browser displays a page. As the user interacts with the web site via the browser, the browser organizes user inputs into parameters to a defined script; it then sends the script and parameters to a server to be executed. But all communication is done through HTML. The server cannot distinguish between commands generated from a user at a browser completing a web page, and a user's handcrafting a set of orders. The malicious user can monitor the communication between a browser and a server to see how changing a web page entry affects what the browser sends and then how the server reacts. With this knowledge, the malicious user can manipulate the server's actions.

To see how easily this manipulation is done, remember that programmers do not often anticipate malicious behavior; instead, programmers assume that users will be benign and will use a program in the way it was intended to be used. For this reason, programmers neglect to filter script parameters to ensure that they are reasonable for the operation and safe to execute. Some scripts allow arbitrary files to be included or arbitrary commands to be executed. An attacker can see the files or commands in a string and experiment with changing them.

A well-known attack against web servers is the escape-character attack. A common scripting language for web servers, CGI, defines a machine-independent way to encode communicated data. Coding convention uses %nn to represent ASCII special characters. However, special characters may be interpreted by CGI script interpreters. So, for example, %0A (end-of-line) instructs the interpreter to accept the following characters as a new command. The following command requests a copy of the server's password file: http://www.test.com/cgibin/query?%0a/bin/cat%20/etc/passwd. CGI scripts can initiate actions directly on the server. An attacker can observe a CGI script that includes a string of this form: <!-#action arg1 = value arg2=value ...> and submit a subsequent command where the string is replaced by <!-#exec cmd = "rm *"> to cause a command shell to execute a command to remove all files in shell's current directory.

Microsoft uses active server pages (ASP) as its scripting capability. Such pages instruct the browser on how to display files, maintain context, and interact with the server. These pages can also be viewed at the browser end, so any programming weaknesses in the ASP code are available for inspection and attack.

The server should never trust anything received from a client, because the remote user can send the server a string crafted by hand, instead of one generated by a benign procedure the server sent the client. As with so many cases of remote access, these examples demonstrate that if user allows someone else to run a program on his machine, he can no longer have confidence that his machine is secure.

3.10.3 Active Code

Displaying web pages started simply with a few steps: generate text, insert images, and register mouse clicks to fetch new pages. Soon, people wanted more elaborate action at their web sites: toddlers dancing atop the page, a three-dimensional rotating cube, images flashing on and off, colors changing, totals appearing. Some of these tricks, especially those involving movement, take significant computing power; they require a lot of time and communication to download from a server. But typically, the client has a capable and underutilized processor, so the timing issues are irrelevant.

To take advantage of the processor's power, the server may download code to be executed on client. This executable code is called active code. Two main kinds of active code are JavaScript and ActiveX controls.

3.10.3.1 JavaScript

Sun Microsystems designed and promoted Java as a truly machine-independent programming language. A Java program consists of Java bytecode executed on a Java virtual machine. The bytecode programs are machine independent, and only the JVM needs to be implemented on each class of machine to achieve program portability. The

JVM contains a built-in security manager that enforces a security policy. A Java program runs in a Java "sandbox," a constrained resource domain from which the program cannot escape. The Java programming language is strongly typed, meaning that the content of a data item must be of the appropriate type for which it is to be used (for example, a text string cannot be used as a numeric).

The original specification, called Java 1.1, was very solid, very restrictive, and hence very unpopular. In it, a program could not write permanently to disk, nor could it invoke arbitrary procedures that had not been included in the sandbox by the security manager's policy. Thus, the sandbox was a collection of resources the user was willing to sacrifice to the uncertainties of Java code. Although very strong, the Java 1.1 definition proved unworkable. As a result, the original restrictions on the sandbox were relaxed, to the detriment of security.

The Java 1.2 specification opened the sandbox to more resources, particularly to stored disk files and executable procedures. Although it is still difficult to break its constraints, the Java sandbox contains many new toys, enabling more interesting computation but opening the door to exploitation of more serious vulnerabilities.

Does this mean that Java's designers made bad decisions? No. A product's security flaw is not necessarily a design flaw. Sometimes the designers choose to trade some security for increased functionality or ease of use. In other cases, the design is fine, but implementers fail to uphold the high security standards set out by designers. The latter is certainly true for Java. There have been problems with implementations of Java virtual machines for different platforms and in different components. For example, a version of Netscape browser failed to implement type checking on all data types, as is required in the Java specifications. A similar vulnerability affected Microsoft Internet

Explorer. Although these vulnerabilities have been patched, other problems could occur with subsequent releases.

A hostile applet is downloadable Java code that can cause harm on the client's system. Because an applet is not screened for safety when it is downloaded and because it typically runs with the privileges of its invoking user, a hostile applet can cause serious damage.

3.10.3.2 ActiveX

Microsoft's answer to Java technology is ActiveX. Using ActiveX, objects of arbitrary type can be downloaded to a client. If the client has a viewer or handler for the object's type, that viewer is invoked to present the object. For example, downloading a Microsoft Word .doc file would invoke Microsoft Word on a system on which it is installed. Files for which the client has no handler cause other code to be downloaded. Thus, in theory, an attacker could invent a type, called .bomb, and cause any unsuspecting user who downloaded a web page with a .bomb file also to download code that would execute .bombs.

To prevent arbitrary downloads, Microsoft uses an authentication scheme under which downloaded code is cryptographically signed and the signature is verified before execution. But the authentication verifies only the source of the code, not its correctness or safety. Code from Microsoft (or Netscape or any other manufacturer) is not inherently safe, and code from an unknown source may be more or less safe than that from a known source. Proof of origin shows where it came from, not how good or safe it is. And some vulnerabilities allow ActiveX to bypass the authentication.

3.10.4 Auto Exec by Type

Data files are processed by programs. For some products, the file type is implied by the file extension, such as .doc for a Word document, .pdf (Portable Document Format) for an Adobe Acrobat file, or .exe for an executable file. On many systems, when a file arrives with one of these extensions, the operating system automatically invokes the appropriate processor to handle it.

By itself, a Word document is unintelligible as an executable file. To prevent someone from running a file temp.doc by typing that name as a command, Microsoft embeds in a file what type it really is. Double clicking the file in a Windows Explorer window brings up the appropriate program to handle that file.

This scheme presents an opportunity to an attacker. A malicious agent may send a file named innocuous.doc, which would expect to be a Word document. Because of the .doc extension, Word would try to open it. Suppose that file is renamed "innocuous" (without a .doc). If the embedded file type is .doc, then double-clicking innocuous also brings the file up in Word. The file might contain malicious macros or invoke the opening of another, more dangerous file.

Generally, we recognize that executable files can be dangerous, text files are likely to be safe, and files with some active content, such as .doc files, fall in between. If a file has no apparent file type and will be opened by its built-in file handler, we are treading on dangerous ground. An attacker can disguise a malicious active file under a non-obvious file type.

3.11 Building Blocks

An attacker simply out to cause minor damage to a randomly selected site could use any of the techniques have described above, perhaps under script control. A dedicated attacker who targets one location can put together several pieces of an attack in order to compound the damage. Often, the attacks are done in series so that each part builds on the information gleaned from previous attacks. For example, a wiretapping attack may yield reconnaissance information with which to form an ActiveX attack that transfers a Trojan horse that monitors for sensitive data in transmission. Putting the attack pieces together like building blocks expands the number of targets and increases the degree of damage.

3.12 Weak keys

For many cryptographic algorithms, some keys are weaker than others (that is, some keys are not as secure as other keys). Strong keys are generated using truly random number generators. For specific algorithms, keys can be tested for their strength. For example, the data encryption standard, DES, has only 16 weak keys out of its 256 possible keys. Because weak keys for an algorithm can be identified, they should not be used.

When an algorithm has keys that are all of equal strength, it is said to have a linear or flat key space. Conversely, if an algorithm has keys that are not all of equal strength, it has a nonlinear key space. The same use of randomness applies to passwords in that the more random the choice of letters and characters in a password, the more secure the password is. However, the more random the sequence of letters and characters in a password, the more difficult it is for a person to remember.

3.13 Mathematical Attacks

Mathematical attacks refer to the use of mathematics to break passwords or cryptographic algorithms as opposed to other approaches, such as brute force, which try all possible combinations of patterns.

A good example of a mathematical attack is the use of factoring algorithms to break the RSA public key cryptography algorithm. Recall that the hard problem in RSA is determining the prime factors of a large number. Numbers on the order of 129 digits have been factored using factoring algorithms and thousands of computers on the Internet. One of the better factoring algorithms is the number field sieve (NFS).

3.14 Birthday Attacks

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, $H(M1) \neq H(M2)$).
- 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.
- It should be computationally infeasible to find a message that corresponds to a given message digest.

- The message digest should be calculated using all of the data in the original message.

3.15 War Driving

In war driving or walking, an attacker scans for 802.11-based wireless network information by using a laptop computer with wireless adapter in promiscuous mode and scanning software such as NetStumbler. Also, a Global Positioning System (GPS) might be used to note the location of compromised nodes.

3.16 War Dialing/Demon Dialing Attack

In war dialing, an attacker uses a program that automatically places calls to a group of telephone numbers in hopes of finding numbers that are connected to modems. In demon dialing, a brute-force, password-guessing approach is used to gain access to a system through a modem.

3.17 Replay

A replay attack occurs when an attacker intercepts and saves old messages and then tries to send them later, impersonating one of the participants. One method of making this attack more difficult to accomplish is through the use of a random number or string, called a nonce, that changes with time. If Bob wants to communicate with Alice, he sends a nonce along with the first message to Alice. When Alice replies, she sends the nonce back to Bob, who verifies that it is the one he sent with the first message. Anyone trying to use these same messages later will not be using the newer nonce. Another approach to countering the replay attack is for Bob to add a timestamp to his message. This timestamp indicates the time that the message was sent. Thus, if the message is used later, the timestamp will show that an old message is being used.

Top 14 network vulnerabilities

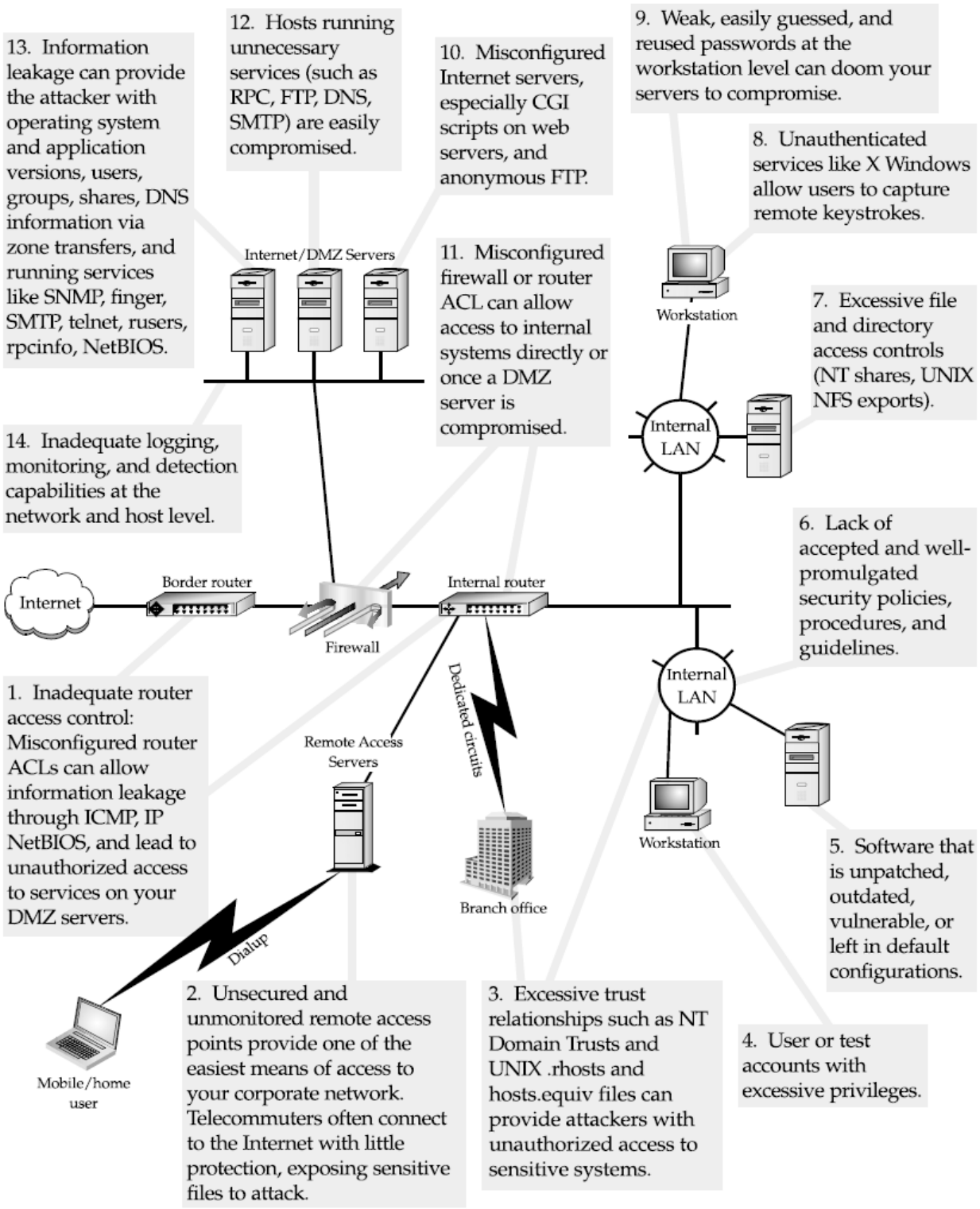


Figure 2: Top 14 network attacks (from “Network Security Bible”)

Summary of Network Vulnerabilities

A network has many different vulnerabilities, but all derive from an underlying model of computer, communications, and information systems security. Threats are raised against the key aspects of security: confidentiality, integrity, and availability, as shown in table.

Table 1: Network Vulnerabilities (from “Security in Computing”)

Target	Vulnerability
Precursors to attack	Port scan Social engineering Reconnaissance OS and application fingerprinting
Authentication failures	Impersonation Guessing Eavesdropping Spoofing Session hijacking Man-in-the-middle attack
	Buffer overflow

Programming flaws	Addressing errors Parameter modification, time-of-check to time-of-use errors Server-side include Cookie Malicious active code: JavaScript, ActiveX Malicious code: virus, worm, Trojan horse Malicious typed code
Confidentiality	Protocol flaw Eavesdropping Passive wiretap Misdelivery Exposure within the network Traffic flow analysis Cookie
Integrity	Protocol flaw Active wiretap Impersonation Falsification of message

	Noise Web site defacement DNS attack
Availability	Protocol flaw Transmission or component failure Connection flooding, e.g., echo-chargen, ping of death, smurf, syn flood DNS attack Traffic redirection Distributed denial of service

Chapter 4 Developing our own Intrusion Detection System

Our study in the thesis was oriented towards the creation of an implementation of an Intrusion Detection System. Our plan was to develop a Network Based Intrusion Detection System so as to give focus on the networking related aspects of the IDS discipline. We targeted Linux as the environment to program as most of the web-servers that run today run on Linux based machines. Having searched and analyzed the way to go, we came to the conclusion that the best language for the implementation would be C/C++ due to the deep system level access that it provides that is mandatory to make an Interface card sniff all packets within the network collision domain that it is connected to, which is needed for the implementation.

4.1 Challenges

The challenges for creating an Intrusion Detection System was first to capture a packets and analyze them for discrepancies to make a packet level analysis, that is to say a network layer analysis of message stream that flows through a point in a network. This is because remote manipulation of a system will definitely consist of network traffic, by which a harmful user may try to communicate with a computer in trying to make it malfunction and hopefully give unsolicited access to inside the network of that computer.

The next challenge was to capture packets from their datalink layer headers. To ensure deep level analysis of packet and better intrusion detection. And also to capture packets from all the hosts connected to the computer's collision domain. This way the IDS will work as a **network sniffer** for a network and scrutinize packets not meant for

the host it is running upon, making the implementation a Network Based Intrusion Detection System.

4.2 Type of Implementation

As mentioned above, the implementation is of type NIDS or Network Based Intrusion Detection System.

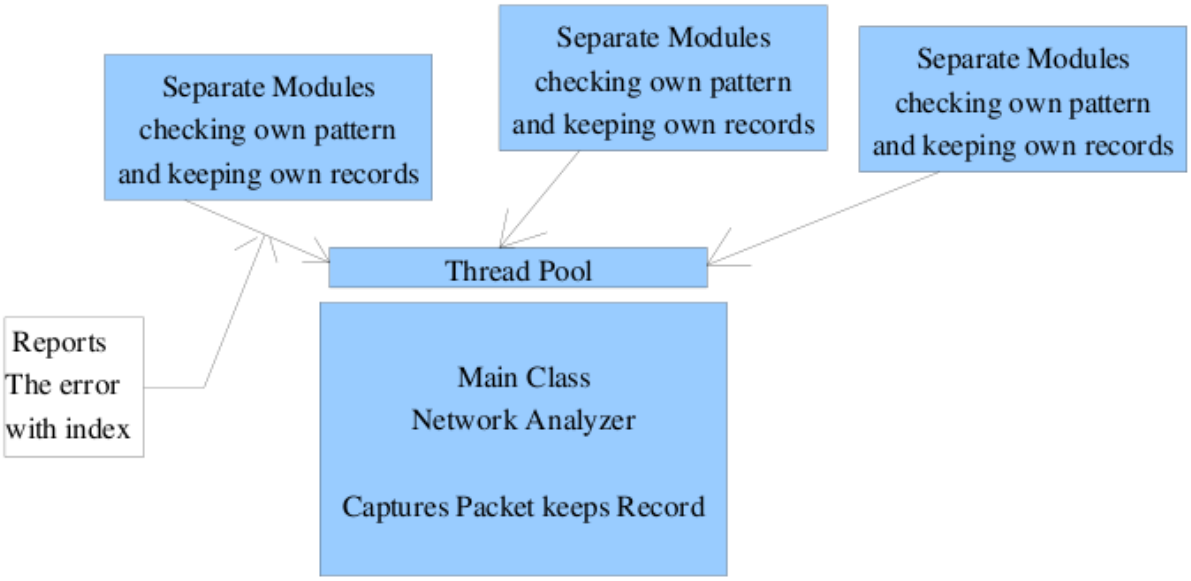
4.3 Planning

1. There will be a central class which will work as the Network Analyzer which will do the bulk of the capturing and recording process
    - It will classify the packets with it's known masks intelligently among
        ** ARP
        ** ICMP
        ** TCP
        ** UDP
        ** IP
        ** OTHERS (not checked specifically)
    - Keep a File based Log of both the "Counts of Packets" and a "HeaderLog" to actually store the packets that came to the daily traffic
        ** There should be a basis by time to how long the records can be kept
    - Keep a master Analysis record, (Array) of current packets , a central record to which all the separate modules will look into (have access to) to do their respective analysis. (kept in memory)

2. All the other modules will individually analyze their respective attack patterns based on the currents packet array.
    - If it needs to analyze based on a single packet then it will do only that much
    - If more needed then it will do accordingly
    - May keep a separate tables of themselves for attack patterns that they are administering
    - Table will keep appropriate record of both current active event and overall record
        ** Including count
        ** Current connections, states
3. Each attack pattern will have separate modules for themselves running in separate threads
    - They will check only their own packet type so that they can check for particular pattern despite the efforts for the attacker to change the signature pattern by delaying or sending other packets
4. The thread list will be maintained by the main class
5. Threads will not be given any parameters, instead a reference to the current packets array
    - All will only read from them and not modify
6. Threads will record to a Findings area where they report Errors or Suspected Behavior
    - They will be provided initially with index so that they know to which current packet did the error generate from
    - This way, the log may also be updated as suspect or criminal

7. Report checking will be done by another thread at the main class
8. All writing and updating will only be done in the main class (may be in a thread) so as to not disrupt the capture process
9. Main class will keep a record or all errors, their type, suspected behavior known pattern of attacks
    - And will notify accordingly if there is a recent change in counts (Alarm) (Highlight)
    - This alarm can be redirected to the firewall or the SysAdm in the future

10. DESIGN FIGURE:



4.4 Initial Stages

During the initial stages of the implementation, difficulties were faced in creating the required code for detecting packets in promiscuous mode, that is detecting all packets that flow thorough a network link. For getting help in knowing how this could be achieved, some open source programs had been studied, which were WinPcap, Snort and Ethereal. WinPcap and Ethereal are Protocol Analyzer softwares, meaning that they sniff on network and return the analyzed and structured form of the communication packets that is sniffed. Snort is an open source Network Based Intrusion Detection System that is developed by many advanced programmers with complex analysis methods using some means of artificial intelligence related discipline and others.

A behavior among the studied softwares was that all of them uses the same common API and library for capturing packets in a network from datalink level, which is pcap.h from libpcap library. This was hence considered as the library to use for capturing packets in our own implementation. The manual pages of Linux on pcap.h gave a very informative view of how to work with pcap.h. Also with the help of some online resources on the use of pcap.h, especially from Tim Carstens on his "Programming with pcap" tutorial, programming for a protocol analyzer and hence the starting of our program was made possible. Capturing in promiscuous mode was made possible also by pcap.h, which could be done by setting the third parameter of the *pcap_open_live()* method of the library, which opens the interface card of the host computer in promiscuous mode.

Also during the initial stages we had plans to include Graphical User Interface for our software. It is not as easy to create one using C/C++, hence some sort of

wrappers were searched for to accomplish the task. Qt Designer tool proved to be a very helpful tool in doing the GUI with C/C++ and was experimented accordingly. However, due to the late stage of being able to make our project work properly, it was not possible to incorporate Graphical User Interface with our implementation.

For threading, the pthread.h library was looked upon, which was the standard library for threading in C/C++. Threading would ensure improved performance which is crucial in heavy traffic network packet analysis.

4.5 The Implementation

4.5.1 Modularization and Work Distribution

The implementation was divided into several stages in development which lead to the formation of several modules. The C code was made such that it was a generic one so that it would be easy to make any modification and addition if on anytime we might want to add some more attack signatures. Hence the code was divided into several files making each function body a separate file entity. This also allowed us to divide up our tasks while working for the implementation. The research of how to capture packets in C and creation of graphical user interface in it has been the major contribution of Rajib Rahman. He was also mainly responsible for creating the reports and presentations that we have prepared. The threading and also graphical user interface was looked upon by Salman Zaman, he also played a major role in the research for the project and also the connectivity design between the modules. Sarwar Alam and Rajib Rahman were also contributors of the main design plan which was followed mostly according to its initial face. Sarwar Alam programed the attack

signatures with help from Rajib Rahman and Salman Zaman for the algorithms. Sarwar Alam also made the connections in programming of the different modules.

As of our initial plans we first had to program for creating the Headers that we would consider in the message stream. The Packet types we covered were Ethernet, IP, TCP, UDP, ICMP, and ARP which makes up almost ninety-nine percent of the network stream in a normal Ethernet Interface. The source codes are in the appendices.

Then for sake of genericness, we created multiple overloads of printing methods that will ensure the portability of the code when printing in to any type of data type. Although focus was eventually given in writing to the console as the GUI was planned out at late stages. Then with the help of “sniffex.c” a sample by Tim Carstens in his “Programming with pcap” tutorial, the Interface Card Initializer was accomplished which meant that the complete protocol analyzer was also built. Some hiccups were encountered here when IP was fragmented, in which case there would be no header in body despite the protocol field. This was solved accordingly.

Arrangements were then done to include a generic thread caller, so that addition of new attack signature would be an easy task. The Internet and the Linux manual pages helped quite a degree in this area. Once the handlers of threads are set, coding for the attack signatures went underway.

4.5.2 Attack Signatures and Algorithms

The initial attack signature that we considered are:

- Echo-Chargen
- Fraggle
- Land

- Ping Flood
- Ping of Death
- Port Scan
- Smurf

Most of them have been implemented as simple way as possible to ensure performance. They are classifiable as Statistical-Analysis Type signature detection and packet anomaly detection type.

In the statistical type signatures, a tolerance level was matched when a fixed number of packets were reached in a relatively quick sequence depending on the attack. For almost most of the attacks, the rate, that is to say the count divided by the time taken are measured and checked whether they exceed the set limit. The limit would vary in different platforms with different speeds of operations and hence should be fine tuned accordingly. Certain problems were faced when flooding type attack patterns were to be checked, as they come so quick that the printing to the console could not catch up. Hence the captured packets are less than what they really are, and hence the tolerance level that is matched is decreased to make it possible to detect. To avoid false alarms, a fixed amount of packets are allowed to come in before analysis are done, about twenty to twenty-five, which is also a matter of fine tuning for the environment worked upon, and the resetting of counts if just the immediate packet was not near enough in terms of time and rate. This eventual corrected scheme worked in solving the problem for stateful packet analysis, which concerns multiple packets in decision making. Global variable STL “map” was used to keep state information.

4.5.3 Pseudocodes

Most of the states are distinguished in terms of IP source address concatenated with "." and IP destination address. A reply is matched with the reverse concatenation.

4.5.3.1 Echo-Chargen

```
If source port is UDP 7 or 19
        and the opposite UDP 19 or 7 in the destination port
when echo port sends to chargen
        set active state for the ipaddress
when echo receives from chargen
        check if currently active
        check if last packet was too old [1in 300sec]
                if passed then increase count else reset
                        if count in the range 20 to 25
                                measure rate
                        if count increases 25
                                reset count and timestamp
                        if rate is greater than tolerance [20 in 60sec]
                                set alarm flag
```

4.5.3.2 Fraggle

```
If  UDP
        set active state for the ipaddress
if ICMP and DESTINATION UNREACHABLE
        check if currently active
        check if last packet was too old [1in 60sec]
                if passed then increase count else reset
                        if count in the range 20 to 25
                                measure rate
                        if count increases 25
                                reset count and timestamp
                        if rate is greater than tolerance [20 in 60sec]
                                set alarm flag
```

4.5.3.3 Land

```
If  TCP and SYN
        if Source and Destination ports are equal
                if Source and Destination IP addresses are equal
                        set alarm flag
```

4.5.3.4 Ping Flood

```
If  ICMP and Request
        check if currently active
                else set active state for the ipaddress and return
        check if last packet was too old [1in 1sec]
                if passed then increase count else reset
                        if count in the range 20 to 25
                                measure rate
                        if count increases 25
                                reset count and timestamp
                        if rate is greater than tolerance [70 in 1sec]
                                set alarm flag
```

4.5.3.5 Ping of Death

```
// Only the single packet signature was handled. Fragmented attack not covered
If  ICMP and Request
        if Packet length > 65535
                set alarm flag

// Proposed for fragmentation
If  ICMP and Request
        check current state
                if continuation add up fragment length
                        if Total Packet length > 65535
                                set alarm flag
```

4.5.3.6 Port Scan

```
// SYN and SYN-ACK scan considered together
If  TCP and (SYN or SYN-ACK)
        check if currently active
                else set active state for the ipaddress and return
        check if last packet was too old [4 / 60sec]
                if passed then increase count else reset
                        if count in the range 20 to 25
                                measure rate
                        if count increases 25
                                reset count and timestamp
                        if rate is greater than tolerance [100 in 1sec]
                                set alarm flag
```

4.5.3.7 Smurf

```
// Assuming router is set to disallow this pattern, hence any
// broadcast is smurf. Proposed idea is to keep an arp record
// and check with the hardware address with the IP address
// whether any sort of spoofing is tried

If  ICMP and Request
        if host byte of the IP destination Address is broadcast [255]
                set alarm flag
```

Chapter 5 Analysis and Testing

Testing was part of the implementation process, and the project is such that without proper working of the modules it is not possible to move on. It was ensured that the protocol analyzer was a complete one and handled all kinds of exception situations such as IP fragmentation, not measuring size of the UDP and using the IP length field instead, checking for valid length of headers and body from both the header fields and the pcap controlled length measure, which measures length in physical layer as the distance between the start and end flag of physical layer data transmission. It was made sure that the protocol analyzer worked for all packets, ARP, IP, TCP, UDP and ICMP in all their variations including the encapsulation of a new IP packet of an ICMP packet which is not of type Request or Reply. Most of the testing and operations had to be done in super user mode with root privileges as access to the interface card is limited to other lower level user accounts.

The program of the thesis can be run as both IDS and protocol analyzer mode. A user manual is included in the appendices. It can accept filter strings that are of the format of “TCPDUMP” which is Linux standard console command to dump packets from the interface card. The manual of Tcpdump is provided with Linux and hence not included here, however, a minor introduction to it would be that it can take several strings such as 'ip', 'tcp', 'port', 'icmp', 'arp', 'and', 'or' and it allows use of parenthesis for grouping logics. The logics it uses is like digital logic as 'and' and 'or' are supported. For example, if the tcp port 1863 or 1683 id to be listened to for analysis, the filter sting

would be “port (1863 or 1683)”. This way testing the software for any particular host could be done, and also if it is empty then all host are scanned.

The programs used to test the working of the attack signature detection was done by using external means such as codes and Linux root level privileged commands. For example the land attack was checked using an external code that sent packets in the destructive format of Land Attack packet, and our IDS was able to catch it and send alarm. Some screenshots are included in the appendices to show the console printout when a Land Attack and port scan were tested. Port Scan was tested by using NMAP port scanner and OS fingerprinting software. The console command was *“nmap -sT -sR -O -PT 192.168.40.1”*. To check smurf, the root terminal code was *“ping -b 192.168.40.255”.* To check ping flood, the command was *“ping -f 192.168.40.1”.* All were successful and also a point to mention was that in a 10Mbps LAN the tolerance that were used worked quite well as not much messages were printed, but a moderate rate of alarming was achieved, which was like ten messages in three seconds when there was a flooding of twenty-thousand packets in ten seconds were being sent. As mentioned before, this is dependent on the platform used and the speed of the network and host computer, and some fine tuning may be necessary to reduce or increase the numbers of messages desired.

An important point in testing the IDS was to check how much of extra alarms and or false alarms are sent during operation. Both false rejection and false acceptance are not tolerable. However the attack patterns that we checked are mostly anomaly based and statistical based analysis methods, in which the former shouldn't

suffer from false alarms. But the latter had chances, but since they are mainly network layer attacks, the signatures were enough to determine whether a packet stream is hostile or not. But the problem that we had to encounter was multiple similar messages for flooding type of attack, which was discussed before as to have been handled properly, and some fine tuning of tolerance values will also help in this matter.

Chapter 6 Future Development and Related Studies

Because of problems we faced in determining how to go about our implementation in the initial stages, we were able to make the working software very late and hence weren't able to add much attack signatures due to the time constraints. However, most other attack types are very similar to the ones we implemented and hence adding them to our implementation will not me a major task. We plan to include them in the near future.

Most of the related studies regarding Intrusion Detection System nowadays focus on how to reduce false alarms and false acceptance and rejection, and to better catch intentional variations of attack signatures. This involves the use of certain 'intelligent' mechanism of analysis. Hence they include heuristic analysis with todays IDSs.

Hence we would like to add heuristics and artificial intelligence as part of our project as future improvement plans. In trying to add more complex attack patterns we might encounter the false alarm incidents, hence we would also want to focus on reducing them.

We would also want to add Graphical User Interface as part of our implementation and also make an executable for the windows platform. Plans are also there to make it an open source resource in the Internet so that further development can be done on it with help from developers from other places.

Conclusion

Network Security is a very important part of corporate world today, even though it seems that vulnerabilities are not high, but serious damage can be caused from a remote point in a network. Most of these attacks are in the message stream in packet format. Although routers and firewall can be set to check attack patterns, but they lay in the middle of a network traffic and controlling attacks from them would have a detrimental effect on network performance. A deep level of analysis method is required and it calls for the use of what can be now realized as the ever so important Intrusion Detection System.

In our report, we have introduced the different types of Intrusion Detection Systems and their place of uses. Also, we have noted out the various forms of network attacks and how they are performed, to give idea of the challenges of Network Security discipline. Then we have pointed out the ways we went about creating our own implementation of the Intrusion Detection System and the problems we faced.

Having worked so far, we have observed that every solution to a network vulnerability gives rise to another form of security threat. Hence it can be concluded that there is no end or ultimate solution for Network Security, instead it demands the constant monitoring and development of it's security measures. So is the Intrusion Detection System, which can always be thwarted if not maintained and updated regularly and properly. Hence there is a scope for continuous research in this area, and it promises to be quite a challenging prospect for both study and career. Hence we are willing to be part of the development of Network Security in the future and we would encourage anyone to be a part of it as well.

Appendices

Appendix A User Manual

MODE	PARAMETER
Protocol Analyzer	p
Record to file	f
Fixed number of capture	<number> [only at the beginning]
Local Host	lo
Filter Expressions	as the tcpdump manual page of Linux
Help	--help,--usage

Appendix B Screenshots

Successful capture of Land Attack and Port scanning or Os fingerprinting attempt:

```
[Attack Types]
TCP LAND Attack

[INFO]
    Packet Number 1
    Sun Jan  7 15:39:53 2007

[ETHERNET HEADER]
    Src Mac: 00:e0:4c:1c:21:94
    Dst Mac: 00:e0:4c:50:34:96
       Type: IP

[IP HEADER]
    Version: 4
   Hddr Len: 20
        TOS: 0x00
  Total Len: 40
         ID: 3868
   Reserved= 0
 Dnt Frgmnt= 0
 MoreFrgmnt= 0
 Frgmnt Off: 0
        TTL: 255
   Protocol: TCP
 HddrChkSum: 55934
       From: 192.168.40.114
         To: 192.168.40.114

[TCP HEADER]
   Src Port: 80
   Dst Port: 80
    Seq Num: 3868
    Ack Num: 0
   Hddr Len: 20
        URG= 0
        ACK= 0
        PSH= 0
        RST= 0
        SYN= 1
        FIN= 0
 WindowSize: 2048
     ChkSum: 38341
 UrgentPntr: 0
```

```
[Attack Types]
TCP Port Scan and OS fingerprinting Attempt

[INFO]
    Packet Number 151
    Sun Jan  7 15:41:49 2007

[ETHERNET HEADER]
    Src Mac: 00:e0:4c:1c:21:94
    Dst Mac: 00:e0:4c:50:34:96
       Type: IP

[IP HEADER]
    Version: 4
   Hddr Len: 20
        TOS: 0x00
  Total Len: 44
         ID: 16794
   Reserved= 0
 Dnt Frgmnt= 0
 MoreFrgmnt= 0
 Frgmnt Off: 0
        TTL: 54
   Protocol: TCP
 HddrChkSum: 28919
       From: 192.168.40.120
         To: 192.168.40.114

[TCP HEADER]
   Src Port: 41533
   Dst Port: 18
    Seq Num: 3450044808
    Ack Num: 0
   Hddr Len: 24
        URG= 0
        ACK= 0
        PSH= 0
        RST= 0
        SYN= 1
        FIN= 0
 WindowSize: 3072
     ChkSum: 51311
 UrgentPntr: 0
   Options (4 bytes):
```

References

- Alessandro Rubini & Jonathan Corbet , “Linux Device Drivers” - 2nd Edition, O’Reilly & Associates, Inc.
- Dr. Eric Cole, Dr. Ronald Krutz & James W. Conley, “Network Security Bible”, Wiley Publishing, Inc., 2005
- Rebecca Bace & Peter Mell, “Intrusion Detection Systems”, NIST
- Pars Mutaf, “Defending against a Denial-of-Service Attack on TCP”
- Stephen Northcutt, “Network Intrusion Detection: An Analyst's Handbook” - First Edition, New Riders Publishing, June 16, 1999
- Biswanath Mukherjee, L. Todd Heberlein &Karl N. Levitt, “Network Intrusion Detection”, Wiley Publishing, Inc., 2005
- Charles P. Pfleeger, Shari Lawrence Pfleeger, “Security in Computing” - Third Edition, Prentice Hall PTR, December 02, 2002
- Joel Scambray, Stuart Mcclure & Geogre Kurtz, “Hacking Exposed: Network Security Secrets and Solutions” - 2nd Edition, Osborne/McGraw-Hill, 2001
- Evangelos P. Markatos, Spyros Antonatos, Michalis Polychronakis, Kostas G. Anagnostakis, "Exclusionbased Signature Matching for Intrusion Detection", CCN, 2002
- "Denial-of-service attack", http://en.wikipedia.org/wiki/Denial-of-service_attack
- Tim Carstens, "Programming with pcap", http://www.iac.rm.cnr.it/~massimo/pcap.htm
- Linux Manual Pages