Introduction to TCP/IP Network Attacks

Guang Yang
sunlight@cs.iastate.edu

Department of Computer Science
Iowa State University
Ames, IA 50011

ABSTRACT

Computation model has experienced a significant change since the emergence of the computer networks, which brought us much benefit that is difficult or even impossible to achieve by the traditional centralized system, such as resource sharing, high reliability, better price/performance ratio, and so on. Meanwhile, there also come many potential threats to the network community like unauthorized access to private information, malicious break-in to other organizations' systems, or intent to render a system to make it unreliable or unusable. Network attacks generally adopt computer networks as transportation media to convey the intrusion or even attack the communication system itself. We will put our focus mainly on the network attacks happened around the TCP/IP (Transmission Control Protocol/Internet Protocol) protocol suite, which is the most widely used communication protocol and the de facto standard among the Internet society.

KEYWORD: Network Attack, TCP/IP.

TABLE OF CONTENTS

1. TCP/IP BASICS

1.1 Protocol Hierarchy

TCP/IP is the most widely used protocol suite today, which was developed under the sponsorship from DARPA (Defense Advanced Research Projects Agency). It is the de facto standard employed to interconnect computing facilities in modern network environments.

TCP/IP protocol suite is designed through a highly structured and layered approach, with each layer responsible for a different facet of communications [1]. This hierarchical architecture makes each protocol layer possible to develop independently without affecting the adjacent layer. Data encapsulation is achieved by various headers among different transportation layers, like IP header, TCP header or application headers. These headers are critical in keeping the state information for each network connection or facilitating the multiplexing and de-multiplexing of communication messages.

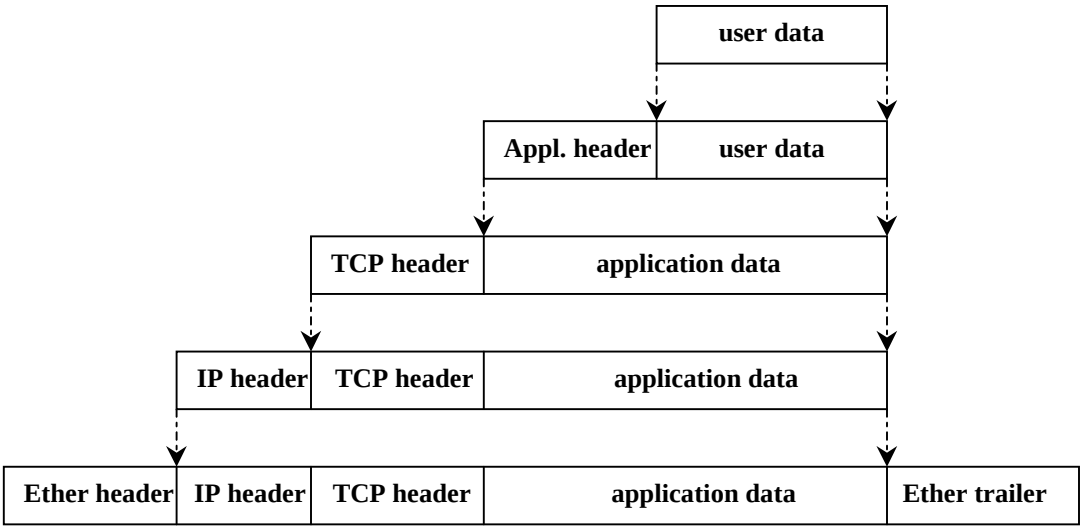

Figure 1: TCP/IP Protocol Stack

1.2 IP

IP (Internet Protocol) is the workhorse protocol of the TCP/IP protocol suite, which provides an unreliable, connectionless datagram delivery service. All TCP, UDP(User Datagram Protocol), ICMP(Internet Control Message Protocol), and IGMP(Internet Group Management Protocol) data are transmitted as IP datagrams [1].

Within IP header, there is important information like source IP address, destination IP address, which plays an important role in routing the packet around the network. A detailed description of IP header can be found in [1].

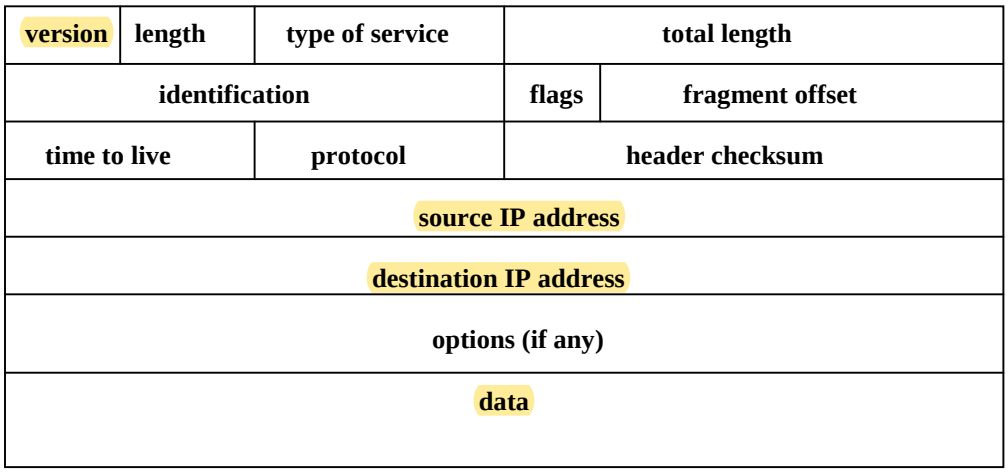

Figure 2: IP Header

Deliver such a packet to the correct destination host is non-trivial, especially in a large-scale network system like Internet. Each intermediate routing device makes best effort to deliver the IP datagram, but there is no guarantee it will reach the destination finally. So, datagram can be lost, duplicated, or delivered out of order [4]. It is the task of higher layer protocol to perform error control and flow control.

1.3 UDP

UDP is a transportation layer protocol, but it does not offer much more functionality other than IP. The checksum field in UDP header provides only a limited ability for error checking.

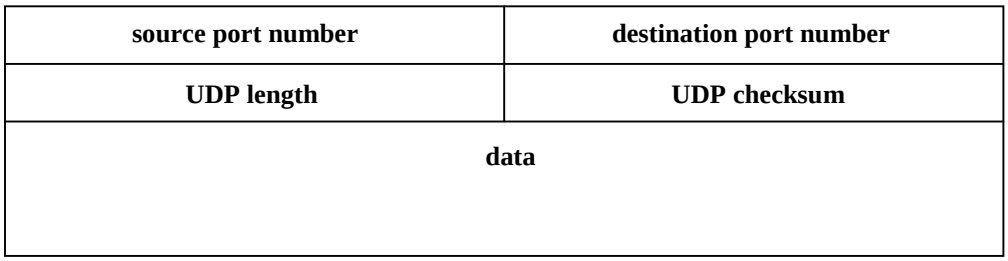

Figure 3: UDP Header

However, due to its simplicity and less overhead comparing with connection-oriented protocols, UDP is suitable for the design of simple request/reply application, such as DNS(Domain Name System), SNMP(Simple Network Management Protocol) and database transaction.

1.4 TCP

TCP is built on top of IP layer, which is unreliable and connectionless. However, TCP provides the higher layer application a reliable connection-oriented service. As a tradeoff, each TCP connection requires an establishment procedure and a termination step between communication peers. Furthermore, TCP also provides sequencing and flow control.

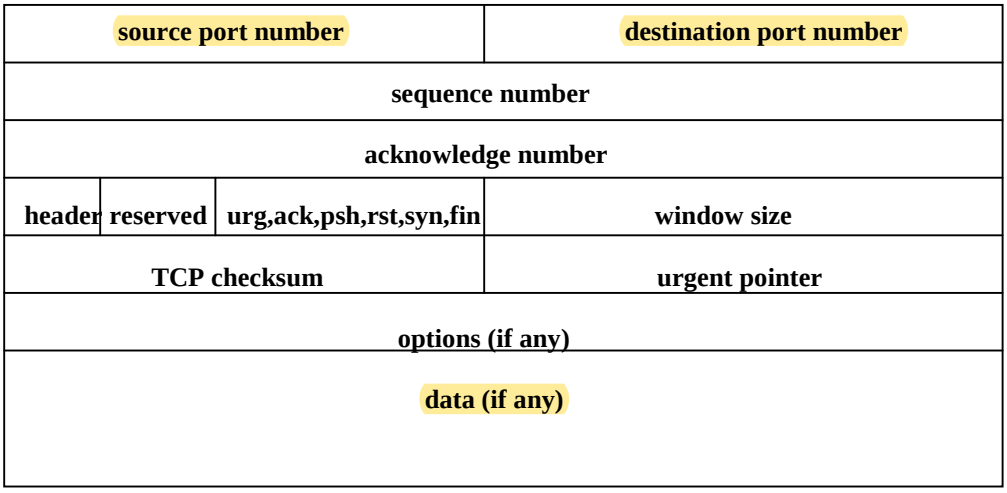

Figure 4: TCP Header

Without options, TCP header occupies 20 bytes as shown in the figure 4. The source and destination port number is used to identify the sending and receiving processes. The sequence number is essential in keeping the sending and receiving datagram in proper order. There are six flag bits with the TCP header, namely URG, ACK, PSH, RST, SYN and FIN, each of them has a special use in the connection establishment, connection termination or other control purposes. Window size is advertised between communication peers to maintain the flow control.

TCP establishes a connection in three steps, namely three-way handshake. Following figure is a typical three-way handshake procedure happened between a source host S and a destination host D.

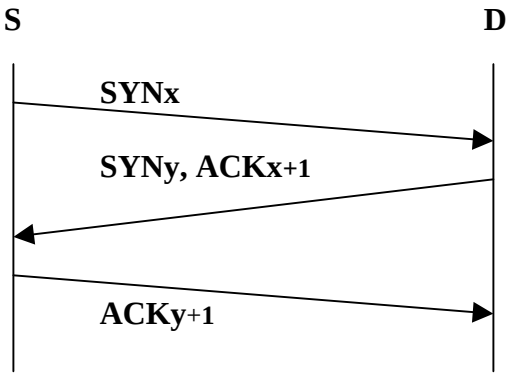

Figure 5: Three-way Handshake

First, source host sends a SYN packet to destination host, telling it the wish to establish a connection and setting its own ISN (Initial Sequence Number) in sequence number field. Upon receiving the request packet, the destination host sends back a SYN_ACK packet with its own ISN and the incremented ISN from source host. Finally, the source host will transmit an ACK packet and data transfer can take place. There is one extra point need to mention. Suppose that host S did not send any SYN packet but received a SYN_ACK packet from host D, it would just send back a RST packet to reset the connection. Later on you will see this point is crucial to some kinds of network attacks.

2. Essential Hacking Techniques

TCP/IP protocol suite is not perfect. There exists a number of serious security flaws inherent in the protocol design or most of TCP/IP implementation [2]. The network hackers just utilize these security holes to perform various network attacks. Among the hacking techniques, three of them are commonly used and reflect some typical problems in TCP/IP protocol suite.

2.1 IP Address Spoofing

As we have seen from previous section, the IP address, either source address or destination address, contained in the IP header is the only information needed by the

intermediate routing devices to make the decision on routing this IP packet. Anyone who has access to the IP layer could easy to spoof the packet's IP source address, then masquerade it as from another host in the network. Under UNIX system, only privileged process can access the raw socket, in other words, IP layer API (Application Programming Interface). However, with the growing popularity of personal computers, this limitation becomes trivial for a family network user with PPP connection accessing the Internet.

2.2 TCP Sequence Number Prediction

From the establishment procedure of TCP connection, we knew, for a conversation to take place, the source host must first get the returned ISN from the destination host. Normally, an ISN is just a more or less random number [2]. Imagine that, a hacker on host X has a way to predict this sequence number, then it could send the following packets to impersonate source host S.

X -> S: SYN(ISNx), SRC = D
S -> D: SYN(ISNs), ACK(ISNx)
X -> S: ACK(ISNs), SRC = T

Figure 6: TCP Sequence Number Prediction

Even though the packet S->D does not got to X, X is still able to know its contents and can send nasty data to D.

In the Berkeley systems, the initial sequence number is incremented by a constant amount (128 in 4.2BSD and 125,000 in 4.3BSD) once per second and by half that amount each time a connection is initiated. Thus, what a hacker needs to do is just initiate a normal connection and keep the ISN received from destination host, then calculate the ISN for the next connection attempt, based on the round-time delay, number of connections after the first connection. Usually, it has high possibility to succeed.

2.3 Port Scanning

Port scanning is not a technique used directly to perform an attack. Instead, its goal is to discover an exploitable communication channel and then launch the real attack. The reason for doing port scanning is that some vulnerable services may not use a fixed port number. Like in SUN NFS system, some application servers listen to an arbitrary port and register that port to a specific server, which is in charge of directory service. For

each client, it will first ask the directory server for the port number of that particular application server, then make connection. Generally, the directory server is always well protected. So, a hacker needs another way to locate his victim.

There are several ways to detect a potential communication channel. For a listening TCP server, the most basic approach is to try to make a real connection. The UNIX system call "connect()" can be used to open a connection with every interesting port. If there is a listening server, "connect()" will succeed, otherwise the port is unused. Another variant method is SYN scanning, which sends a SYN packet to the victim as if it is going to create a real connection. As mentioned in TCP three-way handshake, a SYN_ACK packet will indicate an active server and a RST message tells a port with no listener. These two methods above have obvious disadvantages. The first one is easy to be audited and the last one will not work with some firewalls or packet filters specially designed to block SYN packets to non-permitted ports. Then comes TCP FIN scanning. Instead of sending SYN probes in SYN scanning, this method adopts FIN packet, then it waits for RST packet from a closed port. In case of an active listener, it will just discard this FIN packet silently without sending anything back.

Unlike TCP, UDP is a type of connectionless protocol. However, its simplicity makes port scanning actually more difficult. Without three-way handshake in TCP, a UDP server does not need to acknowledge any probe packet. On the other hand, for a closed port, no UDP error message is required to return. Fortunately, most hosts send ICMP "port unreachable" message for a packet intends to an unused UDP port. Then it will give hackers some clues. But, neither UDP packets nor the ICMP messages are guaranteed to arrive, due to the unreliable nature of UDP. So, a port scanner also needs to deploy some retransmission policy in case the lost packets produce a bunch of false positive results.

3. Network Attacks

Several types of network attacks have been found up till now, each of them utilizes one or more security vulnerabilities in the TCP/IP protocol specification or some well-known weakness in the implementations of TCP/IP. There includes, IP address spoofing, TCP sequence prediction as we have mentioned early, others like, SYN flooding, DNS misuse, Ping of Death, or some newly come-out Java-related attacks. However, according to their basic techniques and impacts to the victim system, we could divide them into two basic categories: denial of service and spoofing.

3.1 Denial of Service

The lifeblood of today's world is information [3]. The denial-of-service attacks attempt to prevent or delay access to the information or the information processing systems. The basic idea behind this type of attack is to tie up a service provider with bogus requests with intent to render it to unreliable or unusable. A slight variant is to launch large number of processes to run up CPU cycles, memory or other resources.

3.1.1 CHARGEN and ECHO

CHARGEN is a simple service provided by most TCP/IP implementation under UNIX. It runs on both UDP and TCP port 19. For every incoming UDP packet, the server just sends back a packet with 0 to 512 randomly selected characters. Another well-known service is ECHO, which is running on UDP and TCP port 7. For each packet coming in, the server just responses with whatever it has received.

These two services are generally used for diagnosing purpose. However, they could be employed by a malicious denial-of-service type attack. Assuming a "chain" has been established between a CHARGEN service and an ECHO service, what will happen? Each of them will produce output continuously, leading to a very large number of packets among the network and thus a denial of service on the machines where the services are offered.

Launching such an attack is surprisingly easy, a simple UDP packet could set a whole network into trouble. Suppose there are two hosts A and B and a hacker on machine X. With the help of IP address spoofing, a hacker could send out a UDP packet to A with B's IP address as the source IP address and 7 as the source port, while setting the destination IP address as A's IP address and 19 as the destination port. When this packet arrived A, A will falsely think B wants to access the CHARGEN service, then sends back a packet to B's ECHO port. At this point, a "chain" has been established successfully. Subsequently, large amount of traffic will be generated within the network where hosts A and B reside. As a result, network users will feel an abrupt drop of the speed of their network applications.

Generally speaking, CHARGEN and ECHO type of attack is a kind of blind attack. There is no particular object from a hacker's point of view. The goal is to slow down a whole network. The idea behind this attack can be easily extended to other UDP services. If a poorly designed UDP server could not correctly deal with the abnormal incoming request, it is highly possible that a hacker could trigger infinite message exchanging between two innocent hosts.

3.1.2 SYN Flooding

Unlike CHARGEN and ECHO attack, SYN flooding is a specially designed attack, which employs a flood of SYN packet to consume all available new network connections on a targeted host, resulting in delays responding to legitimate network connection requests and eventual halting the service provider [3]. Theoretically, this attack applies to all TCP connections, such as WWW, Telnet, e-mail, and so on.

Under most UNIX systems, several memory structures need to be allocated for each TCP connection establishment. Take BSD system as an example, a socket structure is used to hold the communication information, like protocol used, address information, connection queues, buffers and flags [4]. Moreover, there are two other memory structures have special meanings to a TCP connection, namely IP control block (inpcb) and TCP control block (tcpcb), which keep the TCP state information, port numbers, sequence numbers and several connection-related timers, etc. Typically, these structures will take at least 280 bytes in total.

A normal scenario of TCP connection starts with the system in LISTEN State while receiving a SYN packet, which is being examined for checksum immediately. If checksum is incorrect, the packet will be discarded silently, with the expectation that the remote site will retransmit it. Otherwise, the TCP control block associated with this connection is being searched for. If no such item is found, it means no server process is waiting for this packet, then the packet will be removed and an RST packet is used to inform remote client process. On the other hand, if a server process is located and the corresponding backlog queue happens to be not full, several memory structures will be allocated for this connection and a SYN_ACK packet will be sent to continue the three-way handshake. At same time, system enters SYN_RECVD State and starts a connection establishment timer. Most TCP/IP implementation uses exponential back-off algorithm and set this timer as 75 seconds. If the final ACK arrives before the timer expires, the request will leave kernel space and goes to backlog queue or application process space. Otherwise, the timer expires and three-way handshake fails. Under both cases, the corresponding memory structures will be de-allocated from kernel space.

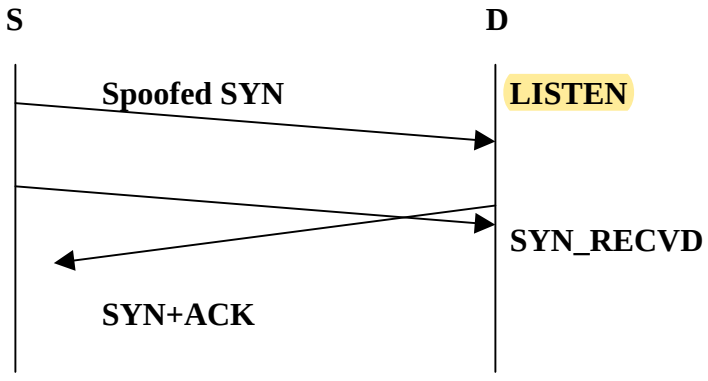

Figure 7: SYN Flooding Attack

From the description above, we knew, a TCP connection process requires significant amount of work and resource at server side. So, generally there is a small limit on the total number of half-open connections for a system. A hacker just explores this limitation and initiates a SYN flooding attack by issuing a large number of connection requests with spoofed source IP address to the victim machine. The server can not tell the difference between a legal request and a malicious request. After receiving the SYN packet, it will respond with SYN_ACK packet as usual. Unfortunately, this time the final ACK packet will not come back anyway. The reason is the hacker issued a request SYN packet with spoofed IP address and that address appeared "unreachable" to the victim machine. There are several reasons for an IP address to be "unreachable". For instance, the machine with that IP address is turned down or even worse there is no machine with that IP address at all. Even if there is some error message like ICMP "host unreachable" or "network unreachable", which may be generated by a router, coming back, the current implementation of TCP/IP always omits this error message and just keeps transmitting the SYN_ACK packet until certain timer expires. If during this timeout, no further memory structures can be allocated, the following connection requests will all be discarded.

The key issue in this type of attack is to choose an "unreachable" source IP address for each attacking packet. There are also several patterns followed by the hackers.

- Single address: all the attacking packet using same IP address
- Short list: there is a small pool of address for a outgoing packet to choose
- No list: the source address is generated randomly

Different addressing method will bring different difficulty to attack detection and protection.

The basis of this attack is that TCP/IP protocol suite does not provide strong authentication on its control packet [4]. The endpoint of a connection has no way to authenticate its communication peer. As a result, based on current technology, it is extremely difficult to trace the original source of the spoofed IP packet. So, the hacker can feel free to perform this kind of attacking without worrying about any legitimate responsibility. Other drawback of TCP/IP specification discovered by this attack is asymmetrical workload for client and server side of each connection: server is required to allocate much more resource than those on the client side are.

3.1.3 Other Denial of Service Attacks

Other forms of denial-of-service type attacks also exist, like Ping of Death, which utilizes a ping packet with abnormal packet size exceeding the TCP/IP specification. As a result, receivers of those ping packets will mystically system crash or stop processing. Most possible reason for it is due to the poorly implemented TCP/IP stack.

There are some types of attacks, which utilize the broadcast property of transmission media, are limited to a LAN (Local Area Network) environment. However, we could not overlook those attacks. Among a modern network system, heterogeneous hardware and software architecture becomes more and more common. It is quite possible that some hosts may be less secure than others, either because of less of physical protection or little concern in operating system design. A hacker can perform a multi-stage attack by first breaking into a less safe host and then compromise the whole network system. One of the threats to a LAN is called SYN_RST generator, which can block most of TCP connections. Suppose a host A want to make a TCP connection with host B, it will first send a SYN packet to B. At the same time, host X also heard this message, due to the broadcast communication media. Before B responds with a SYN_ACK packet, X quickly sends out a RST packet to A, shutting down the intended connection. In this case, the more fast a machine is, the more possible it could destroy other connection intent. Another example aims to attack the flow control mechanism in TCP conversation. In order to prevent a fast sender to overrun the buffer of a slow receiver, each TCP packet has a window size advertisement for their communication peer. During the communication process, a third party host could impersonate the destination host, sending a packet with zero window size. Then, each communication party may be halted due to the lack of buffer advertised by its communication peer.

With more and more use of Java in the web computing, some attacks by the malicious applets should raise us more attention. Most of the applet attacks fall into the type of denial of service, in which a Java applet launches huge amount of resource-sensitive computation to consume CPU and memory resources of the client machine or props up numbers of windows to block a browser user to access other applications.

3.2 Spoofing

Spoofing is the mostly used hacking technique in network attacks. Due to the distributed nature of computer networks, the main method being used to exchange data among different hosts is message passing. Therefore, strong authentication is not easy to archive comparing to that in a centralized system, especially among arbitrary communication peers. Network hackers just explored into this weakness and devised many various network attacks, either spoofing themselves as legitimate clients or original servers.

3.2.1 Client-Side Spoofing

In client-side spoofing, a hacker impersonates himself as an authorized client and in turn gains service from an unintelligent server. A hand-on example is “r-utilities” under most of UNIX systems.

"R-utilities", like rlogin, rsh and rcp, are a set of commands for remote login and file transfer among different UNIX systems. While easy to use, these commands are also fairly easy to be compromised [5]. And even worse, some of them could be used as a tool to comprise a whole system. The security problem underlying them is the authentication scheme adopted by this set of commands.

Take rlogin as an example, which is a simple client-server type protocol that uses TCP as its transportation layer protocol. For a pair of hosts A and B, both of them "trust" with each other. After setting several corresponding items in the "/etc/hosts.equiv" or ".rhosts" files, a trusted user with accounts of same names on both hosts could login from one host to another without being prompting any password information. In fact, the client is being authenticated via its host name or IP address.

In 1995, CERT (Computer Emergency Response Team) issued a security advisory addressed a kind of attack called "IP Spoofing". In which, hackers created packets with spoofed source IP address, then exploited applications that use authentication based on IP address, like "r-utilities" [6]. IP spoofing consists of several steps and uses both methods we mentioned before, IP address spoofing and TCP sequence number prediction. Following are two scenarios happened during two rlogin sessions, one is a normal one, while the other is a spoofed attack.

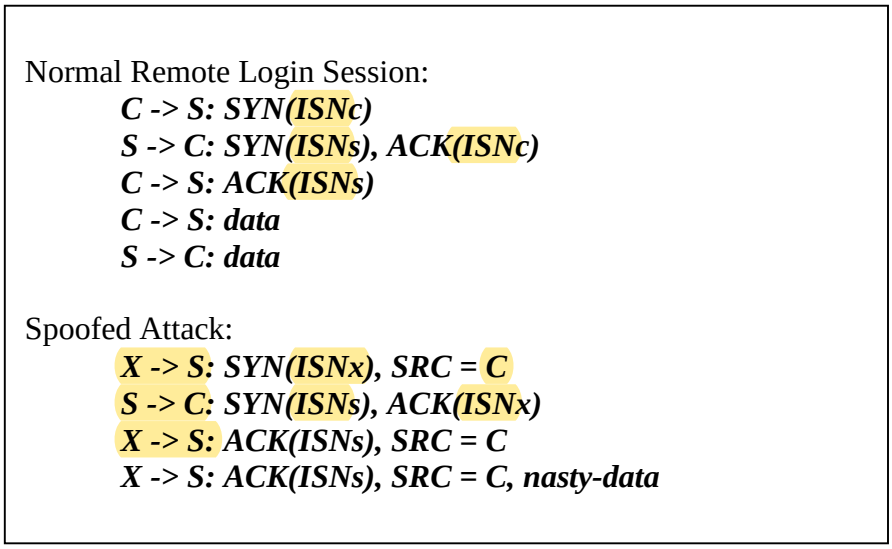

Figure 8: IP Spoofing

As seen in a normal remote login session, a three-way handshake is being used between client and server host to establish a TCP session. Till the end of three-way handshake, the client receives a sequence number from server and then increments it and sends it

back to the server as an acknowledgement. Subsequently, the server verifies this returned sequence number and granted the client permission for further data exchange. Without the server's sequence number, a client could not proceed the rlogin session with the remote server.

However, with the help of TCP sequence number prediction technique, a hacker could talk to a rlogin daemon even without having to receive the server's sequence number returned in three-way handshake. Usually, a "IP Spoofing" attack adopts following steps:

- First of all, a victim host is chosen and a pattern of trust is discovered, like which host is the trusted host of victim host. In other words, a user of trusted host can use "r-utilities" to access the information on victim host.
- Then, the trusted host is being disabled, possibly by SYN flooding to crash that machine or block all the normal network traffic to it.
- Next, a normal TCP connection request packet is sent to the victim host and gets back a valid sequence number from the victim host. Based on round-trip delay and TCP sequence number generation algorithm, a hacker could guess what is the next sequence number to be used by the victim host with high possibility of success.
- At this point, an attack is ready to go. The hacker then sends out a SYN packet with the trusted host as its source IP address. Even though the returned SYN_ACK packet form the victim host could not reach hacker, he can still continue the connection by sending out the final ACK message with correct sequence number generated by previous step.
- Up till then, the victim host S falsely thinks there is valid connection request form the trusted host C. Then, the hacker could send nasty data from host X to compromise or intrude the victim host.

One thing needs to be clarified in this type of attack is, when the hacker form host X masquerades himself as a trusted client and sends out a SYN packet, the returned SYN_ACK packet will go to the real host C. As mentioned in previous chapter, the host C will immediately response with a RST packet and the attack will fail because the intended connection will be shutdown as soon as the victim host received this packet. Therefore, a SYN flooding attack is always performed as the first step in "IP Spoofing". As a result, the returned SYN_ACK packet would not reach the destination host C but get lost on the way. The reason is the host C is busying in dealing with large amount of bogus requests and running out of system resources.

IP spoofing is a typical example of client-side spoofing attack. And all the applications with loosely authentication mechanism also face the threats from this type of intrusion.

3.2.2 Server-Side Spoofing

Server-side spoofing employs a similar idea as the spoofing from the client-side. However, the goals and the methods being used for these two types of attacks are of a little difference. For the client-side spoofing, as we mentioned in the example of rlogin, a hacker impersonates himself as an authorized user and then gains data from some information provider. On the other hand, a server-side spoofing is used in a reverse way. In order to obtain private information form individual clients, a hacker masquerades himself as a real service provider and steals the trust from system users.

The ideas behind the server-side spoofing attacks can be properly expressed by a real world example. Suppose some guy made a machine, which looks extremely like an ATM (Automatic Teller Machine), with a screen, keyboard and a slot to insert ATM card. The only difference is that machine does not have the function of a normal ATM. Instead, it could only record the number of a ATM card and the holder's PIN (Personal Identification Number), then reports some error message to mislead the user this machine has temporary mechanical problem. If such a machine can be placed at the entrance of a shopping center, the result will be disastrous. A user may mystically lose large amount of money just because he/she once used an out-of-ordered ATM several days before.

Same idea has been adopted by a kind of attack called web spoofing, which utilizes the neglect of a browser user when he/she is addicting in web surfing. Some well-pretended HTTP (Hypertext Transfer Protocol) links are put on some popular web sites. When a victim user happens to click that link, all the following connection will be hijacked by a hacking server, which hides itself between the user browser and the real web server. Not much advanced techniques are being used in this attack. Some small Java script applet, together with a little HTTP and CGI (Common Gateway Interface) knowledge, will set the web user into trouble. Especially with the growing popularity of electronic commence, this type of attack becomes even more serious. A malicious webmaster can easily grab the personal information from a web shopper, like his/her Social Security Number, credit card number, when the user is purchasing stuff from online store.

There are many variants of server-side spoofing techniques, some of them generate the vulnerabilities of low-level protocols, like ARP/RARP (Address Resolution Protocol/Reverse Address Resolution Protocol). But the natures of those attacks are all quite similar. A hacker generally uses misleading appearance or bogus message to let clients falsely consider him as the real service provider, then leads to privacy disclosure.

4. Case Study

Some systems, like routing infrastructures, DNS (Domain Name System) and so on, play critical roles in the execution of a network system, especially the Internet. Therefore, the security issues of those systems should be discussed into more details.

4.1 Routing Infrastructure Attacks

As we described at the very beginning of this paper, all the TCP/IP services are built above a connectionless packet delivery system [7]. With a layered protocol stack in mind, every message is transferred in the form of IP packet, which is the basic unit of data traveled among distributed network devices. For a large-scale and heterogeneous network environment, like the Internet, making a packet to the right destination is the task of routing infrastructures.

Internet adopts a hierarchical routing architecture, which relives each single router from storing too much path information. A router makes routing decision of an IP packet based on a data structure called routing table, which holds the status of each path linked to that router. Periodically, a router generates LSU (Link State Updates) that describe the latest status of the links to the router and disseminates those updates to other neighbor routers. Then, based on LSU received, routers update their own routing tables and cooperate in forwarding the IP packets from source to destination [8].

Potential threats to the routing infrastructures come mainly from spoofing type attacks and some of them can lead to results of denial of service. A faulty router can modify the packets passing through it or discard the packets at all, then bring some networks or hosts unreachable. Furthermore, a malicious configured or compromised router can send bogus routing control packets to other routers, which may in turn cause all the packets switch to itself and then eavesdrop the content within the packets.

4.2 DNS Misuse

DNS is not inherent to TCP/IP protocol when it is first proposed. However, with millions of networks and hosts interconnected by the Internet, IP address becomes quite inconvenient for user level communications. An alternative approach is to map low-level IP address into meaningful hostname, which is the main motivation of DNS.

DNS is a distributed database system, which handles mapping high-level host names into low-level IP addresses, or vice versa. Much like routing infrastructures, DNS is composed by a large number of name servers in a distributed hierarchical architecture,

while each individual name server handles requests from a limited domain. If a name server does not know how to resolve a particular query, it may forward the query to another name server, which either has much more information or is more specific to that particular domain which the query asked.

Most DNS implementation adopts UDP as transportation layer protocol. Then besides the vulnerabilities of itself, it also inherits security weakness from UDP, like lack of state information, difficulty in user authentication, and so on. With the similar architecture as the routing infrastructures, DNS face the same threats from spoofing type attacks. A misused name server could be used easily by a hacker to masquerade himself as from any trusted host, for a resolver query can be intercepted and a hacker-controlled name server can response the query with whatever IP address a hack intends to be. A recently found bug in Java class verifier has a tight relation with this kind of attack, in which a malicious applet could connect with any host other then the host from which it was downloaded.

Caching is widely used by DNS to improve the system efficiency. In DNS specification, there is also a little concern for the data integrity and consistency of cache. So, an attack by sending spoofed information to a name server in a straightforward way will not work. Instead, a hacker use another approach called “Ask Me” to poison a name server’s cache by malicious data item [10].

Imagine there is a hacker on host X, who has full control of name server B and wishes to give the following wrong mapping information to name server A:

- IP of host X -> Name of host A
- Name of host A -> IP of host X

As NS B can not directly send out the false information to NS A, instead it asks NS A to resolve a mapping that can only be handled by NS B itself. As a result, NS A will forward this request back to NS B. Then NS B appends the incorrect mapping information between host name A and its IP address to the response to NS A. With this little trick, the cache of name server A will be poisoned by a malicious record. After this point, the hacker on host X could go ahead and launch some more serious attacks, such as attacks to name-based applications like “r-utilities”.

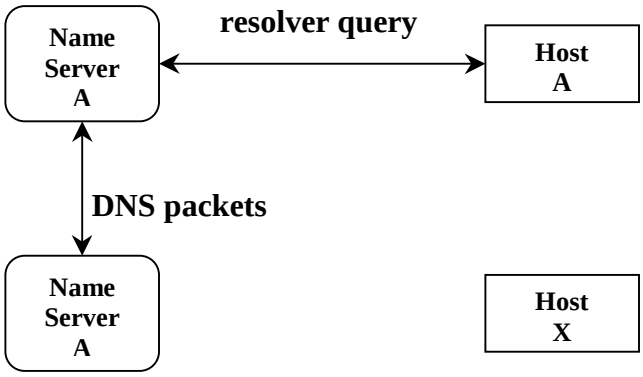

Figure 9: DNS Misuse

Besides some similar results mentioned in the routing infrastructure attacks, like misleading the packet flow, a DNS attack can greatly facilitate the attacks aimed at name-based applications. In sum, a combined attack on the DNS system and the routing mechanism can be catastrophic [2].

4.3 NFS – No File Security

NFS (Network File System) is created by SUN Microsystems and is the predominant distributed file system [9]. NFS allows files to be shared among multiple hosts and works in a way, which is transparent to the user applications.

A NFS server host exports one or more file systems to be used only by client hosts, which are permitted to use them. However, the NFS server grants the access based merely on the IP address of client machine, which is not difficult to spoof. In fact, NFS stores the user configuration on the server size, like whether the user is authorized to use this system, what level of access right a particular user has, such as read only or read-write. The need for user authentication is left for client host. So, from a compromised host, a malicious user could claim him as whoever he wants to be. Even worse, the whole NFS protocol is specified as a set of RPCs (Remote Procedure Call), which is built upon UDP. And UDP is trivial to spoof as we always mentioned in previous parts of this paper.

At application level, the NFS adopts a stateless approach, which means the server has no idea about what files are being opened by remote clients or what parts of the files are in process. The benefit of this approach is the system is easy to recover. The downside of it is, a client can keep the file handles, which are previously assigned by the server, and continue to use them, even though the server has no longer trusted that client host.

4.4 X-Windows

X Windows System, or X is a client-server application used mainly for display management among single or multiple UNIX systems. A server resides at a user machine and handles requests from clients, which may be local clients or users from remote hosts. Then server will dispatch these inputs to the corresponding client applications. X provides three good features: location independence, hardware independence and operating system independence [9].

The weakness of X exists in the access control method being used by X servers. When client machines are specified by "xhost", an X server allows access only from the hosts, which it grants permission. However, there is no restriction on which process of that machine can access the X server. So, a hacker can spoof as an X client and receive inputs from other user process on the same host or even from other hosts. Furthermore, the hacker could even modify the message, which does not belong to him or block it from reaching real user process. On the other hand, an X server can also be spoofed. Then it caused the display of information, which come from other users' applications.

If a host has been set by "xhost" to be accessible to your X server, then a user at that host could do:

- Destroy any (or all) of your windows
- Open new windows on your terminal
- View the contents of your screen remotely
- Log all your keystrokes, including your passwords
- Generate whatever X event, which he wants to

"Xauth" is a slight improvement of security mechanism of X system, which requires the remote X client to know a secret quantity for your display, which is called MIT-MAGIC-COOKIE-1. However, if a hacker happens to know that quantity, things does not make any change comparing with the case of "xhost".

4.5 Distributed Coordinated Attacks

DCA (Distributed Coordinated Attack) is not a new type of network attack, but it brings new challenges to the corresponding detection and protection system. Usually, a hacker's behavior can be traced by the system level auditing tools or through watching out one or a set of particular hosts. However, in a DCA, a large scope of hosts are being used, some of them even does not know it has been involved into a network attack.

A DCA password guessing attack is such an example, in which many components at different sites performing different tasks in a coordinated way in order to achieve a same goal. First of all, a web page, which includes a sophisticated Java applet, is being downloaded by a user's browser. Next step, with the help from some DNS attacks, the

Java applet connects to a victim host and attempts a sequence of password guessing in a brute force way. Once an entry succeeds, the Java applet will soon send back a copy of password to the hacker's host. During this attack, from the point view of victim host, one can only trace a series of login attempt from the browser's host. But the real hacker sitting in a dark corner of the Internet could not be found easily.

5. Possible Detection and Protection Methods

The distributed nature of computer networks makes it hard to detect a potential attack or protect a computer system from hackers' compromise. However, some techniques have been developed in this field, such as firewall, cryptography, and so on.

5.1 System Configuration Improvements

To defend against an incoming attack, the first thing to do is to make the system itself strong. There are some issues here, such as correcting the errors in system configuration files, make the right adjustment of system parameters like TCP time out timer, length of pending request queue.

5.2 Router Configuration Improvements

Router is a key device for internetworking. The correct configuration on the routing policy could efficiently reduce the possibilities of some types of network attacks. An obvious improvement should be like this:

- Configure external interfaces to block packets that have source IP address from the internal network
- Configure internal interfaces to block packets to the outside that have source IP address from outside of the internal network

5.3 Firewall

The first generation firewall is just a packet filter, like screening router, the security level depends largely on the experience of the network administrator. Second generation firewall is built on top of application level protocol. For each type of service, there must exist a proxy, which in charge of connection relay and can also enforce some security or authentication policies on both sides of the communication. Furthermore, the current firewall product also provides some extra functionality, like NAT (Network Address

Translation), SYN flooding protector, and so on. They are useful in protecting some particular types of attacks.

5.4 Infrastructure Improvements

There are some security holes inside the specification of the protocol itself, there is no external way can help resolve them, like the IP address spoofing. The only approach is to enhance the protocol itself. There are some works already been done for the protocol enhancement, like some protocols borrowing some ideas form cryptography. In the new IPv6, some special security features have also been proposed.

6. Conclusion

In sum, network security is always a big issue. Network attacks just reflect various problems in the existing network architecture and protocol organization. We just categorize them and list some possible ways to deal with them.

Tuesday, November 18, 1997

References

[1] W. R. Stevens, *TCP/IP Illustrated Vol. 1 – The Protocols*, Addison-Wesley, 1994.

[2] S. M. Bellovin, "Security Problems in the TCP/IP Protocol Suite", *Computer Communications Review*, Vol. 19, No. 2, pp. 32-48, April 1989.

[3] C. Cobb and S. Cobb, "Denial of Service", *Secure Computing*, pp.58-60, July 1997.

[4] C. L. Schuba, I.V. Krsul, Makus G. Kuhn, E.H. Spafford, A. Sundaram, D. Zamboni, "Analysis of a Denial of Service Attack on TCP", Purdue University, 1996.

[5] S. Dash, "Integration of DNSSEC (key-server) with Ssh Application", Iowa State University, 1997.

[6] CERT, "TCP SYN Flooding and IP Spoofing Attacks", Carnegie Mellon University, Sept. 1996.

[7] D. E. Comer, *Internetworking with TCP/IP: Vol. I – Principles, Protocols and Architecture*, Third Edition, Prentice Hall, 1995.

[8] S. Cheung, K. N. Levitt, C. Ko, "Intrusion Detection for Network Infrastructures", *The 1995 IEEE Symposium on Security and Privacy*, Oakland, CA, May 1995.

[9] R. C. Summers, *Secure Computing – Threats and Safeguards*, McGraw-Hill, 1997.