A Protocol Layer Survey of Network Security

JOHN V. HARRISON AND HAL BERGHEL

Center for Cybermedia Research
University of Nevada, Las Vegas
Las Vegas, NV 89154
USA

Abstract
It is difficult to estimate how much more the Internet will impact modern society. The use of this ubiquitous computing and communication platform to provide instant worldwide communication, increase business productivity, create more effective markets, and overcome geographical boundaries, is only now starting to be fully appreciated. Unfortunately, all of the benefits mentioned above, and the potential for any future benefits, are at risk of being lost due to security vulnerabilities. Cybercriminals and cyberterrorists, including both state-sponsored and individual, are attacking computing and networking systems at an increasing rate. And all individuals, corporations and governments who utilize these systems are at risk.

This chapter provides a survey of this electronic information battlefield. This survey differs from others in that it presents common categories of attacks from the perspective of the TCP/IP protocol stack layers. By presenting categories of attacks by protocol layer, one may focus on the nature of the vulnerabilities, rather than become mired in the minutiae. After presenting categories of attacks, an overview of the techniques that has been created to defend against the various attack categories is presented.

ISSN: 0065-2458/DOI 10.1016/S0065-2458(04)64003-4

1. Introduction

John Alger [2] defines information warfare as those actions intended to protect, exploit, corrupt, deny, or destroy information or information resources in order to achieve a significant advantage, objective or victory over an adversary. Adversaries can be nations, corporations or individuals. Some examples of the objectives of these adversaries include vandalism, political activism, theft, fraud, extortion, harassment, espionage, terrorism and the destruction of nation states.

As the sophistication, frequency, cost of computer-based information warfare attacks continues to increase, the inherent limitations of traditional approaches towards hardening current software and hardware infrastructure has become apparent. This problem is partially due to the wide-scale deployment of Internet-connected hardware and software that was originally designed for limited access, isolated, unconnected environments. It is also partially due to the ever-increasing complexity and interdependency of modern software components.

The Internet has irreversibly changed society. Its use in business and industry, education, health care, government and individuals is now ubiquitous and has become a mainstay of modern communication and commerce. An almost limitless level of information sharing is now possible. Any information resource can be made available to anyone, regardless of location, income or social status, on demand. We are now realizing the dream of Vannevar Bush of a Memex—an indexed information warehouse in which any individual could store all personal and public information and communication of relevance in a way that could be immediately accessed [9].

Unfortunately, all of these benefits are in jeopardy. Hostile nations, criminal organizations, unethical business enterprises, cyberterrorists, hackers and neophyte script kiddies are all attacking computer systems and the networks on which they reside with increasing frequency, and in so doing affecting increasing numbers of individuals, corporations and governments with which these networks are associated. Their motivations for information warfare span the gamut of vandalism, theft, fraud, defamation, violence, terrorism and the destruction of governments. Everyone who uses the network, whether they know it or not, are both exposed to, and at risk from, these attackers [6].

This chapter is in a sense a defensive electronic order of battle. As a survey, it differs from others in that common categories of attacks in a structure consistent with the TCP/IP protocol stack layers [18,44] are presented. By presenting categories of attack techniques, as opposed to the myriad of esoteric technical details necessary to understand any one single instance of an individual attack, the reader can avoid becoming overwhelmed by "trees" that block a view of the "forest." After presenting categories of attacks, we then present an overview of the technology that has been created and deployed to defend against the various attack categories.

Our objective is to provide the reader with an understanding of the attacks that have been, or soon will be, launched at the reader, as well as the reader's organization and nation. Some attack methods, e.g., deployment of malicious e-mail carrying a virus payload, have received substantial attention in both technical and non-technical literature. However, other types are much less well known—a situation this chapter is meant to remedy.

2. Overview of TCP/IP

This section contains a very brief overview of the *Transmission Control Protocol* (*TCP*)/*Internet Protocol* (*IP*) protocol, which is used to facilitate communication on computer networks of all sizes. Some of the attacks classified below require a basic understanding of TCP/IP. Readers who have a background in TCP/IP can skip this section.

Like many other network protocols, TCP/IP consists of "layers" that form a "stack." Each layer adds new functionality not present in the layers below. The stack of layers abstract the physical details of the network hardware to such a degree that the highest layer can view a connection between two communicating software applications to view their communication link as reliable channel of messages between them.

The functions that are implemented in each layer conform to a standards specification. These layers insulate software executing on the nodes of the network from changes to, or replacement of, the underlying network technology. These nodes could be computers, routers, switches, hubs, and other devices that wish to communicate over the network.

TCP/IP is a four-layer protocol. The layer closest to the network hardware, referred to as the “lowest” in the TCP/IP layer hierarchy, is the *link* layer. The link layer is implemented within the network adapter and its corresponding device driver. Ethernet is the most common type of underlying network technology for devices using TCP/IP. Ethernet includes a link layer specification.

The data transmitted over Ethernet networks is partitioned into units of data referred to as *frames*. A frame contains a hardware destination address and hardware source addresses. These addresses identify the specific network adapters on the communicating devices. In theory these hardware addresses need not be unique. However hardware manufactures are allocated ranges of addresses that are disjoint, so it is rare for two network hardware components to share an address unless specifically required.

Frames have a maximum size that is dependent on the network hardware in use. Units of data to be transmitted that are larger than the maximum frame size must be fragmented into units less than the maximum, and then reassembled on arrival at their destination.

Residing just above the link layer in the TCP/IP stack is the network layer. The network layer is implemented using Internet Protocol (IP). IP is the mechanism that transmits units of data across multiple networks, which comprise the Internet, to arrive at its destination. These units of data are transmitted across each individual network using the link layer frames.

Every node that is attached to the Internet is assigned an IP address. In theory, these addresses are unique globally. However, in practice some address ranges, which are termed *private* addresses, are reused in manners that do not result in addressing conflicts. All network devices that possess IP addresses also possess link layer (physical addresses). Together the two addresses form a (local) mapping that can be used to infer an IP address given a physical address and vice versa.

IP includes specific protocols for reasoning with this mapping. The Address Resolution Protocol (ARP) is used to obtain the physical address corresponding to an IP address. ARP uses IP to broadcast a ARP request on the network. When a host receives an ARP request containing its own IP address, it sends an ARP reply message containing its hardware address. There is also a Reverse ARP (RARP) protocol, which is used by a host to determine its own IP address if it has no way of doing this except via the network.

Internet Protocol would be unnecessary if every computer was connected to the same Ethernet cable. All messages could be sent directly to the destination computer using its physical address. Inherent limitations on both the size of an Ethernet network, as well as restriction that only one computer can transmit successful at a time, forces multiple networks be created hence the need for IP addressing.

IP is a connectionless protocol. Each unit of data transmitted, termed a *packet*, is transmitted as an independent message and is not viewed as having a relationship with any other packets. Furthermore, IP does not ensure that a packet will reach its final destination, or whether packets that do arrive will arrive in the original order that they were sent. There is no information in a packet to identify it as part of a sequence of related packets. Any mechanism for ensuring that transmitted data arrives at its destination, and in the correct order, is provided by a higher protocol in the suite.

An IP packet consists of an IP header and payload data. The header includes a 4-bit protocol version number, a header length, a 16-bit total length field, an 8-bit protocol identifier, a header checksum, control fields and the 32-bit source and destination IP addresses.

The protocol field identifies which higher-level TCP/IP protocol is using the IP layer to encapsulated its data. When a packet arrives at the destination for which it is intended, IP extracts the encapsulated data and provides it to the designated higher-level protocol module.

The *Time-To-Live* (TTL) control field is initialized to an arbitrary value by the sender. The value of the field is decremented by one by every IP-aware network device that the packet passes through in transit to its destination. When the value reaches zero, the packet is considered unable to reach its destination. It is then discarded and the sending IP layer is sent a notification of the event using the Internet Control Message Protocol (ICMP). ICMP is one protocol used by network devices for transmitting diagnostic messages.

IP addresses are logically partitioned into at least two parts, namely the network identification (ID) and the host identification. Together the host identifier specifies a particular network device on the network specified by the network identifier. The network ID is often further decomposed to represent a hierarchy of networks (or subnets).

Transmission Control Protocol (TCP) implements the transport layer of TCP/IP. It is termed a “connection-oriented” protocol. Two hosts that wish to communicate must establish a connection before application data can be transferred between them. TCP provides reliability of transmission that is unavailable with IP.

For error detection, TCP computes and uses checksums for both the header and payload data. When data is received, TCP sends an acknowledgement back to the sender. If the sender does not receive an acknowledgement within a certain predetermined time, the data is re-sent. TCP includes mechanisms for ensuring that data that arrives out of sequence is properly re-sequenced.

TCP implements a *flow control* mechanism. This mechanism ensures that a fast sender does not overwhelm a slow receiver with data. TCP employs IP for data transmission but adds it own data unit abstraction, which is termed a *segment*.

Each segment contains 20 bytes of header information in addition to the IP header. The TCP header contains a 16-bit source and destination *port number* fields. “Port number,” in this context, represents a software port, not a hardware port. An IP address and a port number taken together uniquely identify a service running on a host. This identifying pair is known as a *socket*.

The header also includes a 32-bit sequence number. From the perspective of the receiver, the sequence number enables the receiving TCP to reconstruct the data stream at the destination in the correct order even when segments are received out of sequence. The 32-bit acknowledgement field is used to convey back to the sender the point in the data stream that has been successfully received.

Before any data can be sent between two hosts using TCP, a connection must be established. One host, acting as the server, waits to receive connection requests from clients. To request a connection, a client sends a TCP segment specifying its own port number and the port that it wants to connect to. The “SYN” (synchronize sequence numbers) flag is set, and the client’s initial data sequence number is specified. To grant the connection, the server responds with a segment in which the header contains its own initial data sequence number. To acknowledge receipt of the client’s data sequence number the acknowledgement field contains that value plus one.

To complete the connection establishment protocol, the client acknowledges the server’s data sequence number by sending back a segment with the ACK flag set and the acknowledgement field containing the server’s data sequence number plus one.

Segments are only sent between client and server TCP transmits on behalf of an application (in the *application* layer). No polling is performed by TCP to monitor the status of the connection. If the communication medium fails, neither participant will become aware of the failure until further data needs to be sent. However, in all popular TCP/IP implementations a timeout implemented by the application will usually terminate a connection after a long period of inactivity.

The *User Datagram Protocol* (UDP) is a second transport layer protocol of TCP/IP. It adds little to the functionality of IP. Like IP, it is an unreliable, connectionless protocol. No connection must be made between the sender and receiver prior to data exchange. There is no inherent mechanism for ensuring that data sent is received.

A unit of data sent using UDP is called a *datagram*. Like TCP, a UDP datagram is addressed using source and destination port numbers. As stated above, port numbers identify which protocol module sent, or is to receive, the data. Most protocols have standard ports that make the identification trivial. The use of standard port numbers makes it possible for clients, and attackers, to communicate with a server without first having to establish which port to use.

Both UDP and TCP protocols use the port number to determine which application-layer protocol should receive the data. The *application layer* is the top layer of the

TCP/IP stack. Applications that use the TCP/IP protocol for communication implement their own protocols above the transport layer. For example, e-mail applications implement the Simple Mail Transport Protocol (SMTP), which itself uses the sockets provided by the transport layer.

3. Offensive Techniques

In this section we present common categories of attacks in an organized structure consistent with the TCP/IP protocol stack layers. A good source of general information about the TCP/IP protocol stack layers, which we borrow from for this chapter, can be found in both [18] and [44]. An understanding of this material will assist the reader in understanding the attacks described. However, for those readers without a preexisting understanding of TCP/IP protocol stack, we provide a basic description of each layer necessary to enable some level of understanding the attack categories that appear in the sections below.

3.1 Physical Layer

The physical layer includes those functions necessary to transmit information along a physical telecommunication medium. It addresses the electrical and mechanical specifications as well as the functional and procedural characteristics, of the environment involved. RS-232, Ethernet, FDDI, Token Ring, SLIP, PPP and DSL are all physical layer protocols.

The deployment of wide-area communication networks (WANs) pose a significant security challenge. They usually rely on large expanses of minimally protected cable infrastructure. They are subject to surreptitious signal monitoring, jamming and disrupting by physical means. The are also susceptible to interference at many points in this infrastructure, which may include optical repeater nodes, switching nodes, operation and management nodes, and the stretches of cable itself [38].

WANs typically carry large volumes of data, making even short duration outages potentially very costly and disruptive. The impact of outages or congestion in wide area networks is becoming increasingly serious as more organizations and individuals rely on their availability

3.1.1 Signal Disruption

While many types of network attacks require either a high degree of technical expertise or significant financial or personnel resources to mount successfully, some physical layer attacks may be accomplished with few resources, little expertise, and, in some cases, a high degree of covertness.

Most of the nation’s network infrastructure is buried in the ground within a few feet of the surface or mounted on utility towers. It is not difficult to dig up and severe cables as illustrated by how often it occurs accidentally. One attacker with a backhoe or several with ordinary tools available at a home and garden store could expose and damage high bandwidth cabling.

The large geographic extent of many network backbones gives attackers a relatively high probability of mounting such an attack covertly, until after the attack succeeds, and then escaping immediately after the attack. Additional attacks could then be mounted in other sections of cable that have been disconnected by the initial attack to prolong the duration of the outage.

There are maps available, some on the Internet, that provide the general location of high bandwidth cables that comprise the national telecommunication infrastructure. Furthermore, many cable locations are well known in the industry, and there are databases available to help construction contractors avoid accidentally digging them up. It is not hard to imagine that a few individuals might be able to locate vulnerable cables and choose attacks to maximize network disruption.

A disruption of several hours may represent little more than an inconvenience to a casual e-mail. However, for organizations relying on network infrastructure availability for sending time sensitive information the one could envision the consequence being much more serious. For example, the attacks might be timed to disrupt communications during events of geo-political significance. Furthermore, since the military of most nations are relying increasingly on both public and private high-bandwidth network infrastructures, an attack could also have military significance.

3.1.2 Packet Sniffing

A packet sniffer, sometimes referred to as a network monitor or network protocol analyzer, can be used legitimately by a network or system administrator to monitor network traffic across the communication medium. Using the information captured by the packet sniffer an administrator can identify bottlenecks and maintain efficient network data transmission. However, an attacker can deploy a packet sniffer for malicious purposes.

In its most basic form a packet sniffer simply captures all of the packets of data that pass through a given network interface. However, if the network interface card is placed into “promiscuous mode,” the packet sniffer is also capable of capturing all packets traversing the network regardless of the source or intended destination. One level further toward stealth is the operation of the network card in “monitor” mode which not only captures all packets traversing the network with which the card is authenticated, but all networks whose signal is within range.

An attacker can capture and analyze all of the network traffic using a covertly installed packet sniffer. Subsequent analysis of the traffic may yield sensitive private information, such as username, password and credit card information if packets were unencrypted.

Detecting rogue packet sniffers on a network is non-trivial. One technique involves monitoring the performance of a machine suspected of hosting a packet sniffer when suddenly exposed to a network traffic burst. Another technique is to monitor domain name queries issued by the suspect machine. Some sniffers issue numerous queries to enable them to match IP addresses to hostnames. This allows the sniffer to present a more user-friendly display. However, passive sniffers are usually very hard to detect in both wired and wireless environments.

In a shared network (or "hub") environment, all hosts are connected to the same communication channel and compete with one another for bandwidth. In such an environment packets meant for one machine are, in fact, visible by all other machines. Thus, any machine in such an environment placed in promiscuous mode will be able to capture packets meant for other machines and can therefore listen to all the traffic on the network.

A network environment in which the hosts are connected to a switch instead of a hub is called a switched network. The switch maintains a table keeping track of each computer's physical address and delivers packets destined for a particular machine to the port on which that machine is connected. The switch is an intelligent device that forwards packets to the destined computer only and does not broadcast to all the machines on the network. It can be viewed as constructing a set of two-node networks (the computer and switch) with internal memory to store packets that are to be forwarded between the two-node networks.

However, there is an additional benefit. Only traffic destined for a node on the two-node network will be transmitted on the two-node network. Packet sniffing, in its most simple form, cannot be directly performed in switched network. However, there are other methods that an attacker can access packets in a switched network, which are described below.

3.2 Data Link Layer

In this section we provide some brief comments about the data link layer before describing attack scenarios. Some of this material was abstracted from [17] and [18].

The data link layer transforms the raw data provided by the physical layer into basic organizational units. As such it deals with data frame formats, low-level error detection, data flow control, and any additional considerations that would be required to establish a reliable link between two hardware devices. It also mediates between multiple devices on a network to specify who has control of the communication medium at any point in time.

A computer connected to the Internet has two addresses. One is the physical address, which is often referred to as a MAC (Media Access Control) address. The MAC address uniquely identifies each node in a specific network and on many machines is stored on an embedded network card. Each network card has a unique MAC address, which is used by the Ethernet protocol for addressing the frames that are transmitted to and from a machine on a specific network. The second address is the Internet Protocol (IP) address. The IP address uniquely determines a computer's connection to the Internet and is meaningful starting at the network layer, which is one level above the link layer in the OSI model.

The Ethernet frame header includes the MAC address of the destination machine but not the IP address. IP addresses are mapped to the corresponding MAC address. For efficiency, the mapping is usually retained in primary memory in a table, which is referred to as the ARP (Address Resolution Protocol) cache. If no entry is found for a particular IP address in the cache, the ARP module broadcasts a "request" frame (ARP request) to all machines on the network. The machine with that IP address responds to the source machine with its MAC address ("ARP reply"). This MAC address is added to the source machine's ARP cache, and is then used by the source machine in all its communications with the destination machine.

3.2.1 ARP Spoofing

ARP is used to obtain the MAC address of a destination machine when a source machine has a frame to send but only possesses the machine's IP address. ARP is a stateless protocol. An ARP reply can be sent, and processed by the destination, even if a corresponding ARP request was not issued.

An attacker can intercept the traffic originating from a machine by performing "ARP spoofing." In this scenario, the attacker issues an ARP reply to the network stating that the gateway of the network is the attacker's machine. The ARP cache of machines that receive this ARP reply will now have an incorrect entry for the gateway in their cache. Their cache is said to be "poisoned."

All the traffic from machines with poisoned caches destined for the gateway will pass through the attacker's machine. Alternatively, a host's ARP cache can be poisoned by setting the gateway's MAC address in the victim host to the broadcast MAC address. In this case also all of the hosts traffic will be sent to the attacker (and all other machines).

3.2.2 MAC Flooding

Switches maintain a translation table that maps various MAC addresses to the physical ports present on the switch. Using this table, a switch can route packets from one host to another without having to route communication beyond the switch. So long as the switch translation table contains all of the relevant MAC addresses, the switch will be effective. However, if a MAC address is referenced that is not included in the translation table, the switch has no alternative but to direct the traffic beyond the switch to the LAN to which the switch is connected (this is called the “fail open” mode). This is the genesis of the MAC Flooding (aka “switch flooding”) exploit.

MAC flooding exploits the memory limitations of switches. The attacker inundates the switch with bogus MAC addresses until the switch’s memory resources are completely consumed. At this point all legitimate traffic to MACs that were in the original translation table will be redirected to the LAN. This turns the switch into a hub, and broadcasts all traffic that should have been internal to the switch over the LAN where it is visible to all the machines on the network, which makes the switched traffic vulnerable to packet sniffing.

3.3 Network Layer

The network layer handles delivery of packets between devices that may be connected to different networks and separated by many others. The network layer is responsible for logical addressing, namely managing the mapping between IP addresses and machines on a worldwide basis. It is also responsible for ensuring the packets sent to machines on different networks navigate through the various networks successfully and arrive at their destination. Network appliances such as routers and gateways operate at the network layer.

3.3.1 Fragmentation Attacks

The network layer, which is based on Internet Protocol (IP), handles movement of packets around and between networks (via routing). All networks have a maximum packet size, based on the characteristics of the underlying link layer. This is called the network’s maximum transmission unit (MTU). To illustrate, the typical MTU for Ethernet is 1500 bytes.

Not all network technologies have the same size MTU. When a packet arrives at a network that has a MTU that is smaller than the size of packet, the packet must be partitioned into fragments. The fragments are then transmitted over the network. The destination machine’s network layer is responsible for reassembly of the fragments to reconstruct the original packet. In order for this reassembly to take place seamlessly, several pieces of information must be included in the fragment header: fragment

ID, fragment offset, length of data payload, and an indicator whether the current fragment is the last fragment in the chain. Packet fragmentation attacks frequently involve manipulation of these key fields. Throughout this discussion it should be remembered that only the first fragment of a fragmented packet has the protocol header!

One method an attacker can use to exploit these vulnerabilities is to manipulate the value of the "fragment offset" field. This field designates where the data arriving in a packet fragment must be placed in the memory area where the entire packet is being reconstructed. One exploit involves setting the value of the fragment offset on a fragment so low that instead of appending the fragment to an earlier one, IP actually overwrites the data, and perhaps part of the packet's header, in the earlier fragment. Using this technique, an attacker can transmit a malicious packet that might otherwise be detected on arrival were it not decomposed into a series of smaller packets each containing a fragment. This is the technique that was used in the "Teardrop" exploit as is illustrated in the following two TCPDump records:

```
hacker.net 22 > target.org 33: UDP (frag 123:64@0++)
hacker.net > target.org(frag 123:20@24)
```

This is read in the following way. The hacker machine sends two UDP packets to the target machine. The fragment ID is "123" in both cases. The first packet says "the following packet contains 64 bytes starting with offset 0." The second packet says, "the following packet contains 24 bytes starting at offset 20." As reassembly takes place in order, the second UDP packet overwrites bytes 21–45 in the original packet. This technique is commonly used to camouflage packet signatures that would normally be flagged by static firewalls and older intrusion detection systems that monitor individual packets but not the entire fragmentation chain.

3.3.2 Smurf Attack

The Internet Control Message Protocol (ICMP) augments the IP protocol. ICMP is used to handle errors and exchange control messages and can be used to determine if a machine on the Internet is responding to network requests. For example, one type of ICMP packet is a "echo request." If a machine receives that packet, that machine will respond with an ICMP "echo reply" packet. This basic message exchange is used to convey status and error information including notification of network congestion and of other network transport problems. ICMP can also be a valuable tool in diagnosing host or network problems and is the basis for the "Ping" network diagnostic utility.

ICMP packets are encapsulated inside of IP datagrams. There are 15 different types of ICMP messages, including "ICMP_ECHO REPLY" (the response) and

"ICMP_ECHO" (the query). The normal course of action is for an "ICMP_ECHO" to elicit a "ICMP_ECHOREPLY" response from a listening server.

On IP networks, a packet can be directed to an individual machine or broadcast to all machines on the network. When a packet is sent to an IP broadcast address from a machine on the local network, that packet is delivered to all machines on that network. An unprotected network may allow a packet to be sent to the IP broadcast address from a machine outside of the local network. If so, the packet will be broadcast to all machines on the target network.

In the "smurf" attack [11], attackers use ICMP echo request packets directed to IP broadcast addresses from remote locations to generate an overwhelming number of ICMP echo reply packets. The network is overwhelmed by the sheer volume of traffic, which interferes with legitimate traffic. The machines generating legitimate traffic are denied access to the network, which is why this attack is considered one of a class called "denial-of-service" attacks.

When the attackers create these packets, they do not use the IP address of their own machine as the source address. Instead, they create crafted packets with a spoofed (false) source address of the attacker's intended victim. When all the machines on the network respond to the ICMP echo requests, they send replies to the victim's machine. The victim is subjected to congestion that could potentially make the network unusable. Attackers have developed automated tools that send these attacks to multiple networks at the same time, causing all of the machines on these networks to direct their responses to the same victim.

3.3.3 Covert Data Channels

A covert channel is a communication mechanism in which information can pass, but which is not ordinarily used for information exchange and hence is difficult to detect and deter using typical methods. Detailed technical information about one infamous covert channel attack can be found in [15].

Ping has a standard packet format recognized by every IP-speaking device and is in widespread use for network management, testing, and performance analysis. Firewalls are often configured to assume ping traffic is benign and then allow it to pass to the protected network. However, Ping traffic can open up covert channels through the networks in which the traffic is permitted.

ICMP_ECHO packets also have the option to include a data section. This data section is used when the record route option is specified, or, more commonly, to store timing information to determine packet round-trip times. However, there is no check made as to the content of the data. This transmitted data section serves as the covert channel. Encoded messages, malicious software and/or commands to preexisting malicious software can reside in the data section.

The infamous “Loki” tool exploits the ICMP data section covert channel. Note that covert channel may be enabled in many other protocols besides ICMP. Any fields of any protocol message that is not critical to ensure accurate could be candidates for a covert channel. In the case of Loki, the options field of the ICMP packet usually contains encrypted data that is reassembled by the compromised target computer.

3.4 Transport Layer

The transport layer is responsible for the *reliable* delivery of an entire message from a source process to a destination process. This message may be comprised of a multiple IP packets. Although the IP network layer handles each packet independently, without recognizing an relationship between them, the transport layer ensures that the entire message arrives at the destination intact and is reassembled in the correct order.

The transport layer handles acknowledgements, error control and flow control, packet sequencing, multiplexing, and any other process-to-process communication required. TCP is the transport layer of the TCP/IP protocol. Many applications rely on the connection-oriented services provided by TCP. Examples of TCP applications that are staples of modern Internetworking include Hypertext Transport Protocol (HTTP), File Transport Protocol (FTP), Secure Shell (SSH), Internet Message Access Protocol (IMAP), Simple Network Management Protocol (SNMP), Post Office Protocol v3 (POP3), Finger, and Telnet.

When applications that employ these protocols are launched, the TCP/IP software on the local device must establish a connection with the TCP software on the destination device. The 3-way handshake takes place between the endpoint computers. Assume that device A is the initiator of the communication and device B is the target of the initiator’s connection process.

(1) Device A sends its TCP sequence number and maximum segment size to Device B. This information is in the form of a TCP/IP “SYN” packet, which is a packet with the SYN bit set.
(2) Device B responds by sending its sequence number and maximum segment size to Device A. This information is in the form of a “SYN/ACK” packet, which is a TCP/IP packet with the SYN and ACK bits set.
(3) Device A acknowledges receipt of the sequence number and segment size information. This information is in the form of a TCP “ACK” packet, which is a TCP/IP packet with the ACK bit set.

The connection between the devices is then open and data can be exchanged between them. Although this appears quite simple, an attacker can manipulate this handshaking process to overwhelm and compromise a TCP/IP enabled device.

3.4.1 TCP/IP Spoofing

IP Spoofing is a method that an attacker can employ to obtain a disguise. With IP spoofing, an attacker gains unauthorized access to a computer or a network by making it appear that malicious traffic has originated from a trusted host. Details of this attack type can be found in [3].

The characteristics of the IP protocol that enable IP spoofing include its connectionless model, where each datagram is sent independent of all others. Furthermore, there is no inherent, built-in mechanism in IP to ensure that a packet is properly delivered to the destination. An attacker can use one of several freely available software tools to easily modify a packet's "source address" field, hence masking its true originating IP address.

Like an IP datagram header, the TCP packet header can also be manipulated. The source and destination port number for a communication session, which are found in the TCP header, determine the network applications performing the communication.

The TCP sequence and acknowledgement numbers are also found in the TCP header. The data contained in these fields is intended to ensure packet delivery by determining whether or not a packet needs to be resent. The sequence number is the number of the first byte in the current packet, which is associated with a specific data stream. The acknowledgement number contains the value of the next expected sequence number in the stream sent by the other communicating party. The sequence and acknowledgement numbers are used by the communicating parties to ensure that all legitimate packets are received. Reliable delivery of packets is facilitated by the TCP layer.

An attacker can alter a source address by manipulating an IP header, hence masking a packets true source. A related attack, which is specific to TCP, is *sequence number prediction*. This attack can lead to session hijacking or host impersonating and requires the use of spoofing techniques. Several attack subtypes are possible using these methods, which are outlined in [45].

Non-Blind (TCP) Spoofing occurs when the attacker's machine is on the same subnet as the victim. The sequence and acknowledgement numbers can be obtained by sniffing, eliminating the potential difficulty of calculating them accurately. This permits the attacker to attempt a hijack of the TCP session. The attacker corrupts the data stream of an established connection, then re-establishes the connection but with the attackers machine in place of one of the authorized parties. The reestablishment is enabled because the attackers machine uses the correct sequence and acknowledgement numbers. Using this technique, an attacker can circumvent authentication techniques employed to establish the connection.

Blind (TCP) Spoofing is a more complex attack because the sequence and acknowledgement numbers are not immediately assessable. The attacker must transmit packets to the target machine in order to sample sequence numbers. By examining the sequence numbers, the attacker must then attempt to guess how the victim TCP layer generates sequence numbers. This is a more difficult task now that most TCP layer software implement algorithms for random sequence number generation. However, if the sequence numbers are compromised, packets can be sent to the victim(s).

Man In the Middle (MITM) attacks employ spoofing techniques. In a MITM attack, an attacker intercepts a legitimate communication between two communicating parties. The attacker then controls the flow of communication and can eliminate or alter the information sent by one of the original participants without the knowledge of either the original sender or the recipient.

A *Denial of Service* (DOS) Attack can be made more effective using IP Spoofing. An attacker will spoof source IP addresses in the offending traffic to make tracing and stopping the attack as difficult as possible. In some cases multiple compromised hosts participate in the attack, and all send spoofed traffic, complicating the task of quickly blocking the offending traffic. A DOS attack enabled by IP Spoofing is considered difficult to defend against because the enabling vulnerability is inherent to the design of the TCP/IP suite. Packet filtering by firewalls, encryption and authentication techniques, which are described below, can be employed to reduce the risk and impact of TCP/IP Spoofing attacks.

3.4.2 SYN Flooding

Assume a client process is attempting to perform the aforementioned handshaking process with a server. One point where an attacker can interfere with this process is where the server system has sent an acknowledgment (SYN/ACK) back to client but has not yet received the ACK message. This incomplete handshaking process results in what is referred to as a “half-open” or “partially open” connection.

The server operating system has built in its primary memory a data structure describing all pending connections. This data structure is of finite size, and it can be made to overflow by intentionally creating too many partially open connections.

The attacking system sends SYN messages to the victim server system. These messages appear to be legitimate connection attempts but in fact represent attempts to connect by a client system that is unable to respond to the SYN-ACK messages, or simply does not exist. In either case, the final ACK message will never be sent to the victim server system.

The data structure that stores connection information, and which records state about partially open connections, will eventually deplete. At that point the server system will be unable to accept any new incoming connections until memory is reclaimed and made available to record state about new connection attempts.

There is a timer associated with a pending connection, which “times out” if the connection does not complete. Therefore, half-open connections forced by the attacker will eventually expire and the server under attack will recover. However, the attacking system can simply continue sending IP-spoofed packets requesting new connections faster than the victim system can expire the pending connections.

The attack does not affect existing incoming connections nor does it affect the ability to originate outgoing network connections. However, in some cases, the victim of such an attack will have difficulty in accepting any new incoming network connection. This results in legitimate connections being refused by the server, hence their services become unavailable. Depending on the implementation, some servers may simply crash.

The location of the attacking system is concealed because the source addresses in the SYN packets are spoofed. When the packet arrives at the victim server system, there is no way to determine its source because packet switched networks forward packets based on destination address.

Several attempts have been proposed to defend against SYN floods. One reduces the memory allocated to record an attempted TCP connection, e.g., 16 bytes. This forces the attacker to send much more SYN traffic to exhaust the victim operating system's memory.

Another technique an attacked host can implement is to allocate no space when a SYN packet is received. Instead the attacked host returns a SYN/ACK with a sequence number that is an encoding of information appearing in the SYN packet. If an ACK packet arrives, representing a non-malicious attempt to complete the handshaking process, the host receiving the packet can extract information about the original SYN and only then allocation the data structure necessary to maintain the connection.

SYN flooding, and how it is enabled by TCP/IP Spoofing, is described in detail in [12].

3.4.3 Port Scanning

An attacker can easily compromise a host system if an attacker can gain access. Attackers gain access by scanning devices on the network for vulnerabilities, then exploiting them. "Port scanning" [25] is the term used for the manual or automated process of port reconnaissance.

Ports are a transport layer concept. A port number functionally determines a process on the host that was assigned the port number by the operating system. An IP address and port number functionally determine the process and the network device (via its IP address) that is hosting the process.

An attacker interested in a particular network may attempt to obtain information about that network and scan for vulnerabilities. Some attackers will attempt to scan large ranges of IP address searching for machines to exploit. Port scanners are useful defense tools in that they can be used to identify vulnerable systems within an organization’s network architecture. There are port scanners available for download at no charge. One of the most popular and effective is called “Network Mapper” or “*nmap*” [33], which will provide a detailed listing of all open ports.

3.5 Attacks Against the Operating System

This chapter focuses on attacks against the network protocol layers. In modern computing systems the software that implements the network protocol layers is intricately coupled with the computer’s operating system. Attacks against the operating system directly affect the implementation of the network protocol layers. In this section we describe attacks against the operating system that are devastating to the operation of the network protocol layers.

3.5.1 Rootkits

Rootkits [40] are an especially dangerous form of malicious software. At present, rootkits can be partitioned into two classes, namely user-mode rootkits and kernel-mode rootkits. User-mode rootkits replace normal operating system components, including the programs and commands that users and administrators rely on, with malicious versions that give the attacker remote access to the machine and mask the attacker’s presence with fraudulent components that appear normal. The information returned to the user of the malicious versions conceals the rootkits presence.

Kernel-mode rootkits compromise the operating system’s kernel. The kernel operating system layer resides between user programs and the hardware of the machine, controlling which programs execute, allocating memory, interacting with the hard drive and accessing network hardware. By compromising the kernel, attackers can use the system to perform malicious actions such as hiding files, processes and network activities. The attacker creates an artificial reality where the operating system appears intact to, and under the control of, system administrators and users. However, in reality the machine is completely compromised and under the full control of the attacker.

Note that attackers do not use the use the root kit to gain access. The attacker must have previously gained access to the victim’s computer by other means and subsequently installed the rootkit. Alternatively, the attacker must have tricked the victim into installing the rootkit himself. A common attack method is to trick the user into installing a backdoor allowing remote access, perhaps with a malicious e-mail attachment. Once access is obtained, the attacker can install the rootkit.

3.5.2 Shell Shoveling and Relays

One objective of an attacker is to possess the capability to remotely execute arbitrary commands on a victim's computer. "Shell shoveling" is a common term used to refer to the acquisition of this capability. The following example illustrates one way shell shoveling could be configured.

Assume that an attacker has been able to install the popular "netcat" utility on the victim's machine, or that the victim had already installed it. The attacker could then use netcat to monitor TCP port 80, which is usually open to allow HTTP traffic, to accept remote commands from the attacker. These commands would then be executed on the victim's machine, perhaps by being piped through a command shell. Any output resulting from command execution could be directed out of the victim's machine using port 25, which is usually open to allow e-mail traffic. The result of the process above is that a remote command shell is "shoveled" to the attacker.

An example command that could be executed on a victim's Windows machine (assuming netcat was installed) could appear similar to:

"nc attacker.com 80 | cmd.exe | nc attacker.com 25"

If the intended victim was a Unix machine, and assuming that popular "Xterm" utility and TCP traffic on port 6000 was unrestricted, a command similar to the one below could be executed:

"xterm -display attacker.com:0.0 &"

3.5.3 Authentication Mechanism Attack

Authentication mechanisms are a high profile target of attackers. Once compromised, the attacker can perform all system functions permitted by the authorized user without experiencing interference by other security mechanisms. Once the breach has occurred, the attacker can cause havoc with impunity.

One of the weakest types, but most widely deployed, authentication mechanisms is password systems. The user presents a login, which is usually well known or easily derived, followed by the user's secret password. The secrecy of the password is fundamental to the effectiveness of the authentication mechanism, which makes it of prime interest to an attacker.

There are several forms of *password cracking* [40] attacks. In one version, an attacker executes a program remotely that issues a series of login attempts in real time. This type of attack is easily repelled by systems that freeze accounts either permanently or temporarily after a series of incorrect password entry attempts. However, in a sense the attacker is still successful in causing the system to lock out a legitimate user. Alternatively, the system under attack can simply respond slowly to password entry during the login process, which will dramatically increase the time necessary to attempt a long list of possible passwords.

A more effective password cracking technique requires that the attacker first obtain the encrypted password file from a victim's machine. Most operating systems store a hashed form of the passwords on the local hard disk. In the case of Windows 2000/XP, this information is included in the SAM file. Once the password file is stolen, the attacker can present the file to a password cracking program, which has access to a lexicon, usually already in hashed forms (such as MD5 or SHA1).

The password-cracking tool attempts to decipher the passwords by comparing each entry with the encrypted value. If the encrypted values match, the hacker has identified a password. If the two values do not match, the tool continues through the entire dictionary, and can even attempt every combination of characters, includes numbers and special symbols. The default is a "brute force" attack that compares the hashed value of every combination of characters from the selected character set against the records in the password file. The limiting factor in how fast passwords can be cracked is how quickly guesses can be hashed and compared. Naturally, more powerful systems can process and test more potential passwords in less time.

A common defense against password cracking attacks is strongly enforced password policy. It may require users to devise passwords that are difficult to guess. A common password restriction is that they be at least eight characters long and include alphanumeric and special characters and not include dictionary terms. Since the complexity of a password may be expressed as $R{*}{*}L$ where the radix, R, is the size of the symbol set and L is the length, in most practical situations increasing L adds more security than increasing R.

For additional security, several automated tools are available that prevent users from setting their passwords to easy-to-guess values or dictionary terms or to reuse passwords without a waiting period. Unfortunately, forcing users to create and memorize lengthy complex passwords is problematic based on the inherent limitations on a person's memory. Many users do not change their passwords frequently or record them in secure places.

3.5.4 Buffer Overflow

A buffer is a contiguous area of memory occurring within the memory space allocated by the operating system to a running process. Buffers are created by computer programs. The programmer's intended use of the buffer is to store data of an expected size and format.

The run-time system of certain programming language environments do not perform bounds checking, or type checking, on the buffer automatically. The programmer is expected to include program instructions to perform the check when necessary. In many software components these checks do not appear. Consequently, a buffer can be made to overflow in the same way as a bucket of water can be made to overflow. The technique of deliberately overflowing a buffer to compromise a software system is known as a *buffer overflow attack*.

In its simplest form, a buffer may be thought of as in-stream memory allocated by a process or imminent short-term use. In the normal operation of the program, the next instruction after the buffer will be either the next executable instruction or a pointer thereto. If a buffer overflows, it will necessarily obliterate that instruction or pointer—hence the vulnerability [5].

One common approach to buffer overflows involves filling the top part of the buffer with “NO OP” instructions, which do precisely nothing, followed by some malicious code. The “spill over” will overwrite the intended next executable instruction with a pointer back to the top part of the buffer. The address of the pointer does not have to be exact, because the NO OP instructions will increment the memory address register until the first line of malicious code is reached.

This buffer overflow strategy has been in wide use for many years and relies upon the fact that there’s a lot of poorly constructed code deployed that doesn’t use bounds checking for buffer input control. It is important to note that buffer overflow problems are not restricted to operating system software. Any application layer software could potentially introduce a buffer overflow vulnerability.

3.6 Attacks Against the User

The user is considered by some to be the actual “top layer” of the TCP/IP network protocol stack. A naïve user can unknowingly circumvent any security feature built into the network protocol layer software. The infamous attacker and social engineer, Kevin Mitnick, believes that human factors are the weakest link in computer security [29]. Consequently, we consider direct attacks on the user as an indirect attack vector against the network layers. This section describes attacks against the user.

3.6.1 Attacks on Privacy and Anonymity

Attackers that wish to compromise information systems to perpetrate financial fraud often face technical obstacles. Naturally, the goal of network and system architects is to present as many obstacles as possible. However, a naïve user, as one of the “softest” targets, can be attacked to obtain sensitive information that will make other attack vectors unnecessary. This subsection addresses methods an attacker can use to acquire the identification of victim and the sensitive information itself.

3.6.1.1 Malicious Cookies.

A *cookie* [4,26] is a piece of information generated by a Web server and stored in the client's computer, in most cases without client intervention. They are intended to store state information regarding the interaction between a client and web server, often referred to as a *session*. Session information can include sensitive information about individuals such as account numbers, purchase information, user preferences and the history of HTML pages viewed.

When used as intended, cookies are useful and benign. The HTTP protocol provides no means to retain the state of the interaction. Coded session information is extracted from the cookie associated with the domain, avoiding unnecessary data entry by the user.

Web servers automatically gain access to cookies that reference the web server's domain whenever the user establishes a connection. Faults in some browsers allowed cookies to be stolen revealing the session information. An attacker could then use the stolen session information in his own interaction with the web server that the victim had been accessing. This is referred to as *session cloning*. The attacker could use the cloned session to fraudulently make purchases, transfer funds, change shipping and billing addresses as if he was the authorized user.

A less severe form of cookie abuse can attack a victim's privacy. A third-party, other than the client or the web server explicitly accessed by the client, can obtain the cookies, and hence the sensitive information, if permitted by the operator of the web server. This can occur, without the permission of the user, when a HTML (web) page explicitly referenced by the client reference a web page served by the third party's web server.

The most common scenario occurs when a Internet advertising firm who contracts to provide banners on others web sites exploits this opportunity to collect user cookies from many web sites. It can analyze these collected cookies to track a victim's web page access across multiple web sites. The web site access data, and any sensitive information extracted from the cookie, can be used to generate profiles of user behavior. This entire process will usually be unbeknownst to the user.

3.6.1.2 Web Bugs.

A web bug [8] is an HTML image tag occurring within a web page that results in malicious actions in addition to simply downloading and rendering an image. The image download request can include encoded personal information. This encoded information can identify a user, a user's email address or other sensitive data.

Web bugs are purposely deployed to covertly collect data that a user may wish to remain private. Advertising companies and large corporations have used web bugs to covertly collect marketing data. In many cases, the graphic specified to be downloaded is comprised of only one pixel, which would make it virtually invisible, especially if the color matches that of the background of the web page being rendered by the browser.

Some examples of the type of information that can be harvested using web bugs are:

- The IP address of the computer the victim is using to view the document.
- The date and time the page was viewed.
- The browser type and monitor resolution.
- The browser type, which can be used to infer the operating system type.
- The value of a cookie from the domain providing the image if previously set.

A more intrusive use of web bugs is including them within HTML e-mail. When the e-mail is viewed and the HTML within the e-mail message is rendered, the image request, containing personal information, is provided to the attacker. This technique allows an attacker, for example a spammer, to automatically validate that an e-mail address the spammer has used is valid and in use by the victim.

This attack can be enabled by any software application utilized by the victim that retrieves images from remote locations for rendering. Office productivity software packages, which are now Internet-enabled, are a current target for those experimenting with web bugs. An example motivation would be to determine who read a document, which might be relevant if the document was not in the public domain and it was questioned who released the document.

Most information available on the World Wide Web (WWW) is represented in a general, uniform format called Hypertext Markup Language (HTML) and is communicated upon request using a standard protocol called Hypertext Transport Protocol (HTTP). The software system that provides HTML documents to a client's web browser via the Internet is termed a web server.

3.6.2 Social Engineering

To launch a social engineering attack [27], an attacker uses social interaction to obtain or compromise information about an organization or its computer systems. An attacker may seem unassuming and respectable, possibly claiming to be a new employee, repair person, or researcher and even offering credentials to support that identity. However, by aggregating information obtained in each interaction, the attacker may be able to piece together enough information to infiltrate an organization's network. If an attacker is not able to gather enough information from one source, he or she may contact another source within the same organization and utilize the information from the first source to gain credibility.

Attackers with both technical and social engineering skills have been especially effective against large organizations that are believed to have sufficient security systems in place. The attacker will attempt to identify vulnerabilities in both computer and network systems, but will also target physical security and good natured, but naïve, employees. Ref. [29] describes these concepts in detail.

3.6.3 Phishing

The term "*phishing*" [24] is used to describe a class of attack made against a computer user. The technique used to gain personal and financial information usually for purposes of identity theft. It involves the use of fraudulent e-mail messages and corporate web pages that appear to come from legitimate businesses.

Authentic-looking messages are designed to fool recipients into divulging personal data such as account numbers and passwords, one's mother's maiden name, credit card numbers and Social Security numbers. When users respond with the requested information, attackers can use it to gain access to the accounts. Phishing can be viewed as an advanced, automated form of social engineering attack.

Despite the passage of the federal Identity Theft and Assumption Deterrence Act of 1998, phishing attacks are increasing in number and effectiveness. The creation of authentic-looking messages is trivial as the attacker can easily reproduce all of the text and graphics necessary to create the fraudulent messages from the legitimate web sites. A common scam is to send e-mail that purports to originate from a legitimate organization with whom the user has an existing business relationship. The e-mail insists that an authentication process be performed. Part of this process will require that the victim provide login credentials. Once the victim has unknowingly provided the credentials to the attacker, the attacker that uses these credentials to perpetrate a fraud.

3.6.4 E-mail Spoofing

Spoofing is the deliberate attempt to mislead or defraud someone by misrepresenting one's true identity. There are several forms of spoofing. Each form can be distinguished by the type of communication employed to mislead the victim. In previous sections of this chapter we addressed spoofing at the data link layer (ARP spoofing) and at the network layer (IP spoofing). In this section we address e-mail spoofing.

E-mail spoofing [13] may occur in different forms, but all have a similar result: a user receives email that appears to have originated from one source when it actually was sent from another source. Email spoofing is often an attempt to deceive a victim into making a statement that may damage his or her reputation, or into releasing sensitive information. It is also commonly used as the precursor for a phishing attack.

Some typical examples of spoofed email are those claiming to be from a system administrator requesting users to change their passwords to a specified string and threatening to suspend their account if they do not comply or email purporting to be from a person in authority requesting a victim to provide sensitive information.

E-mail spoofing is technically quite easy to perform because common e-mail protocols, such as Simple Mail Transfer Protocol (SMTP), lack a source address authentication mechanism. This allows e-mail to be sent that contains any source e-mail address desired by the attacker.

3.6.5 Keystroke Loggers

A keystroke logger records every keystroke a user types on a specific computer's keyboard. As a hardware device, a keystroke logger is a small device that interfaces between the keyboard and computer. Since most workstation keyboards connect to the rear of the computer, the device can often reside covertly on the victim's machine. The keystroke logger captures all keystroke performed by the victim. Unless the keystroke logger has a covert data transfer utility, the device must be physically removed to retrieve the captured keystrokes.

A keystroke logger program does not require hardware to be installed to function. An attacker with physical access to the computer can install it. The software will covertly record the keystrokes either for later removal from the victim's computer using a storage device, or alternatively, it can transmit the recorded keystrokes to the attacker over the Internet.

Some keystroke loggers do not require physical access to the victim's machine for installation if the machine is connected to a network. The logger can be installed remotely by at attacker, or can fool the victim into installing it. The software will then transmit the recorded keystrokes to the attacker over the Internet.

3.6.6 Spyware

Spyware [43] is a label given to software that executes on a victim's computer and covertly transmits data collected about the user to another party. In more aggressive situations, the software is installed without the victim's permission or knowledge. In less aggressive situations, a victim will be notified when the spyware is to be installed, and is even given an opportunity to block the installation. However, in many of these cases the notification is often very obscure, for example, it appears as complicated legal language embedded within a long privacy policy. In other cases, the spyware software is coupled with other software that the victim wishes to install. When the victim installs the desired software, the spyware is also covertly installed.

Spyware, like legitimate software, executes with the same permissions as the user who installed it. Therefore, the spyware possesses the capability of performing a wide range of malicious actions. Some types of spyware, such as adware, browser helper objects and dialers, as well as malicious activity they perform, are described below.

Spyware can track the history of web sites visited by the victim and then transmit this information to the attacker. Spyware can also collect demographic information about the user, such as age, geographic location, and gender. There is nothing to stop spyware from collecting, and then reporting, names, social security numbers, credit card numbers, and other data that may reside on a computer or had been entered into a web form.

Spyware is sometimes deployed in conjunction with useful software, often called *freeware*, that the user purposely installs but does not have to pay for. In this scenario the freeware vendor partners with an on-line advertising firm that provides the spyware to create a revenue model. The vendor of the freeware distributes the spyware along with the freeware. When the user installs the freeware the spyware is also installed. When the freeware is executed, the spyware, which is concurrently executed, will download banners advertisements from a web site specified by the attacker that are then displayed as part of the user interface of the freeware. The freeware vendor is compensated for its part in this process.

In some cases the spyware will simply use the freeware installation as a means to become installed. It will then display the banner advertisements autonomously, i.e., without requiring the execution of the freeware. Spyware that acts in the manner described above is sometimes referred to as *adware*.

A variant form of spyware is termed a *Browser Helper Object* (BHO). A BHO is spyware that parasitically couples itself to a common web browser, although it requires an unknowing user's permission to do so. It employs a code extension mechanism inherent in the browser to allow the BHO to execute when the browser is executed. Although the BHO can offer, and then deliver, functionality requested by the user, the BHO can implement additional, malicious, functionality.

There are many malicious actions that may be performed by the BHO. It may monitor the websites visited by the victim and transmit the data to the attacker. It may intercept requests for specific web pages and replace them with those specified by the attacker, which is an attack known as *browser hijacking*. The BHO may replace the result of a web search with that specified by the attacker.

A *dialer* is spyware variant that covertly changes the dial-up connection setting of communication software. When the victim uses his modem to connect to his local Internet service provider, the communication software instead calls a high cost-per-minute service telephone number such as a long distance or other toll number.

One of the most insidious forms of spyware is presented to the victim as a spyware removal utility. When the victim installs the malicious "spyware removal" utility, the software exhibits precisely the same behavior that the victim has intended to prevent.

3.7 Large Scale Attack Techniques

There are many methods an attacker can use to attack a computer system. However, some techniques are effective for launching an attack on a large number of systems. In certain cases, the large number of compromised systems can be forcibly converted into an army of attackers and directed to launch a particularly debilitating attack on one or more additional systems. These techniques are discussed in this section because they may attack or affect multiple layers of the TCP/IP stack.

3.7.1 Virus

A virus [10] is software that attaches itself to a seemingly innocuous file, either executable or non-executable, with the deliberate intention of replicating itself. Most viruses perform other, often malicious, functions. A virus requires human action to propagate, such as opening an infected e-mail or executing an infected software application.

3.7.1.1 Boot Virus. Boot viruses place themselves in the disk sector whose code the machine will automatically execute during the boot process. When an infected machine boots, the virus loads and runs. After a boot virus finishes loading, it will usually load the original boot code, which it had previously moved to another location, to ensure the machine appears to boot normally.

3.7.1.2 File Virus. File viruses attach to files containing executable or interpretable code. When the infected code is executed the virus code executes. Usually the virus code is added in such a way that it executes first. After the virus code has finished loading and executing, it will normally load and execute the original program it has infected, or call the function it intercepted, so as to not arouse the victim's suspicion.

3.7.1.3 Macro Virus. Macro viruses are a specialization of file virus. They copy their malicious macros to templates and/or other application document files, such as those modified by an office productivity software suite. Early versions would place themselves in the macro code that was the first to execute when infected templates or documents were opened. However other macros require the user to invoke an application command, which runs the malicious macro.

3.7.1.4 Script Virus. Script viruses confuse the victim because they do not appear to be executable files. Standalone Visual Basic Script (VBS) and JavaScript (JS) programs have suffixes that a naïve user does not associate with an executable program. Consequently, script viruses became a popular virus type for attackers launching their attack using mass e-mailing.

3.7.1.5 Image Virus.

An image virus attaches itself to compressed image files, e.g., JPEG. Merely viewing the image with a vulnerable web browser could invoke a buffer overflow and activate the virus. The infected image could be distributed via e-mail. It could also be distributed via its presence on a web site.

3.7.1.6 Companion Virus.

Companion viruses do not directly infect boot sectors or executables. Instead, a companion virus simply assumes the same name as a legitimate program but with an extension that will cause an operating system to give it higher precedence for execution. When the file is involved at the command line without the extension, the victim will expect the legitimate program to execute but instead the companion virus will execute.

3.7.2 Worm

A worm [39] is self-replicating malicious software that propagates across a network, spreading from vulnerable system to vulnerable system, without human intervention. Because worms use one set of victim machines to scan for and exploit new victims, and then allow these victims to perform the same task, worms propagate exponentially. Many of the worms released in the last decade have spread extremely quickly throughout the Internet despite possessing inefficient targeting methods.

3.7.2.1 Flash Worm.

A “flash worm” accelerates the propagation rate by pre-scanning the Internet for vulnerable systems. Through automated scanning techniques from static machines, an attacker can find thousands and thousands of vulnerable systems before actually releasing the worm. The attacker then initializes the worm with the IP addresses of the systems that it has determined in advance possess the vulnerability.

As the worm spreads, the addresses of these vulnerable systems would be split up among the segments of the worm propagating across the network. By using a large initial set of vulnerable systems, it is believed that an attacker could infect almost all vulnerable systems on the Internet before any significant defense could be mounted. Fortunately, no flash worm has been released as of the time of this writing.

3.7.2.2 Multi-Platform Worms.

Most worms that have been created and released into the wild were constructed to attack machines running a single software architecture, e.g., Microsoft Windows, Unix or Linux. It is envisioned that a more destructive worm will be created that will contain exploits for a variety of popular operating systems. Such worms will require security personnel and system administrators to apply patches in a coordinated fashion to many types of machines, which will be a more complex process and require more time. This delay will allow the worm to cause more damage.

3.7.2.3 Polymorphic Worms. Unlike recent worms, which have been relatively easy for security experts to detect and determine their functionality, a polymorphic worm will invent new disguises for itself whenever it compromises a new machine. Detection of a polymorphic worm is more difficult because the worm restructures its code each time it executes. A polymorphic worm will obscure, or encrypt, its payload, hence concealing its functionality. Reverse engineering the worm to obtain its functionality is more difficult. The extra delay needed for the analysis of the worm will allow the worm more time to propagate before adequate defenses are conceived.

3.7.2.4 Zero-Day Exploit Worms. The usual defense against a worm is to patch operating systems that possess a vulnerability that a worm may exploit. This method has been reasonably effective since the time necessary to create a worm will in many cases surpass the time necessary to patch. Unfortunately, this is not always the case since important business applications may be adversely affected by the installation of a patch, hence analysis and testing must first be performed before the patch can be installed.

Another complications is that new vulnerabilities are discovered almost daily. An attacker may discover a significant vulnerability and devise a worm that exploits it before a patch is created. Security professionals will either not have a patch or will not be able to install a patch in time to block the worm. Because of the lack of defense preparation time, these worm attacks are considered part of a class known as “zero-day exploits.”

3.7.3 Trojans and Backdoors

A *Trojan* is a malicious software program that creates a mechanism by which the attacker can remotely access and control the victim’s computer. The mechanism created for remote access and control is referred to as a “*backdoor*.” The Trojan may be executed on the victim’s computer covertly by the attacker if the attacker can compromise the victim’s computer to gain access. However, attackers use numerous ways to trick a user into executing a Trojan.

3.7.4 Denial of Service Attacks

The objective of a *Denial of Service* (DOS) attack is to make one or more computer resources unavailable to perform the function for which they are intended. A computer or other network device can launch a DOS attack. When an attacker can acquire the use of multiple hardware devices, perhaps in diverse geographic locations, the attacker can launch a *Distributed* Denial of Service (DDOS) attack.

One classic type of DOS attack is to generate a massive amount of network traffic addressed to the victim's host or network. This host will exhaust its memory resources attempting to consume the traffic. Consequently, it will become unavailable to legitimate traffic. Alternatively, the victim's network will become completely congested passing network traffic. The network will be unable to accept legitimate traffic. This will again result in legitimate traffic becoming impeded for blocked.

When only a single hardware device is used to launch the DOS attack, the effects of the attack can often be mitigated relatively quickly. Once the source of the DOS attack is identified, network traffic streaming in from the attacker can be blocked. The traffic can be blocked at the destination using firewall technology. Alternatively, an Internet service provider (ISP) can be notified and directed to block the traffic prior to arrival at the victim's network or host "upstream."

A DDOS attack is more difficult to mitigate. There could be from several to tens of thousands of hosts, which are compromised and controlled by the attacker, involved in the launching of the attack. In this case, it is very difficult for the victim's security professionals to repel the attack and still ensure that its computers and network can respond effective to legitimate traffic and requests.

There are several methods by which an attacker can create an "army" of computers to launch a DDOS attack. A worm can be created whose payload is a Trojan that creates a backdoor to accept commands from the attacker. Alternatively, a virus can be spread that tricks a victim into installing a Trojan that creates the backdoor.

Note that the creation of the backdoor is not strictly necessary. An attacker can preprogram the Trojan to launch an attack at a predetermined time. In this manner, the Trojan, and the entire DDOS attack, become "fire and forget," to use military parlance.

Some attackers create a hierarchy of command to control their army of compromised machines. These compromised machines that comprise the army are referred to as "zombies." An attacker may not wish to directly communicate with every zombie as it could risk compromising the attacker my revealing his/her IP address or geographic location. Instead, the attacker issues commands to a few delegates, who can be viewed as "senior officers." Only these delegates, which are themselves compromised machines, communicate direct with the remainder of the zombie army. Note that there is no technical reason why the hierarchy of command must be limited to two levels. The more levels involved, the more difficult it becomes to identify the attacker.

4. Defenses

This section surveys the different classes of technologies that have been developed to defend against the attacks described in the previous section.

4.1 Authentication

Authentication is the process of determining whether someone (or something) is in fact, who he (or it) declares itself to be. A principal is the party whose identity is verified. The verifier is the party who demands assurance of the principal's identity.

As described in [14], authentication mechanisms can generally be categorized as verifying one or more of the following about an individual:

- *Something you are*. This includes biometrics techniques like fingerprint scans, retina scans, voiceprint analysis or handwriting analysis.
- *Something you know*. The classic example is a common password system.
- *Something you have*. This includes physical authentication mechanisms such as challenge-response lists, one-time pads, smart cards, and dongles.

Some systems combine these approaches to produce what is termed as two-factor authentication. In two-factor authentication schemes there is a security process in which the user provides two means of identification, one of which is typically a physical token, such as a card, and the other of which is typically something memorized, such as a code known only by the user. In this context, the two factors involved are "something you have" and "something you know."

A bank debit card is a common example of two-factor authentication. The customer must possess the card and also be able to provide a personal identification code (PIN), usually from memory. Some security procedures now require *three-factor authentication*, which involves possession of a physical token, a password along with biometric data.

Authentication is distinct from *authorization*, which is the process of giving principals access to system objects based on their identity. Authentication verifies an identity but does not address the access rights of the individual to a particular resource. Authorization is usually performed after the principal has been authenticated, and may be based on information local to the verifier, or based on authenticated statements made by others.

Modern computer systems provide service to multiple users and require the ability to accurately identify the user making a request. In traditional systems, the user's identity is verified by checking a password entered by the user during the login process. The system records the user's identity and relies on it to determine what operations may be performed.

There are several popular techniques for authentication, which are described below.

4.1.1 Password Systems

As described above, most systems that support an authentication mechanism implement a password system. Users authenticate by providing a login name and a password, which the system and user secretly share. This secrecy is meant to guarantee that the person attempting to authenticate, by knowing the login and the password, must be the person they purport to be. Password authentication relies on the "something you know" authentication principle.

It is common for users to release their login and password to other users for many reasons. Users will "lend" the use of their password to others for a variety of reasons. Login names are commonly public information and many times users choose passwords that are affiliated with some aspect of their lifestyle. Sometimes this might be as simple as their initials, spouse's name or a well-known sports figure. Sometimes a user will retain a default login password pair provided by the software vendor, e.g., Oracle Corporations default password of "scott/tiger" for the Oracle relational database management system. In a worst case, a user may simply select the password, "password."

Fortunately, there are system administration tools to prevent the use of weak passwords. Furthermore, organizations can use policy and training to encourage users to select strong passwords and to reduce the likelihood that they will share the password with others.

Password based authentication introduces additional vulnerabilities when employed over computer networks. This is because passwords sent across the network can be intercepted, and subsequently used by an eavesdropper, to impersonate the user. Although demonstrated to have some severe deficiencies, in many contexts a password system does provide a level of defense from attackers.

4.1.2 Kerberos

Kerberos [30] is a network authentication service. Its development was motivated by the need to replace "authentication by assertion" systems. In this unsophisticated authentication technique, a process simply asserts to a network service that it is running on behalf of a principal. This method has been employed in some versions of the Unix remote login program (“rlogin”).

Another unsophisticated authentication technique requires the principal to repeatedly enter a password for each access to a network service. Clearly this represents an inconvenience for the principal. More importantly, it is insecure when accessing services on remote machines. Assume the password was used to authenticate with the first machine. If that machine needed the services of a second machine, the (clear text) password would be needed again. It would have to pass (in clear text) through the first machine to get to the second. Here the clear text represents a vulnerability.

Kerberos is based on the key distribution model developed by Needham and Schroeder [31]. Although it relies on the “something you know” authentication principle, Kerberos eliminates the need to demonstrate possession of private or secret information by divulging the information itself. The principal presents a “ticket” that is issued by the Kerberos “authentication server” (AS). In concept, presenting the ticket is similar to presenting a drivers license as identification. In this case the authenticator is an issuing body, namely the local Department of Motor Vehicles, that is trusted to bind a ticket (the license) to an individual (the principal). The service then examines the ticket to verify the identity of the user. If verified, the principal is considered authenticated.

Both the principal and the service are required to have keys registered with the AS. The principal’s key is derived from a password selected by the principal. The service’s key is a randomly selected.

Ref. [46] describes Kerberos using the physical concept of “strongbox,” which is a metal box with a key lock. Assume that messages are written on paper and are “encrypted” by being locked in a strongbox by means of a key. A Principal is initialized by making its own secret physical key and registering a copy of this key with the AS. One the keys are registered, the Kerberos handshaking protocol proceeds as described below:

First the Principal sends a message to the AS indicating that the Principal would like to communicate with the Service. When the AS receives this message, it makes up two copies of a new key. This key is called the session key. It will be used in the direct communication exchange between the Principal and Server following authentication. The AS places one of the session keys in Strongbox 1, along with a piece of paper with the name Principal written on it. The AS locks Strong Box 1 using the Principal’s key.

Note that “Strong Box 1” is really just a metaphor for an encrypted message, and that the session key is really just a sequence of random bytes. If Strong Box 1 only contained the session key, then the principal would not be able to tell whether the response came back from the AS, or whether the decryption was successful. By placing the Service's name in Strong Box 1, the principal will be able to verify both that the strong box came from the AS, and that the decryption was successful.

The AS then places the second copy of the session key in a second strong box, namely Strong Box 2. The AS includes a piece of paper with "Principal" written on it in Strong Box 2. It locks the Strong Box 2 with the Service's key. The AS then returns both strong boxes to the Principal.

The Principal unlocks Strong Box 1 with the Principal's key, extracting the session key and the paper with the Service's name written on it. Note that the Principal can't open Strong Box 2 because it's locked with the service's key. Instead, the Principal puts a piece of paper with the current time written on it a new strong box, namely Strong Box 3, and locks the box with the session key. The Principal then hands both boxes, namely Strong Boxes 2 and 3 to the Service.

The Service opens Strong Box 2 with the Service's own key, extracting the session key and the paper with the Principal's name written on it. It then opens Strong Box 3 with the session key to extract the piece of paper with the current time on it. These items demonstrate the identity of the user.

The Kerberos system is especially noteworthy because it serves as the security foundation for modern versions of the Microsoft operating system.

4.1.3 Biometrics

Biometrics [23] refers to the automatic identification of a person based on his/her physiological or behavioral characteristics. This method of identification is preferred over traditional methods involving passwords and PINs (personal identification numbers). The person to be identified is required to be physically present at the point-of-identification and identification based on biometric techniques obviates the need to remember a password or carry a physical token. Biometric authentication relies on the "something you are" authentication principle.

With the increased use of computers interconnected by wide area computer networks, it is necessary to restrict access to sensitive/personal data. By replacing traditional authentication techniques, e.g., PINs, biometric techniques can potentially prevent unauthorized access to, or fraudulent use of, ATMs, cellular phones, smart cards, desktop PCs, workstations, and computer networks. PINs and passwords may be forgotten, and token-based methods of identification like passports and driver's licenses may be forged, stolen, or lost. Biometric systems are designed to overcome these deficiencies.

Various types of biometric systems are being used for real-time identification. The most popular are based on face, iris and fingerprint matching. However, there are other biometric systems that utilize retinal scan, speech, signatures and hand geometry. A biometric system is essentially a pattern recognition system that makes a personal identification by determining the authenticity of a user's specific physiological or behavioral characteristic.

Depending on the context, a biometric system can be either a verification (authentication) system or an identification system. Verification involves confirming or denying a person's claimed identity. In identification, one has to establish a person's identity. Each one of these approaches has its own complexities and could be addressed using a specific biometric system.

4.1.4 Physical Authentication

Mechanisms for physical authentication rely on the "something you are" authentication principle. The most common form of physical authentication is a hardware token such as a *smart card* or *dongles*.

Smart cards are small plastic cards that contain an embedded integrated circuit. Most are similar in size to a standard credit or debit card. One fundamental problem in securing computer systems is the need for tamper-resistant storage of encryption keys. Smart cards provide this functionality as well as the ability to upgrade and/or replace security technique that becomes compromised.

Early generation smart cards provided a memory function. Information, such as a unique identifier, could be stored on the card. Modern smart cards are essentially small computers. They contain embedded microprocessors, run their own operating systems and include non-volatile primary memory. The operating system for a smart card is usually installed on the card by the manufacturer and cannot be changed without sophisticated equipment, if at all. A unique serial number is programmed within each card.

Smart cards are broadly categorized based on their type of interface to the device they communicate with, which is called a *reader*. The two types of interfaces are referred to as "contact" and "contact-less." "Contact" smart cards use electrical contacts, placed on the cards in accordance with international standards, to allow them to be read by devices known as smart card readers. "Contactless" smart cards use low frequency radio waves to provide power and to communicate with smart card readers. Most contactless smart cards can be read from a distance of about fifteen centimeters even if still contained within a wallet or handbag.

The information stored within the card can be used for authentication in a variety of application domains, namely financial payments, critical health care information, immigration control and even as a unique identifier for Internet use. Due to their suitability as a physical authentication technology, smart card use is becoming more common. However, there are vulnerabilities associated with smart cards, which are described in [37].

A *dongle* is a hardware device with a similar function as a smart card except that there is no reader. Dongles typically directly connect to a computer via a serial or

USB port. Dongles have been traditionally used to prevent software privacy. The protected software will not operate properly on the computer unless the dongle is present.

4.2 Encryption

As described in [32], *encryption* and *decryption* allow two parties to communicate without any other party being able to view the communication. The sender encrypts, or scrambles, information before sending it. The receiver decrypts, or unscrambles, the information after receiving it. While in transit, the encrypted information is unintelligible to an interceptor, e.g., an attacker.

Encryption is the process of transforming information so it is unintelligible to anyone but the intended recipient. Decryption is the process of transforming encrypted information so that it is again intelligible. A cryptographic algorithm, also called a cipher, is a mathematical function used for encryption or decryption. In many cases, two related functions are employed, one for encryption and the other for decryption.

The ability to keep encrypted information secret is based not on the cryptographic algorithm, which is usually widely known, but on a number called a key that must be used with the algorithm to produce an encrypted result or to decrypt previously encrypted information. Decryption with the correct key is simple and relatively efficient. Decryption without the correct key is very difficult, and in some cases impossible for all practical purposes.

Related to encryption is the concept of *tamper detection*, which allows the recipient of information to verify that it has not been modified in transit. Any attempt to modify data or substitute a false message will be detected. *Nonrepudiation* prevents the sender of information from claiming at a later date that the information was never sent.

4.2.1 Symmetric-Key Encryption

With *symmetric-key encryption*, the encryption key can be calculated from the decryption key and vice versa, or more commonly, the same key is used for both encryption and decryption. Implementations of symmetric-key encryption can be highly efficient, so that users do not experience any significant time delay as a result of the encryption and decryption.

Symmetric-key encryption also provides a degree of authentication support, since information encrypted with one symmetric key cannot be decrypted with any other symmetric key. As long as the symmetric key is kept secret by the two parties using it to encrypt communications, each party can be sure that it is communicating with the other, assuming that the messages received are not garbled.

Symmetric-key encryption is effective only if the symmetric key is kept a secret by both parties involved. If a third party obtains the key, the communication's confidentiality and authentication is compromised. An unauthorized person with a symmetric key can decrypt messages sent with that key and encrypt new messages for transmission as if they came from one of the two authorized parties.

Symmetric-key encryption plays an important role in the SSL protocol, which is widely used for authentication, tamper detection, and encryption over TCP/IP networks. SSL also uses techniques of public-key encryption.

4.2.2 Public-Key Encryption

Public-key encryption, which is also called asymmetric encryption, involves a pair of keys, namely a public key and a private key, associated with an entity that needs to authenticate its identity electronically or to sign or encrypt data. Each public key is published, and the corresponding private key is kept secret. Data encrypted with a public key can be decrypted only with the corresponding private key. In general, to send encrypted data to someone, you encrypt the data with that person's public key, and the person receiving the encrypted data decrypts it with the corresponding private key.

Compared with symmetric-key encryption, public-key encryption requires more computation and is therefore not always appropriate for large amounts of data. However, public-key encryption can be used to send a symmetric key, which can then be used to encrypt additional data. This is the approach used by popular security protocols, e.g., the *Secure Sockets Layer* (SSL) protocol.

Data encrypted with a private key can be decrypted only using the corresponding public key. This would not be a desirable way to encrypt sensitive data, however, because it means that anyone with the public key, which is by definition published, could decrypt the data. Nevertheless, private-key encryption is useful, because it means one can use a private key to sign data with your digital signature. Party A's software can then use the appropriate public key to confirm that the message was signed with Party B's private key and that it hasn't been tampered with since being signed.

4.2.3 Digital Signatures

A digital signature [32] proves that electronic data was signed by the individual who claims to have signed it. It is analogous to a handwritten signature. Once a signer has signed data, it is difficult for the signer to deny doing so at a later time. This assumes that the private key has not been compromised or out of the owner's control.

This quality of digital signatures provides a high degree of non-repudiation, which means digital signatures make it difficult for the signer to deny having signed the data. In some legal jurisdictions, a digital signature may be as legally binding as a handwritten signature.

Digital signatures are used for tamper detection and authentication. They rely on a mathematical function called a message digest that produces a *one-way hash*. A one-way hash is a number of fixed length whose value is unique for the hashed data. Any change in the data, even deleting or altering a single character, results in a different hash value. The content of the hashed data cannot, for all practical purposes, be deduced from the hash alone.

It is possible to use a private key for encryption and the corresponding public key for decryption. Although this does not prevent an eavesdropper from intercepting, decrypting and viewing the data, it is a necessary part of digitally signing any data. Instead of encrypting the data itself, the signing software creates a one-way hash of the data, which then uses the private key to encrypt the hash. The encrypted hash, along with other information, such as the hashing algorithm, serves as a digital signature.

To validate the integrity of the data, the receiving software first uses the signer's public key to decrypt the hash. It then uses the same hashing algorithm that generated the original hash to generate a new one-way hash of the same data. Note that information about the hashing algorithm used is sent with the digital signature. Finally, the receiving software compares the new hash value against the original hash value.

If the two hash values do not match, the data may have been tampered with since it was signed, or alternatively, the signature may have been created with a private key that does not correspond to the public key presented by the signer. If the two hash values do match, the data has not changed since it was signed and the recipient can be certain that the public key used to decrypt the digital signature corresponds to the private key used to create the digital signature.

Note that confirming the identity of the signer, however, also requires some way of confirming that the public key really belongs to a particular person or other entity. *Certificate Authorities* (CAs) declare the validity of public keys.

4.3 Firewalls

A *firewall* [19] attempts to defend networked computers from intentional, hostile intrusion. A firewall may be a hardware device or a software program running on a secure host computer. It must have at least two network interfaces, one for the network it is intended to protect, and one for the network it is exposed to.

A firewall sits at the junction point, or gateway, between the two networks, which are often a private network and a public network such as the Internet. Early firewalls were simply routers. The firewall would segment a network into different physical subnetworks and attempt to prohibit damage that could spread from one subnet to another. Network firewalls borrow from the concept of a physical firewall, which insulates a physical building from an adjacent building in case of fire.

A firewall inspects all traffic routed between two (or more) networks. A firewall may allow all traffic through unless it meets certain criteria, or it may deny all traffic unless it meets certain criteria. Firewalls may discriminate based on type of traffic, or by source or destination network (IP) addresses and transport layer “ports.” They may employ complex rule sets that are applied to determine if the traffic should be allowed to pass. A modern firewall can filter both inbound and outbound traffic. It can also log all attempts to enter a private network, and trigger real-time alarms, when hostile or unauthorized entry is attempted or detected.

There are several broad categories of firewalls: packet filters, circuit level gateways, application level gateways and multilayer inspection firewalls. Packet filtering firewalls work at the network level or the IP layer of TCP/IP. They are usually part of a router. A router is a device that receives packets from one network and forwards them to another network.

In a packet filtering firewall each packet is compared to a set of criteria before it is forwarded. Depending on the packet and the criteria implemented using the logical rules, the firewall can drop the packet, forward it or send a message to the originator. Rules can reason with source and destination IP address, source and destination port number and protocol used and other protocol characteristics.

Circuit level gateways work at the session layer or the TCP layer. They monitor TCP handshaking packets to determine whether a requested session is legitimate. However, they do not monitor packets transmitted after the initial handshaking packets.

Application level gateways, also called *proxies*, are similar to circuit-level gateways except that they are application specific. Incoming or outgoing packets cannot access services for which there is no proxy. Application level gateways inspect the application level content of packets, which forces a proxy implementation to be very fast and efficient.

Multilayer inspection firewalls combine the benefits of the types of firewalls described above. They filter packets at the network layer, determine whether session packets are legitimate and analyze the contents of packets at the application layer.

4.4 Intrusion Detection Systems

Intrusion detection systems (IDSs) [22] detect intrusions made by attackers in host computers and networks. They alert individuals upon detection of an intrusion by sending out e-mail, pages or Simple Network Management Protocol (SNMP) traps.

This provides an administrator of the IDS with a notification of a possible security incident.

An IDS may automatically respond to an event by logging off a user, blocking traffic, closing sessions, disabling accounts, executing a program or by performing some other action. IDSs detect and respond to threats from both inside and outside a network or host computer.

Most IDSs are categorized as either "host-based" or "network-based." Host-Based IDS (HIDS) collect and analyze system, audit and event logs that originate on a host computer. They may also analyze patterns of executed commands, system calls made by applications, or of access to specific system resources by users and processes.

As opposed to monitoring the activities that take place on a particular host computer, network-based intrusion detection systems (NIDS) analyze data packets that traverse networks. Packets are inspected, and sometimes compared with empirical data, to analyze their legitimacy. NIDS detect attacks from outside a defender's network that attempt to abuse network resources or allow an attacker entry. However, NIDS can also be employed within a defender's network.

The TCP/IP packets that initiate an attack can be detected by a properly configured, administered and monitored NIDS. If the data within the packet is encrypted, then the effectiveness of a NIDS may be limited.

There are at many techniques that are employed by an IDS to detect an attacker. One technique directs the IDS to search for anomalous behavior. An IDS establishes a baseline of normal usage patterns, and activity that deviates from the pattern is reported as a possible intrusion. Usage patterns can be analyzed, such as profiling the programs that users execute daily. If the IDS detects that a warehouse clerk has begun accessing human resource applications, or is using a C++ compiler, the IDS would then alert its administrator.

An IDS may have access to a database of previously identified patterns of unauthorized behavior to predict and detect subsequent similar attempts. These specific patterns are called signatures. For a HIDS, one example signature could be a specific number of failed login attempts. For a NIDS, a signature could be a specific bit pattern that matches a section of a network packet header.

Another technique that an IDS can employ is to search for unauthorized modifications of specified files. Attempts at covert editing of files can be detected by computing a cryptographic hash at periodic intervals. The hash can be checking for changes over time. This process does not require constant monitoring by the administrator.

Some attackers perform reconnaissance of a victim's network for several months prior to launching the debilitating attack. A sophisticated IDS can correlate data obtain from the attacker's reconnaissance with other data to either forecast the attack or obtain better forensic evidence after the attack.

4.4.1 Honeypots and Honeynets

A *honeypot* [41] is an information system resource whose value lies in unauthorized or illicit use of that resource. It is a resource that has no authorized activity, nor any production value. When appearing as a device on a network, a honeypot should receive very little if any traffic because it has no legitimate activity or production function. Any interaction with a honeypot is unauthorized or malicious activity. Any connection attempt to a honeypot is most likely a probe or an attack.

The amount of log data collected by a honeypot will be significantly less that of a production system. However, the data collected will likely represent "true positives" in terms of representing malicious activity since honeypots do not rely on knowledge of preexisting attack types to provide useful information. Attacks and probes that can covertly evade network security devices like firewalls by utilizing encryption will still be detected by a honeypot.

Honeypots only detect those attacks that specifically involve the honeypot itself. An attack on a production system that does not affect the honeypot will not be detected by the honeypot. Furthermore, allowing the honeypot to interact with an attacker introduces the risk that the attacker will compromise the honeypot. This would allow the attacker to use the honeypot as a platform to attack other systems.

Honeypots can be aggregated to build *honeynets* [21]. A Honeynet is a network that contains one or more honeypots. As honeypots are not production systems, the honeynet also has neither production activity nor authorized services. Therefore, any interaction with a honeynet implies malicious or unauthorized activity.

A honeytoken [42] is any type of digital entity that performs a similar function to a honeypot. Specifically, a honeytoken is a digital resource whose value lies in the detectable, unauthorized use of that resource. Some examples could be a bogus social security number or credit card number, a dummy financial spreadsheet or word processing document, a database entry, or even a bogus login. If the honeytoken traverses the network or is found outside of the controlled network, it is clear that a system has been compromised.

4.5 Antivirus Technology

Antivirus software [28] is specifically written to defend a system against the threats posed by a virus. There are a number of techniques that antivirus software can employ to detect a virus. Some are presented in this section.

Signature scanning is employed by the majority of antivirus software programs. Signature scanning involves searching the target computer for a pattern that could indicate a virus. These patterns are referred to as *signatures*. These set of signatures are updated by the software vendors on a regular basis to ensure that antivirus scanners can detect the most recent virus strains. A deficiency of this approach is that the antivirus software cannot detect a threat made by a virus if it does not possess the virus' signature.

Heuristic scanning attempts to detect preexisting as well as new viruses by looking for general characteristics of malicious software. The primary advantage of this technique is that is does not rely on bit level signatures. It relies on general "rules of thumb" describing what code a virus might contain. However, this method does suffer from some weaknesses.

One weakness is a propensity to generate false positives. As the technique relies on heuristics, which are not always completely accurate, it may report legitimate software as being a virus if the software that implements the virus exhibits traits believed to be consistent with a virus. Another weakness is that the process of searching for traits is more difficult for the software to achieve than looking for a known bit pattern signature. Therefore, heuristic scanning can take significantly longer than bit pattern signature scanning.

Finally, the functions encoded in a new virus may not be recognized as malicious. If a new virus contains a function that has not been previously identified, the heuristic scanner will likely fail to detect it.

Behavior blocking is an antiviral technique that focuses on the behavior of a virus attack rather than the virus code itself. For example, if an application attempts to open a network port or delete a system file, a behavior blocking antivirus program could detect this as typical malicious activity, and then alert an administrator of a possible attack.

Most antivirus software available today employs a mixture of these techniques in their antivirus solutions in an attempt to improve the overall protection level.

4.6 Construction of Secure Software

Errors introduced during the software development process are a leading cause of software vulnerability [36]. The errors are not identified during the software testing process. Instead, they are identified after the software is deployed into many organizations around the world. If the software is a popular desktop application, the size of the deployed user base is in the millions.

In many cases the errors are discovered by irresponsible individuals or individuals with malicious intent. The information is then almost immediately distributed to attackers worldwide using the Internet. The attacker community then conspires to formulate strategies to exploit the errors before a software patch can be created and distributed by the software vendor, and then installed by system administrators. The process where software vendors and system administrators react to errors only after attackers identify and exploit software errors is referred to as the "penetrate and patch" software remediation process. The ineffectiveness of this process has resulted in temporary, but spectacular, worldwide software failures [16,40].

Consequently, some experts believe that one of the most important issues in defensive computing is the process in which software is constructed [47]. Defects introduced in the software construction process, and also the maintenance process, introduce security holes that require substantial resources to correct. The resources may be additional "add-on" system components like intrusion detection systems and firewalls. These resources may also include the cost of the development of software patches, their subsequent distribution, and finally the cost incurred by the end user to interrupt live systems to install and test following the installation of the patch. In a worse case the resources might also have to include massive funds to pay legal costs when either the software vendor, or enterprise whose security was breached, must respond to civil complaints due to the losses incurred by end users and large organizations. The techniques for building secure software fall outside of the scope of this chapter but are described in detail in [47].

Security engineering [1] is an emerging discipline focused on the construction of systems that will "remain dependable in the face of malice, error or mischance." It is a discipline that focuses on the tools, processes and methods needed to design, implement and test entire systems to ensure security, and to adapt existing systems as their environment evolves. Security engineering subsumes "system engineering" techniques, such as business process analysis, software engineering and testing because system engineering exclusively addresses error and mischance, but not malice. It is hoped that software professionals educated in security engineering principles will produce less vulnerable systems and therefore reduce or eliminate the "penetrate and patch" cycle.

The techniques introduced by security engineering require cross disciplinary knowledge in fields as diverse as cryptography, computer security, hardware tamper-resistance, formal methods, applied psychology, organizational methods, audit and the law. A full explanation would fall outside of the scope of this chapter. Interested readers are directed to [1].

In addition to the development of secure software, software systems should provide tools, utilities or build-in functionality to enable system administrators, and even end users, to easily monitor system security and perform security related tasks. For example, modern versions of the Microsoft Windows™ operating system contains a system management facility to provide access control features to restrict access to files and executable programs based on individual users and groups of these users. User passwords are also administered with this facility, which is quite useful.

Unfortunately, the facility does not allow the user to identify programs, either malicious or non-malicious, that have taken the liberty of configuring themselves to execute when Windows™ starts. This would be a useful feature of the facility because there are at least ten different methods a program can use to ensure it starts when Windows boots [34,20]. Determining if these methods have been employed requires inspection of special system folders, cryptically named system files and obscure registry entries as well as possessing knowledge of the esoteric details of the purpose and format of these data items. There is no easily assessable, inherent support for testing to determine which methods have been employed. Even technical users find it time consuming to investigate all of the possibilities without the use of 3rd party tools.

5. A Forecast of the Future

The intensity of the conflict in cyberspace is increasing. In fact, the scenario that is now unfolding presents the ideal environment for the development of a "weapon of mass *disruption*." The scenario is characterized by the following developments:

- There is an increase in the penetration of the Internet in every area of business, communications, government, financial systems, the military and our personal lives.
- There is an increase in the dependence on these Internet-based information systems.
- The complexity of computer systems is far surpassing the average personal ability to understand the vulnerabilities. In fact, this complexity is surpassing even professional software technicians' ability to understand, much less address, the vulnerabilities.
- Average people are using computers controlled by software with severe vulnerabilities that are connected by high-speed networks to attackers all over the world, and outside U.S. legal jurisdiction.
- Attackers are developing more sophisticated, and more damaging, attacks and are identifying more attack vectors.
- There is an increase in both the number and competence of attackers.

These developments amount to what could be called a "perfect storm," or ideal environment, for a complete breakdown of the U.S., and even worldwide, data communication infrastructure. This interruption has the potential to be devastating a variety of government, financial, medical, media and military organizations.

Perhaps even worse, once this breakdown occurs, it is conceivable that many people will lose faith in those professing the potential benefits of computers and the Internet. The promise of the Internet will remain unfulfilled. It may take many years to restore the public confidence in the technology, resulting in what could be called a technological "depression."

Although the scenario described above is quite bleak, there may be time to mount a concerted effort to ward off a major failure of the national data communication infrastructure. Unfortunately, it is not clear that there is enough momentum within our society, including those organizations that produce and use software and networks. In fact, as we concluded the writing of this chapter, the Washington Post reported that the government's "Cybersecurity" Chief had abruptly resigned from the Homeland Security Department after only serving one year [7]. The Washington Post reported that sources indicated that the government's lack of attention paid to computer security issues was a factor in the Chief's decision.

6. Defensive Precautions

Software professionals worldwide are becoming more aware of the threats against computer systems posed by attackers of all types. There are significant network security resources available to software professionals who wish to take steps to create a stronger defense. Hopefully, this will result in better security for enterprises, governments and Internet service providers.

As software professionals harden information infrastructure the weakest "link" in the security "chain" will be naïve end users and those end users who are aware of the threat but do not possess the technical knowledge to mount an adequate defense. Therefore, in this section we describe precautions that an Internet user can undertake to resist an attacker. These precautions are based on [35].

- *Install, use and maintain anti-virus* (*AV*)*, and spyware, software obtained from reputable vendors*. AV software searches your computer for the presence of malicious code. Uses should obtain and install AV software. They should also ensure that the software is actively monitoring their system. Finally, they must ensure that they are receiving the signatures, i.e., identifying characteristics, of the most recent threats. These signatures can usually be obtained via a paid subscription service from AV vendors.
- *Download and install software patches frequently*. It is difficult and expensive to produce software with zero defects. Consequently, most software products contain defects. This holds true for proprietary commercial products as well as those products that result from open source initiatives. Therefore, professional software producers provide a mechanism for the software to be modified to correct the defects, even after the software has been deployed. These modifications, called *patches*, can be obtained from the software producer. They can then be

installed manually. Some organizations provide extra software that automates this process, which causes patches to be downloaded and installed as soon as they become available. The speed of this process is important to prevent attackers from exploiting the defect before the software can be patched.

- *Beware of e-mail attachments*. E-mail attachments often contain an attacker's malicious executable. Antivirus software will sometimes detect this code and report it to the user. One heuristic for handling attachments is simply to never execute code that is contained within an attachment. Even if the attachment is presented as having been sent by a friend or co-worker, the sender's address may have been spoofed. Even if the sender's address can be verified, the attachment may still be unsafe as the sender, who you assume has the best intentions, may be unaware of malicious code within the attachment. Attachments that appear to be application specific non-executable, e.g., word processing documents, spreadsheets, photos, and movies, may in fact be, or contain, malicious code. One technique to reduce ones risk if a attachment has to be used is to download the supposed non-executable file, using the browser's "save as" feature, and then open the file using the intended application. If the file is not a legitimate file for the application, the application will normally report this.
- *Use extreme care when downloading and installing programs*. The Internet provides users with many opportunities to download software, often at no cost. Like e-mail attachments, any software that is downloaded from the Internet can also be malicious. A user must be convinced that the organization that directly produced, or distributed, the software is reputable, and also that the organization is technically capable of ensuring that the software is safe.
- *Use a software firewall program*. Software firewalls monitor the Internet traffic that enters into, and exits from, a user's computer. Rules can be established that block traffic in either direction that is believed to initiated directly, or indirectly, by an attacker. These rules can either be manually created, which is made easier via use of a friendly interface that is provided by the firewall. Alternatively, the firewall program may come preconfigured with rules that are effective for use with popular user software. Some users are unaware that egress traffic, namely traffic leaving their own computer, can be malicious or part of an attack.
- *User a hardware firewall*. Hardware firewalls can reduce ones profile and exposure to an attacker. Attacks launched against the user's computer must be made indirectly via the hardware firewall. Some inexpensive hardware firewalls can block attacks that involve particular TCP/IP ports and obscure the user's hardware and software configuration. Each firewall also has a small, hardened operating system residing in read-only memory, which makes the firewall more difficult to compromise.

- *Ensure there is a backup of all files*. Clearly no one wishes to lose expensive software, or more importantly, valuable, private documents, photos, movies, music and other digital assets residing on ones computer. Unfortunately, there are many ways that the files that contain these assets can be lost, compromised or destroyed. Computers can be stolen, users make errors managing folders, media of all types can fail and hackers can compromise systems. A simple, but often overlooked safety precaution is to make backup copies of all files. Operating system and application executable file are somewhat protected by default since a copy resides on the original media, or can usually be obtained from the vendor. However all files a user directly, or indirectly, creates must be protected via backup by a user or by a competent system administrator.
- *Use strong passwords and change them frequently*. Certain types of passwords are easy for an attacker to guess, with or without automated assistance. Some individuals use their own name, their own user name, or the default password provided by the software. Some users simply choose "password." These are trivial to guess. Even non-obvious passwords are easily identified using *password cracking* programs, which are freely available on the Internet. Longer passwords comprised of seemingly random strings of upper and lower case letters, numbers and special symbols are best. Even then, the password should be changed frequently and not used for multiple systems. This avoids the problem of one password becoming known to an attacker who can then use it to compromised other systems. Many systems, and ethical system administrators, will test the strength of a users password to ensure that it can withstand an attack.
- *Use a file encryption program and access control*. If network security is breached the resulting damage can be mitigated through the use of encryption and access control. Encrypting important files will deny the attacker from deriving much value from the files. Other than exhibiting the files to demonstrate a successful intrusion, there is little more the attacker can do with them. *Access control* restricts access to digital assets and other system resources. Specific users can be assigned different rights to perform different actions on different resources. If an attacker is successful in deceiving an authentication mechanism, the damage that the attacker can do is limited to what the impersonated user could do. The access control method quarantines the damage to the resources accessible to the compromised user.
- *Ensure you are communicating with who they say they are*. Attackers will usually pursue the path of least resistance when attempting to breach a system. Often that path is a naïve, trusting or exceptionally friendly employee. Without proper awareness, employees can be tricked into providing information that an attacker can exploit. Pausing to confirm the identity of someone requesting

information, by asking a colleague to confirm the individual's identify, or alternatively ending an incoming call to allow one to call back to confirm a phone number, are techniques that a user can employ to avoid becoming a victim.

The steps above will not guarantee that an attacker will never be successful. However, advising users to follow these precautions will reduce the chance of an attacker's success.

7. Conclusion

This chapter provided a survey of the electronic information battle currently being waged on the global data communication infrastructure. This survey differed from others in that it presented common, albeit not all, categories of attacks in a structure that is consistent with the TCP/IP protocol stack layers. Categories of attacks were presented as opposed to the myriad of esoteric technical details necessary to understand any one single instance of an individual attack.

After presenting categories of attacks, we presented an overview of accepted techniques that have been created to defend against the various attack categories. A list of precautions that an average user can employ to resist an attacker was provided. We then presented our somewhat pessimistic view of the future of computer and network security with the hope of challenging others within the software community to address this important topic.

References

[1] Anderson R.J., *Security Engineering: A Guide to Building Dependable Distributed Systems*, Wiley, ISBN 0471389226, January, 2001.

[2] Alger J., "Introduction to information warfare", in: Schwartau W. (Ed.), *Information Warfare, Cyberterrorism: Protecting Your Personal Security in the Information Age*, second ed., Thunder's Month Press, New York, 1996, pp. 8–14.

[3] Bellovin S.M., "Security problems in the TCP/IP protocol suite", *Computer Communications Review* **2** (19) (April, 1989) 32–48.

[4] Berghel H., "Caustic cookies", Digital Village, *Communications of the ACM* (April, 2001) 19–22.

[5] Berghel H., "The Code Red worm", *Communications of the ACM* (November, 2001) 15–19.

[6] Berghel H., "Malware month of the millennium", *Communications of the ACM* (December, 2003) 15–19.

[7] Bridis T., "U.S. Cybersecurity Chief Resigns, Washington Post (.com)", The Associated Press, Friday, October 1, 2004.

[8] Web bugs, URL: http://www.leave-me-alone.com/webbugs.htm, 2002.
[9] Bush V., "As we may think", *The Atlantic Monthly* (July, 1945).
[10] Computer Associates International, Inc., "Computer viruses—an introduction", URL: http://www3.ca.com/solutions/collateral.asp?CID=33330&ID=897&CCT=, 2004.
[11] "CERT advisory CA-1998-01 smurf IP denial-of-service attacks", URL: http://www.cert.org/advisories/CA-1998-01.html, March 13, 2000.
[12] "CERT advisory CA-1996-21 TCP SYN flooding and IP spoofing attacks", URL: http://www.cert.org/advisories/CA-1996-21.html, November 29, 2000.
[13] CERT Coordination Center, "Spoofed/forged email", URL: http://www.cert.org/tech_tips/email_spoofing.html, 2002.
[14] Chapman B.D., Zwicky E.D., *Building Internet Firewalls*, first ed., O'Reilly & Associates Publishers, ISBN 1-56592-124-0, November, 1995.
[15] "daemon9", *Phrack Magazine* **7** (49) (August, 1996), URL: http://www.phrack.org/show.php?p=49&a=6.
[16] Denning D., *Information Warfare and Security*, Addison–Wesley, ISBN 0-201-43303-6, 1999.
[17] Dhar S., "SwitchSniff", *Linux Journal* (March 05, 2002), online, URL: http://www.linuxjournal.com/article.php?sid=5869.
[18] Forouzan B., *TCP/IP Protocol Suite*, second ed., McGraw-Hill Higher Education, ISBN 0-07-119962-4, 2003.
[19] FreeBSD Handbook, The FreeBSD Documentation Project, Copyright 2004.
[20] Gralla P., *Windows XP Hacks*, first ed., O'Reilly, ISBN 0-596-00511-3, August, 2003.
[21] Honeynet Project, "Know your enemy: Honeynets"; URL: http://project.honeynet.org/papers/honeynet/index.html, November, 2003.
[22] Innella P., McMillan O., "An introduction to intrusion detection systems", URL: http://www.securityfocus.com/infocus/1520, December, 2001.
[23] Jain A.K., Pankanti S., Prabhakar S., Hong L., Ross A., Wayman J.L., "Biometrics: a grand challenge", in: *Proc. International Conference on Pattern Recognition (ICPR)*, vol. II, Cambridge, UK, August, 2004, pp. 935–942.
[24] Kay R., "Phishing", URL: Computer World Online, http://www.computerworld.com/securitytopics/security/story/0,10801,89096,00.html, January, 2004.
[25] Liska A., *Network Security: Understanding Types of Attacks*, Prentice Hall Publishers, 2003.
[26] Mayer-Schönberger V., "The cookie concept", URL: http://www.cookiecentral.com/c_concept.htm.
[27] McDowell M., "Avoiding social engineering and phishing attacks", URL: http://www.us-cert.gov/cas/tips/ST04-014.html, July, 2004.
[28] Microsoft Corporation, "The antivirus defense-in-depth guide", URL: http://www.microsoft.com/technet/security/guidance/avdind_0.mspx, August, 2004.
[29] Mitnick K., *The Art of Deception*, Wiley Publishing, 2002.
[30] Neuman B.C., Ts'o T., "Kerberos: an authentication service for computer networks", *IEEE Communications* **32** (9) (September, 1994) 33–38.
[31] Needham R.M., Schroeder M.D., "Using encryption for authentication in large networks of computers", *Communications of the ACM* **21** (12) (1978) 993–999.

[32] Netscape Corporation, "Introduction to public-key cryptography", URL: http://developer.netscape.com/docs/manuals/security/pkin/contents.htm, October, 1998.
[33] "Network Mapper", URL: http://www.insecure.org/nmap/.
[34] Otey M., "Windows program startup locations", *Windows IT Pro Magazine*, URL: http://www.windowsitpro.com/Articles/ArticleID/27100/27100.html, December, 2002.
[35] Rogers L., *Home Computer Security*, Software Engineering Institute, Carnegie Mellon University, 2002, URL: http://www.cert.org/homeusers/HomeComputerSecurity/.
[36] SANS Institute, "The twenty most critical Internet security vulnerabilities (updated) $\sim$ The Experts Consensus, version 5.0", URL: http://www.sans.org/top20/, October, 2004.
[37] Schneier B., Shostack A., "Breaking up is hard to do: modeling security threats for smart cards", in: *USENIX Workshop on Smart Card Technology*, USENIX Press, 1999, pp. 175–185.
[38] Shake T.H., Hazzard B., Marquis D., "Assessing network infrastructure vulnerabilities to physical layer attacks", in: *Proceedings of the 22nd National Information Security Systems Conference*, 1999.
[39] Skoudis E., "Cyberspace terrorism", *Server World Magazine* (February, 2002); URL: http://www.serverworldmagazine.com/monthly/2002/02/superworms.shtml.
[40] Skoudis E., *Malware—Fighting Malicious Code*, Prentice-Hall, 2004.
[41] Spitzner L., "Honeypots—definitions and value of honeypots", URL: http://www.tracking-hackers.com/papers/honeypots.html, May, 2003.
[42] Spitzner L., "Honeytokens: the other honeypot", URL: http://www.securityfocus.com/infocus/1713, July, 2003.
[43] "The spyware guide", URL: http://www.spywareguide.com/txt_intro.php, 2004.
[44] Stevens R.W., Wright G., *TCP/IP Illustrated*, vols. 1–3, Addison–Wesley, Boston, 1994.
[45] Tanase M., "IP spoofing: an introduction", *Security Focus* (March, 2003), URL: http://www.securityfocus.com/infocus/1674.
[46] Brian Tung B., "The Moron's guide to Kerberos, version 1.2.2", University of Southern California, Information Sciences Institute, December, 1996, URL: http://www.isi.edu/gost/brian/security/kerberos.html.
[47] Viega J., McGraw G., *Building Secure Software*, *Professional Computing Series*, Addison–Wesley, 2002.