ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

Intrusion Detection System

Kranthi K Lammatha
Chief Technology Officer (ITU)
Email Id: Kranthi@itu.edu

ABSTRACT

Computer networks bring us not only the benefits, such as more computing power and better performance for a given price, but also some challenges and risks, especially in the field of system security. During the past two decades, significant effort has been put into network security research and several techniques have been developed for building secure networks. Packet filtering plays an important role in many security-related techniques, such as intrusion detection, access control and firewall. A packet-filtering system constitutes the first line of defense in a computer network environment. The key issues in the packet-filtering technique are efficiency and flexibility. The efficiency refers to the ability of a filter to quickly capture network packets of interest, while the flexibility means the filter can be customized easily for different packet patterns.

In this thesis, we present a real-time packet-filtering module, which can be integrated into a large-scale network intrusion detection system. The core of this packet-filtering module is a rule-based specification language ASL (Auditing Specification Language), which is used in describing the packet patterns and reactions for a network intrusion detection system. The important features of ASL that are not provided by other packet-filtering systems are protocol independence and type safety. ASL provides a number of new features that distinguish it from other languages used for intrusion detection and packet filtering, such as packet structure description and protocol constraint checking.

Wedevelop the algorithms and heuristics for constructing fast packet filter from ASL specifications. Our algorithms improve upon existing techniques in that the performance of the generated filters is insensitive to the number of rules. We discuss implementation of these algorithms and present experimental results.

INTRODUCTION

Computation models have experienced a significant change since the emergence of computer networks, which allow heterogeneous computers to communicate with each other. During the past two decades, most centralized systems have been replaced by a number of interconnected computers. This factor has led to more computing power, increased flexibility and better performance/price ratio.

However, at the same time, we also face many new challenges and risks with networked computing, such as lack of communication reliability, coordination, resource management, and so on. As more and more computer networks are brought into electronic commerce, transaction management, and even national defense, people begin to pay increasing attention to system security.

Network Security and Potential Threats

There are a number of security issues for a computer network environment [1]:

Availability: The system must be functional and correctly provide services.

Confidentiality: The data transmitted from one system to the other must be accessible only for the authorized parties.

Authentication: The identity associated with the data

Cite this article as: Kranthi K Lammatha, "Intrusion Detection System", International Journal & Magazine of Engineering, Technology, Management and Research, Volume 5 Issue 10, 2018, Page 70-82.

must be correct. The identity can apply to a user, host or software component.
Integrity: The data being processed or transmitted can be modified only by the authorized parties.
Non-repudiation: Neither the sender nor the receiver of data is able to deny the fact of data transmission.

A system that meets the above criteria can be considered as a secure computer network system. A hacker, who wants to attack a network, thus thinks of ways to compromise the above criteria [1]:
Interruption: Destroy a system or make it unavailable or unusable.
Interception: Obtain unauthorized access to data.
Modification: Compromise data integrity, e.g. modify messages sent from one system to another.

Intrusion Detection

As defined by Heady et al. [2], an intrusion is any set of actions that attempt to comprise the integrity, confidentiality or availability of theresource.
Intrusion leads to violations of the security policies of a computer system, such as unauthorized access to private information, malicious break-in into a computer system, or rendering a system unreliable or unusable.

A full-blown network security system should include the following subsystems:
Intrusion Detection Subsystem: Distinguishes a potential intrusion from a valid network operation.
Protection Subsystem: Protects the network and security system itself from being compromised by the network intrusions.
Reaction Subsystem: This part either traces down the origin of an intrusion or fights back the hackers.

The focus of this thesis is on the intrusion detection subsystem, which constitutes the first line of defense for a computer network system. There are a number of approaches in this field. Most of them fall into three primary categories: anomaly detection, misuse detection and hybrid schemes.

The anomaly detection approach is based on a model of normal activities in the system. This model can either be predefined or established through techniques such as machine learning. Once there is a significant deviation from this model, an anomaly will be reported. By contrast, a misuse detection approach defines specific user actions that constitute a misuse and uses rules for encoding and detecting known intrusions [3]. The hybrid detection approach uses a combination of anomaly and misuse detection techniques.

OVERVIEW OF TCP/IP BASED NETWORK INTRUSION

TCP/IP is the common language used in the world of computer networks. Nevertheless, there exist several security flaws in the protocol design or implementation of TCP/IP. As a result, network hackers, who intend to compromise the target network systems by exploiting these security holes, have invented various intrusion methods.

TCP/IP Basics

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

Protocol Hierarchy

TCP/IP is designed through a layered approach, with each layer responsible for a different facet of communication [4]. This hierarchical architecture makes each protocol layer possible to evolve independently without affecting the adjacent layers. In addition, data encapsulation is achieved through various headers among different transportation layers like IP header, TCP header or other application headers as shown in Figure 2.1. These headers are important in keeping the state information for each network connection and facilitating multiplexing and de-multiplexing of transmission messages.

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

IP

IP is the workhorse protocol of the TCP/IP protocol suite. It provides an unreliable, connectionless datagram delivery service. All the TCP, UDP (User Datagram Protocol),ICMP (Internet Control Message Protocol), and IGMP (Internet Group Management Protocol) data are transmitted as IP datagrams [4].

An IP header has the information like source IP address and destination IP address, which plays an important role in routing a packet around the networks. A detailed description of IP header can be found in [4]. Figure 2.2 shows the structure of a normal IP header.

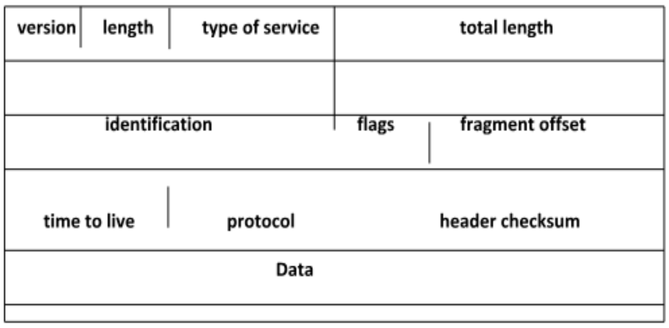

Figure 2.1 IP Header

Delivering a packet to the correct destination is non-trivial, especially in a large-scale network system. Each intermediate routing device makes best effort to deliver the IP packet, but there is no guarantee that it will reach the destination finally. So, a packet can be lost, duplicated, or delivered out of order [4]. It is the task of higher layer protocols to correct such errors.

UDP

UDP is a transport layer protocol, but it does not offer much functionality over and above that of IP. The port numbers in UDP header identify the sending process and the receiving process [4], while the checksum provides a limited ability for error detection (Figure 2.3).

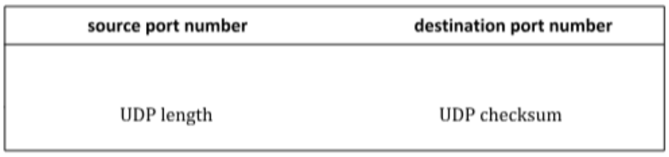

Figure 2.3 UDP Header

However, due to its simplicity and low overhead compared to connection-oriented protocols, UDP is suitable for the design of simple request/reply application protocol, such as DNS (Domain Name System), SNMP (Simple Network Management Protocol), and so on.

TCP

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

Without any option, a TCP header occupies 20 bytes as shown in Figure 2.4. The sequence number is essential in keeping the sending and receiving datagram in proper order.

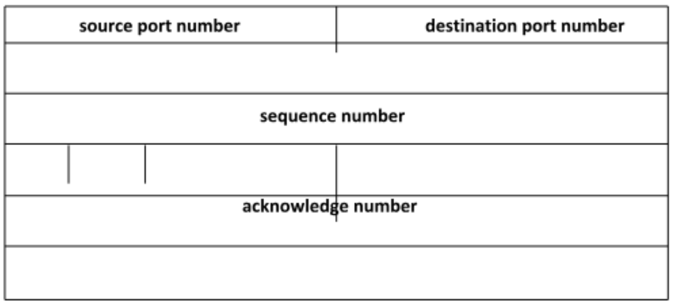

Figure 2.4 TCP Header

There are six flag bits within a TCP header, namely URG, ACK, PSH, RST, SYN and FIN, each of which has a special meaning in connection establishment, connection termination or other control phases. Window size, which specifies how many bytes of data can be accepted each time by the receiving side, is advertised between the two communication peers for the purpose of flow control.

TCP establishes a connection in three steps, commonly known as a three-way handshake. Figure 2.5 shows a typical three-way handshake procedure between a source host S and a destination host D.

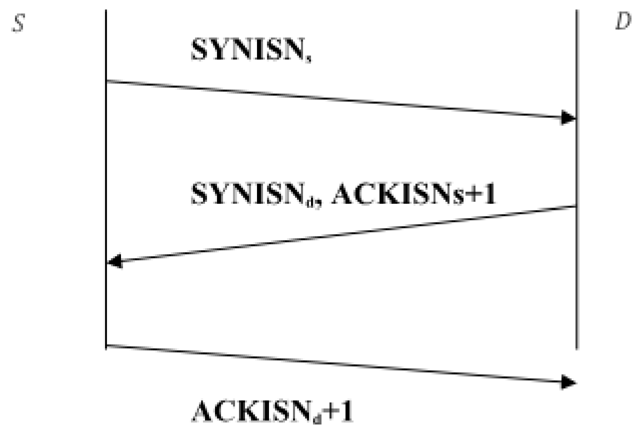

Figure 2.5 Three-Way Handshake

First, S sends a SYN packet to D in order to establish a connection. Meanwhile, S sets its own ISN (Initial Sequence Number) in sequence number field of the packet. Upon receiving the request packet, D sends back a SYN_ACK packet as the acknowledgement including its own ISN and the incremented ISN from S. As the acknowledgement packet reaches the source host S, S immediately transmits an ACK packet back to D. In the last ACK packet, S needs to include the incremented ISN of D as the confirmation of the connection. At this point, the connection has been setup. There is one extra point that needs to be mentioned: suppose that host S does not send any SYN packet but received a SYN_ACK packet from host D, it will then send back a RST packet to reset the connection.

Common Vulnerabilities

During the past two decades, many security problems of TCP/IP protocol suite have been discovered. Meanwhile, the network hackers created a large number of intrusion methods to exploit those vulnerabilities. Most of the examples in this section are taken from Bellovin’s excellent paper on TCP/IP security [5].

IP Source Address Spoofing

As we have seen from the previous section, the IP address (either source address or destination address) contained in an IP header is the only information needed by an intermediate routing device to make a decision on how to route the IP packet. So, anyone who has access to the IP layer can easily modify the source address in the IP header of a packet, spoofing itself as from another host or even from a non-existing host.

TCP Sequence Number Prediction

From the three-way handshake, we know that to establish a TCP connection between two communication peers, the source host must obtain the ISN of the destination host from its acknowledgement packet. Usually, an ISN is more or less a random number [5].

If a hacker can predict the ISN, he/she can impersonate host S by sending a request packet with the IP source address changed to S. Although the hacker will not get the SYN_ACK packet sent by D, he/she can still finish the establishment process by sending back an ACK packet to host D with predicted ISN. As shown in Figure 2.6, ISNg represents the guessed ISN of host D by the hacker.

In Berkeley systems, the initial sequence number is incremented by a constant number (128 in 4.2BSD and 125,000 in 4.3BSD) once per second and by half that number each time a connection is initiated [6]. Thus, what a hacker needs to do is just to initiate a normal connection and remember the ISN received from the destination host. After that, the hacker could calculate the ISN for the next connection attempt, based on the round-trip delay and the number of connections after the first connection. This approach has a high probability of succeeding.

X → D: SYN(ISN_x), SRC = S
D → S: SYN(ISN_d), ACK(ISNx)
X → D: ACK(ISN_g), SRC = S

Figure 2.6 TCP Sequence Number Prediction

Port Scanning

Strictly speaking, port scanning is not a technique used directly to perform an intrusion. Instead, its goal is to discover an exploitable communication channel and then launch the real attack. The reason for doing port

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

scanning is that some vulnerable services may not use a fixed port number. As in the SUN NFS system, some application servers run at an arbitrary port and register the port number to a specific server, which is called directory server. For the client programs of a particular application server, they need to first check with the directory server to obtain the port number for that application server. Usually, the directory server is well protected. So, a hacker needs another way to locate his victim.

There are several methods that can be used to detect a potential communication channel. For a listening TCP server, the most elementary approach is to make a real connection. The UNIX system-call connect can be used to open a connection with every port that the hacker intends to examine. If there is a listening server, the connect call will succeed. Otherwise the port is unused. Another method is through SYN scanning, in which a SYN packet is sent to the victim as if it is going to create a real connection. As mentioned in TCP three-way handshake, a returned SYN_ACK or RST packet indicates the presence or absence of an active server on the port. Another variant of this approach is TCP FIN scanning. Instead of sending SYN probes like in SYN scanning, this method sends FIN packet and waits for a RST packet from a closed port. In case of an active listener, it will discard the FIN packet silently without sending anything back.

Unlike TCP, UDP is a connectionless protocol, whose simplicity makes port scanning more difficult. Since UDP does not require a three-way handshake to establish a connection, a UDP server does not need to acknowledge any probe packets. Also, no error messages are returned for closed ports. However, most hosts send ICMP “port unreachable” message for a packet intended for an unused UDP port. This gives hackers some clue. Since neither UDP packets nor the ICMP messages are guaranteed to be delivered due to the unreliable nature of the protocol itself, a port scanner needs to have some retransmission policy to ensure that lost packets do not lead to erroneous results.

Network Intrusions

A number of network intrusions have been found till now, each of which utilizes one or more security vulnerabilities in TCP/IP protocol specifications or implementations. These intrusions include IP source address spoofing, TCP sequence number prediction as mentioned earlier, and other intrusions like SYN flooding, DNS misuse, Ping of Death, or some Java-related attacks. However, based on the intrusion patterns and impacts to the victim systems, we can divide the intrusions into two main categories: denial of service and spoofing.

Denial of Service

The lifeblood of today’s world is information [8]. The denial-of-service intrusions attempt to prevent or delay access to the information or the information processing systems. The basic idea behind this type of intrusion is to tie up a service provider with bogus requests in order to render it unreliable or unusable.

CHARGEN and ECHO

CHARGEN is a simple service provided by almost all TCP/IP implementation under UNIX. It runs on both UDP and TCP port 19. For every incoming UDP packet, the server sends back a packet with 0 to 512 randomly selected characters. Another well-known service is ECHO, which runs on UDP and TCP port 7. The server just responds to the client program with whatever it receives.

These two services are normally used for the diagnostic purpose. However, they can be employed by a malicious denial-of-service type intrusion. Assuming a “chain” has been established between a CHARGEN service and an ECHO service, what will happen next? Each of them will produce output continuously, leading to a huge number of packets among the network and thus a denial of service on the machines where the services are provided.

Launching such an intrusion is surprisingly easy. A simple UDP packet could set the whole network into

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

trouble. Suppose there are two hosts A and B and a hacker on machine X. With the help of IP source address spoofing, a hacker can send out a UDP packet to A with B's IP address as the source 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 is received by A, A will falsely think that B is requiring the CHARGEN service, and 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 in the performance of their network applications.

Generally speaking, CHARGEN and ECHO type of intrusion is a kind of blind attack. There is no particular objective from a hacker's point of view. The goal is to slow down the speed of the whole network.

SYN Flooding

Unlike the simple CHARGEN and ECHO intrusion, SYN flooding is a specially designed attack that employs a flood of SYN packets to consume the limited resource on the targeted host. It results in delays to legitimate network connection requests and eventually halts the service provider.

As in the TCP/IP implementations for UNIX, a number of memory structures need to be allocated for each TCP connection request. Take BSD system as an example: a socket structure is used to hold the communication elements (e.g. protocol being used), address information, request queues, buffers and flags [4]. Moreover, there are two extra memory structures with 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. Typically, these structures will use a few hundred bytes of memory [7].

A normal scenario of a TCP connection process starts with a system in LISTEN state receiving a SYN packet, which is to be examined for checksum immediately. If the checksum is incorrect, the packet will be discarded silently, with the expectation that the remote site will retransmit a new packet. Otherwise, the TCP control block associated with this connection is searched for. If no such item is found, it means no server process is waiting for this packet, and then the packet will be removed and an RST packet is returned to inform the remote client. By contrast, if a server process is located, several memory structures will then be allocated for this connection and a SYN_ACK packet will be sent back as an acknowledgement to the sender to continue the three-way handshake. Meanwhile, the system enters into the SYN_RECVD state and starts up a connection establishment timer. The connection of this stage is always called a half-open connection. Most TCP/IP implementations set the timer to expire after 75 seconds. If the final ACK packet arrives before the timer expires, the request will leave kernel space and go to application space. Otherwise, the three-way handshake fails. Under both cases, the corresponding memory structures will be released from kernel space.

From the description above, we know that the process of TCP connection establishment requires significant amount of work and resources at the server side. So, in most systems, there is a limit on the total number of half-open connections. A hacker exploits this limitation and initiates a SYN flooding attack by issuing a large number of connection requests with a spoofed source IP address to the target host, which cannot tell a malicious request from a legal request. After receiving the SYN packet, the target host will respond with SYN_ACK packet as usual. Unfortunately, this time the final ACK packet will never come back, for the request SYN packet has a spoofed source address and that address is "unreachable" to the target host (Figure 2.7). There are several reasons for an IP address to be "unreachable". For instance, the machine with that IP address is turned down, or there may be even no host with that IP address at all. Actually, there may be some error messages like ICMP "host unreachable" or "network unreachable" generated by a router, coming back during the time when the target host waits for the final ACK packet. But current implementations of TCP/IP typically ignore such error messages. Before the timer used for TCP connection establishment expires, the memory allocated for a connection request will stay in the kernel. As large numbers of bogus connection requests come to the target host, it will run out of kernel memory quickly. As a result, if there is no more memory structures can be allocated for the following connection requests, they will be discarded silently.

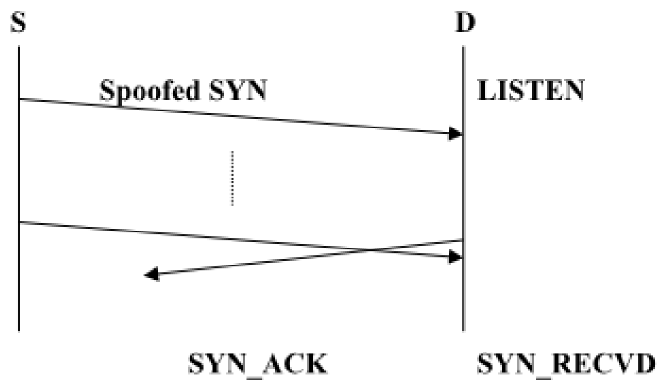

Figure 2.7 SYN Flooding

The key issue in this type of intrusion is how to choose an “unreachable” source IP address for an attacking packet. There are several patterns followed by the hackers.
Single address: all the attacking packet using same IP address
Short list: there is a small pool of addresses for every outgoing packet to choose
No list: the source address is generated randomly
Different addressing method poses different challenges for an intrusion detection system.

The basis for this type of attack is that TCP/IP protocol suite does not provide strong authentication on its control packet [9]. The endpoint of a connection has no way to authenticate its communication peer. As a result, it is extremely difficult to trace the original source of the spoofed IP packet. Therefore, a hacker can feel free to perform this kind of intrusion without worrying about being tracked down.

Other Denial of Service Intrusions

Other forms of denial-of-service type intrusions also exist, like Ping of Death. Ping of Death explores a bug in some TCP/IP implementations that cannot handle the fragmented IP packet correctly. In this case, a hacker first breaks a normal packet into a series of fragments, then modifies the last one and makes the total length of all fragments exceed the maximum packet length specified in TCP/IP protocol. When the receiving host assembles those fragments, it will overflow its buffer in the TCP/IP stack due to the abnormal size of the arrived packet. As a result, system on that host will crash.

There are some intrusions, which utilize the broadcast property of transmission media, are limited to a LAN (Local Area Network) environment, especially Ethernet. However, we cannot overlook those intrusions. It is possible that some hosts are less secure than other hosts on a LAN. A hacker can perform a multi-step intrusion by first breaking into a less secure host and then compromise the whole network. One of the threats to a LAN is called SYN_RST generator, which can block most of the TCP connections. Suppose that a host A wants to make a TCP connection with host B, it will first send a SYN packet to B. If host X also hears this message, because of the broadcast communication media, before B responds with a SYN_ACK packet, X can quickly send out a RST packet to A, shutting down the intended connection. Another intrusion example is one in which the flow control mechanism of TCP communication is attacked. In order to prevent a fast sender from overrunning the buffer of a slow receiver, each TCP packet has a window size for its communication peer. During the communication process, a third party host can impersonate the destination host, sending a packet with zero window size. Then, both communication parties can be halted due to the lack of buffer advertised by the communication peer.

With the increasing use of Java in the web computing, intrusions by malicious Java applets are another source of concern. Most of the applet intrusions fall into denial of service, in which a Java applet consumes a lot of CPU and memory resources of the client machine.

Spoofing

Spoofing is another important hacking technique in the network intrusions. Due to the distributed nature of computer networks, the primary method used to exchange data among different hosts is message passing. Therefore, strong authentication is not easy to achieve compared to that in a traditional centralized system, especially among arbitrary communication peers. A network hacker exploits this weakness and creates many intrusion methods, either spoofing himself as a legitimate client or server.

Client-Side Spoofing

In client-side spoofing, a hacker impersonates himself as an authorized client and in turn gains services from a server. An example is provided by the "r-utilities" on most UNIX systems.

"R-utilities", like rlogin, rsh and rcp, is a set of commands for remote operations among different UNIX systems. The security hole underlying "r-utilities" is the authentication scheme used by this set of commands.
Take "rlogin" as an example, which uses TCP as its transportation layer protocol and is a simple client/server application. With two hosts A and B, each of which "trusts" the other one, we can configure the file "/etc/hosts.equiv" or ".rhosts" on each host to let a user with accounts on both hosts to login from one host to another without being prompted for a password. In effect, the user is authenticated via the host name of the machine he/she is currently logged on.

In 1995, CERT(TM) Coordination Center issued a security advisory addressed a kind of intrusion called "IP Spoofing", in which the hackers created packets with spoofed source IP address, then exploited applications that use authentication based on IP address, like "r-utilities" [7]. IP spoofing consists of several steps and uses both address spoofing and TCP sequence number prediction. Following are two scenarios that can happen, one is a normal "rlgoin" session, while the other is a spoofing intrusion (Figure 2.8). Usually, IP Spoofing takes the following steps

Normal Remote Login Session:

C →S: SYN(ISNc)
S →C: SYN(ISNs), ACK(ISNc)
C →S: ACK(ISNs)
C →S: data
S →C: data

Spoofed Intrusion:

X →S: SYN(ISNx), SRC = C
S →C: SYN(ISNs), ACK(ISNx)
X →S: ACK(ISNs), SRC = C
X →S: ACK(ISNs), SRC = C, malicious messages

Figure 2.8 IP Spoofing

• First, a victim host is selected and a pattern of trust is discovered, e.g. which hosts the victim host trusts. In the example shown in Figure 2.8, the victim host is S, while it trusts host C.
• Then, C is "shut down", either by SYN flooding that machine or by intercepting the entire network traffic to it. Alternatively, the attack may be initiated when C is down due to other reasons, such as maintenance.
• Next, a normal TCP connection request packet is sent to the victim host S to get back a valid sequence number. Based on the round-trip delay and the TCP sequence number generating algorithm, a hacker could predict the next sequence number that will be used by S.
• At this point, S can be intruded upon. The hacker sends a SYN packet to S with the trust host C as source IP address. Even though the SYN_ACK packet will not return to the hacker, he/she can still finish the connection establishment by sending out the final ACK packet with the guessed sequence number from the previous step.
• The victim host S all along concludes a valid connection request from trusted host C. Then, the hacker could send data from host X.

One thing that needs to be clarified in this intrusion is, when the hacker from host X masquerades himself as a trusted client and sends out a SYN packet, the returned SYN_ACK packet from the victim host will go to the real host C. As mentioned in the previous chapter, the host C will immediately respond with a RST packet and the intrusion will fail because the intended connection will be shut down when the victim host S received this packet. Therefore, SYN flooding is always performed as a preparing step in “IP Spoofing”. As a result, the returned SYN_ACK packet would not reach the destination host C but gets lost on the way. The reason is that the host C is busy in dealing with large amounts of bogus requests and runs out of system resources.

IP spoofing is a typical example of client-side spoofing intrusion. All the applications with loose authentication mechanism based on IP address also face the threats from this type of intrusion.

Server-Side Spoofing

Server-side spoofing employs a similar idea. However, the goals and the methods used are a bit different. For the client-side spoofing, as we mentioned in the example of “rlogin”, a hacker impersonates an authorized user and then gains data from an information provider. By contrast, the server-side spoofing is executed in the reverse way. In order to obtain confidential information from individual clients, a hacker masquerades as a real service provider and steals sensitive information from service users.

The idea behind server-side spoofing intrusion can be properly expressed by a real life example. Suppose that someone creates a machine that looks extremely like an ATM but does not provide the real functionality of a normal ATM. Instead, it records the number of an ATM card and its holder’s PIN (Personal Identification Number), then reports some error message to mislead the user that this machine has temporary mechanical problem. If such a machine were placed at the entrance of a shopping mall, the result would be disastrous. A user may lose large amounts of money just because he/she once used an out-of-order ATM several days before.

Same idea is employed in web spoofing. First step is to put some HTML (Hypertext Makeup Language) links in some popular web pages. When a victim visits that page and clicks that link, all the following connections is hijacked by a malicious server, which hides itself between the user browser and the real web server. No sophisticated technique is used in this attack. Some simple Java script applet, together with a little HTTP (Hypertext Transfer Protocol) and CGI (Common Gateway Interface) knowledge, is sufficient to hijack such connections. With the growing popularity of electronic commerce, this type of intrusion becomes even more dangerous. A malicious server can easily grab personal information from a web shopper, such as credit card information.

Service Specific Intrusions

In this section, we survey some service specific intrusions, such as finger daemon attack, routing infrastructure intrusion, DNS misuse and several attacks to NFS (Network File System) or X-Windows system.

Routing Infrastructure Intrusions

As described at the beginning of this thesis, all the TCP/IP services are built on 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 traveling among distributed network devices. In a large-scale and heterogeneous network environment, like the Internet, delivering a packet to the right destination is the task of routing infrastructures.

Internet adopts a hierarchical routing architecture, which relieves a single router from storing huge amount of path information. A router makes routing decision of an IP packet based on a data structure called routing table, which keeps the status of each path linked to that router. If RIP (Routing Information Protocol) is used, a router will periodically generate LSU (Link State Updates) that describe the latest status of the links to the router and disseminate those updates to the other neighboring 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 the spoofing intrusions and some of them can lead to the results of denial of service. A faulty router can modify the packets passing through it or discard the packets at all. This may bring some networks or hosts unreachable. Furthermore, a malicious or compromised router can send bogus routing control packets, like LSU, to other routers, which may in turn cause all the packets switch to itself and it can then eavesdrop the content within the packets. Another scenario is that a router sends bogus LSU's that makes other routers think that some reachable hosts are unreachable.

DNS Misuse

DNS is not a part of TCP/IP protocol suite when it was first proposed. However, with millions of networks and hosts interconnected by the Internet, IP address becomes inconvenient for an end-user to make connections. An alternative approach is to map low-level IP addresses into meaningful hostnames, which is the main motivation of using 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 number of domains. 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.

Most DNS implementations adopt UDP as the transportation layer protocol. So, besides the vulnerabilities of DNS, security flaws from UDP, like lack of state information and weakness in user authentication are also inherited. With a similar architecture as the routing infrastructures, DNS faces the same threats from spoofing type intrusions. A misused name server could be easily used by a hacker to masquerade himself as from any host, for a hacker-controlled name server can intercept a resolver query and can respond with whatever IP address a hacker intends to be. A recently found bug in Java class verifier has a tight relation with this kind of intrusion, in which a malicious applet could connect with any host other than the host from which it was downloaded.

Caching is widely used by DNS to improve the system performance. In DNS specifications, there is a little concern for the data integrity and consistency of caching. Therefore, an intrusion by sending spoofed information to a name server in a straightforward way will not work. Instead, a hacker uses another approach called "Ask Me" to poison a name server's cache by malicious data items [10].

Imagine there is a hacker on host X, who has full control of name server B and intends to provide 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 cannot directly send this malicious mapping to NS A, 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. NS B then appends the above incorrect mapping information at 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 can go ahead and launch some more serious intrusions, for instance, an intrusion towards address-based applications like "r-utilities".

In addition to the likely results mentioned in the routing infrastructure intrusions, such as misleading the packet flow, a DNS intrusion can greatly facilitate the attacks aimed at address-based applications. In sum, a combined intrusion on the DNS system and the routing mechanism can be catastrophic.

INTRUSION DETECTION AND PACKET FILTERING

From the discussion in the previous chapter, we can find most intrusions take advantages of vulnerabilities in the system design and implementation. However, it is impractical for us to eliminate all the errors in the existing systems or replace all the old systems with new error-free systems given the established base of software. An alternative approach in protecting a system from intrusion is to detect and isolate the problem before it can impact the system performance or functionality. In this chapter, we review some basic techniques in enhancing system security and address some general issues in real-time packet filtering, which is important for network intrusion detection.

Current Techniques in Network Security

A number of techniques have been invented in the past few years to help a system administrator in strengthening the security of a single host or the whole computer network. We review a couple of most widely used techniques.

Audit Trails

As defined by the National Computer Security Center in its Rainbow series of system security guide, an audit trail is"A chronological record of system activities that is sufficient to enable the reconstruction, reviewing, and examination of the sequence of environments and activities surrounding or leading to an operation, a procedure, or an event in a transaction from its inception to final results." [8]

Audit trail can be used in determining whether an unexpected or unauthorized behavior has occurred in a system. Therefore, it can be invaluable to a system administrator for network management and security analysis. In practice, almost every operating system used today provides auditing and logging utilities. Most of them are in the form of log files, which record information from a user's most recent login time and a user's originating host, to every message generated by operating system kernel.

The hacker, who knows where to find the log files and how to modify their contents, can easily make a system, look as if nothing has happened. Generally speaking, auditing is a kind of post event protection mechanism. In other words, by the time an intrusion is logged, the hacker may have already broken into the system. Therefore, audit trail may not be very useful in terms of protecting a system from break-ins. Moreover, an experienced hacker can typically defeat or circumvent the auditing mechanisms. Nevertheless, system auditing can be an effective deterrent for inexperienced hackers, since it provides a mechanism to trace their activities.

Firewall

A recent trend in network security enhancement involves the use of firewall, which is a collection of filters and gateways that shield trusted networks within a locally managed security perimeter from the external untrusted networks [1].

Screening Router

Screening router is a router, which in addition to forwarding packets likes a normal router, also examines data in the packets, and applies some predefined access control policies on the packets to determine whether they can be forwarded to the next hop or should be discarded. The packet-filtering function performed by a screening router is implemented by examining a small portion of data in the header part of each packet, such as source and destination address in the IP header and port number in the TCP or UDP header. Meanwhile, some security policies are tested against each packet by the screening router, mostly for the purpose of access control. With carefully configured policies, the screening router can be effective in preventing some classes of network intrusions. For example, we can set up the policy on a screening router as follows:

- Configure the external interface of the router to block incoming packets that have source IP address from the internal network;
- Configure the internal interface of the router to block outgoing packets that have source IP address from the external network.

This is effective in preventing most IP-Spoofing based attacks aimed at or launched from within the local hosts.However, since a screening router does not check the information other than protocol headers, it is unable to prevent attacks that depend on packet content. For instance, it is not capable of detecting attacks such as DNS cache poisoning.

Application Gateway

Due to the above-mentioned limitations of a screening router, various application gateways are created to implement high-level policies in a firewall strategy. As the name implies, an application gateway works at the level of application layer protocols rather than being limited to IP or TCP level. Application gateways provide one or more of the following functionality: relay, proxy and server filter.

Relay gateway, passes the data between the two sides of a firewall system. In some special environments, like a company using "local" IP addresses (i.e. visible only within the company) for internal network, a relay gateway should also provide the function for translating these addresses before they are sent out.

Proxy is of most importance to a firewall system, for most of access control policies are enforced through application proxies. Usually, a proxy gateway is application specific. When a client program inside the firewall requires a connection with an outside server, an application proxy on the firewall will handle the request first. It applies some security policies against the connection request. If the connection request is granted, the proxy will make real connection to the server outside the firewall. Beyond this point, a proxy gateway acts no more than a relay gateway.

Server filter works in the opposite direction as an application proxy. It handles the incoming connection requests from external network to the internal servers. When receiving a connection request, the server filter dispatches it to the corresponding application server. The benefit obtained from using server filter is that we are able to perform access control without changing too much for the original application servers.

As an application gateway examines more data in a network packet than a screening router does, it provides more power in network intrusion detection and prevention. On the downside, it requires more system resources and more processing time. As a tradeoff, modern firewall security systems always adopt a combination of screening router and application gateways (Figure 3.1). Usually, a screening router is placed as the first line of defense, which is used to filter out invalid network traffic by applying the policies against IP address, TCP, UDP port, and so on. Then the packets left are forwarded application gateways, which implement higher-level security policies.

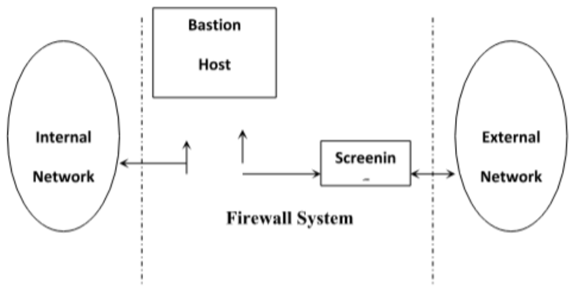

Figure 3.1 Screening Router and Application Gateway

Packet Filtering

Packet filtering technique was invented for diagnostic and analysis purpose in network management. Later on, it began to be used by the network security systems. As we mentioned earlier, it forms the foundation for the firewall strategy. Neither screening router nor application gateway can live without packet filters. At present, the packet-filtering technique also plays an important role in network intrusion detection.

General Issues

The key issues on building a packet filter are:
Real-time performance: the packet filter should be able to quickly capture a raw packet from data link layer and process it in a short period of time.
No packet dropping: no packet dropping is allowed, especially for a network intrusion detection system. The

information missed from dropped packets can make the whole detection scheme fail.

Flexibility: the specification of packet patterns can be modified easily to support different communication protocols.

Scalability: in terms of a system for network intrusion detection, new intrusion signatures can be added into the packet filter without degrading performance.

Summary:

Network Intrusion Detection attempts to discover unauthorized access to a computer network by analyzing traffic on the network for signs of malicious activity. Any activity aimed at disrupting a service or making a resource unavailable or gaining unauthorized access can be termed as an intrusion. Network Intrusion Detection System (NIDS) aims to detect attempted compromises by monitoring network traffic for indications that attempted compromise is in progress, or an internal system is behaving in a manner which indicates it may already be compromised. Intrusion detection system is generally considered to be any system designed to detect attempts compromise the integrity, confidentiality or availability of the protected network and associated computer systems. Intrusion detection systems, or IDSs, have become an important component in the Security Officer's toolbox. In some cases the IDS may also respond to anomalous or malicious traffic by taking action such as blocking the user or source IP address from accessing the network. Network Intrusion Detection Systems are placed at a strategic point or points within the network to monitor traffic to and from all devices on the network. Ideally you would scan all inbound and outbound traffic.

References

[1] Larry J. Hughes, Jr. Actually Useful Internet Security Techniques, New Riders Publishing, Indianapolis, IN, 1995.

[2] R. Heady, G. Luger, A. Maccabe, and B. Mukherjee. A Method To Detect Intrusive Activity in a Networked Environment. In Proceedings of the 14th National Computer Security Conference, pages 362-371, October 1991.

[3] AbdelazizMonnji. Languages and Tools for Rule-Based Distributed Intrusion Detection, PhD thesis, FacultesUniversitaires, Notre-Dame de la Paix, Belgium,

[4] W. R. Stevens. TCP/IP Illustrated Vol. 1 – The Protocols, Addison-Wesley Publishing Company, Inc. Reading, MA, 1994.

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

[6] Morris R. A Weakness in the 4.2 BSD UNIX TCP/IP Software, Computer Science Technical Report No 117, AT&T Bell Laboratories, Murray Hill, NJ, 1985.

[7] CERT. TCP SYN Flooding and IP Spoofing Attacks, Carnegie Mellon University, Pittsburgh, PA, September 1996.

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

[9] 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, West Lafayette, IN, 1996.

Author Details

Kranthi K Lammatha
Chief Technology Officer (ITU)
Email Id: Kranthi@itu.edu