CS425: Computer Networks

Firewall Implementation

Ankit Kumar	Akshay Mittal	Ashish Gupta	Sayandeep Ghosh
Y8088	Y8056	Y8410	Y8465

October 31, 2010

under the guidance of
Prof. Dheeraj Sanghi
Department of Computer Science and Engineering
IIT Kanpur

Contents

1 Objective

Firewall is a system designed to prevent unauthorized access to or from a private network. They can be implemented in both hardware and software, or a combination of both. All datagrams entering or leaving the intranet pass through the firewall, which examines each datagrams and blocks those that do not meet the specified security criteria. Our objective is to build a firewall that blocks unauthorized access while permitting authorized communications using packet filtering.

2 Motivation

Mechanisms that control internet access handle the problem of screening a particular network or an organization from unwanted communication. Such mechanisms can help prevent outsiders from obtaining private information, changing information or disrupting communication on organizations' internal internet. Unlike authentication and privacy mechanisms, which can be added to application programs, internet access control usually requires to basic components of internet infrastructure. In particular, successful access control requires a careful combination of restrictions on network topology, immediate information staging, and packet filters. A single technique has emerged as the basis for internet access control. The technique places a block known as an internet firewall at the entrance to the part of the internet to be protected.

3 Introduction and Overview

A firewall is a set of related programs, located at a network gateway server, that protects the resources of a private network from users from other networks. An enterprise with an intranet that allows its workers access to the wider Internet installs a firewall to prevent outsiders from accessing its own private data resources and for controlling what outside resources its own users have access to.

There are several types of firewall techniques:

- **Packet filter:** Looks at each packet entering or leaving the network and accepts or rejects it based on user-defined laws. Packet filtering is fairly effective and transparent to users, but it is difficult to configure. In addition, it is susceptible to IP spoofing.
- **Application gateway:** Applies security mechanisms to specific applications, such as FTP and Telnet servers. This is very effective, but can impose a performance degradation.
- **Circuit-level gateway:** Applies security mechanisms when a TCP or UDP connection is established. Once the connection has been made, packets can flow between the hosts without further checking.
- **Proxy server:** Intercepts all messages entering and leaving the network. The proxy server effectively hides the true network addresses.

4 Packet Level Filters

Filter based firewalls are configured with a table of laws that characterize the packets that they will, and will not, forward. These laws take into account the IP addresses of source, destination, ports, protocol etc. Commercial routers offer a mechanism that augments normal routing and permits a manager to further control packet processing. Informally called a packet filter, the mechanism requires the manager to specify how the router should dispose off each datagram. For example, the manager might choose to filter (i.e. block) all datagrams that come from a particular source or those used by a particular application, while choosing to route other datagrams to their destination. When a datagram first arrives, the router passes the datagram through its packet filter before performing any processing, if the filter rejects the datagram, the

router drops it immediately.

The filter can be configured in two modes. In the first mode, the default action can be defined to route the datagram while the IP combinations in the laws define the datagrams that need to be blocked. The other configuration is the reverse, i.e. the default action be blocking for the datagram and only the datagrams which abide by the filter laws are routed correctly. The first mode does not work for an effective firewall for three reasons.

- The number of well-known ports is large and growing. Thus a manager would need to update such a list of laws continuously because a simple error of omission can leave the firewall vulnerable to attacks.
- Much of the traffic on the internet does not travel to or from a well known port. In addition, the programmers can choose port numbers for their private client server applications in services like RPC assign port numbers dynamically.
- Listing ports of well-known services leaves the firewall vulnerable to tunneling, a technique in which one datagram is temporarily encapsulated in another for the transfer across part of an internet. Tunneling is used to circumvent security by arranging for a host or router on the inside to accept encapsulated datagrams from an outsider, remove one layer of encapsulation, and forward the datagram on to the service that would otherwise be restricted by the firewall.

It is because of these reasons that, we have configured our filter mechanism to operate in the second mode in which blocks all datagrams by default except those destined for specific networks, host and protocol ports for which external communication has been approved by the laws.

5 Content Based Filtering

Content filtering is the technique whereby content is blocked or allowed based on analysis of its content, rather than its source or other criteria. It is most widely used on the internet to filter email and web access. Content filtering is commonly used by organisations such as offices and schools to prevent computer users from viewing inappropriate web sites or content, or as a pre-emptive security measure to prevent access of known malware hosts. Filtering rules are typically set by a central IT department and may be implemented via software on individual computers or at a central point on the network such as the proxy server or internet router. Depending on the sophistication of the system used, it may be possible for different computer users to have different levels of internet access.

We implemented the content based filtering by allowing the router to peep into the data section of the packet recieved before routing. A set of restricted words is specified each of which is a representative of the prohibited content in a message. The router looks for these words in the datagram and blocks the datagram in case it is found. Note that the content based filtering takes place after the datagram has passed the packet level filtering.

6 Implementation Details

6.1 Law

This is a simple record that hold information associated with a specific law. The law contains the IP addresses of the source and destination that are to be matched against. Port numbers further supplement the information contained in the law.

Field	Description
Source IP	Source IP address that will match the law
Dest IP	Destination IP address that will match the law
Source Port	Source port that will match the law
Dest Port	Destination port that will match the law
Src Mask	A Flag indicating weather the Source IP in law is a network mask
Dest. Mask	A Flag indicating weather the Dest IP in law is a network mask
Action	Accept (Route and Notify), Deny (drop), Reject(Drop and Notify)
Protocol	Protocol which will match: TCP, UDP

Table 1: Structure of law

6.2 Organization of laws

6.2.1 A trivial data structure: Array

- In a conventional packet filtering firewall, the laws are stored in an array.
- Each time a packet is processed, the array has to be scanned top to bottom, regardless of the laws it contains.
- At large, busy Internet junctions, this may cause packet losses which will require upgrading the processor.

6.2.2 Our data structure: LawTree

- A LawTree is a binary tree organization of laws where each internal node TreeNode represents a prefix of an IP address. Each TreeNode extends the prefix of its parent by appending a "0" or "1" depending upon whether it is the left or right child of its parent respectively.
- Each law is encapsulated in a LawNode and sits in the law tree indexed by its source IP address field.
- The choice to sort by the source IP was made based on the fact that popular firewalls filter packets by source rather than by destination addresses.
- A law consisting of an IP address and mask will be inserted at the position you reach by walking down the tree using the significant bits of the IP address. Each node contains a linked list of laws having the prefix associated with the node as the source IP address because for a single source address may exist more than one law (for example, for different network interfaces, destination address, and other options).
- The flags src_mask and dst_mask are used to decide if the source address and the destination address present in the law represent IP addresses of the host machines or the subnet masks for source and destination networks respectively. In the case when these flags are set then the law is encapsulated as a LawNode into the list of that TreeNode whose prefix represents the corresponding mask.
- There is an option to expand the data structure to 2 linked trees (at only memory overhead - no overhead in calculation complexity), where one is sorted by source IP and the other by destination IP to get faster search in case of a destination IP based search.

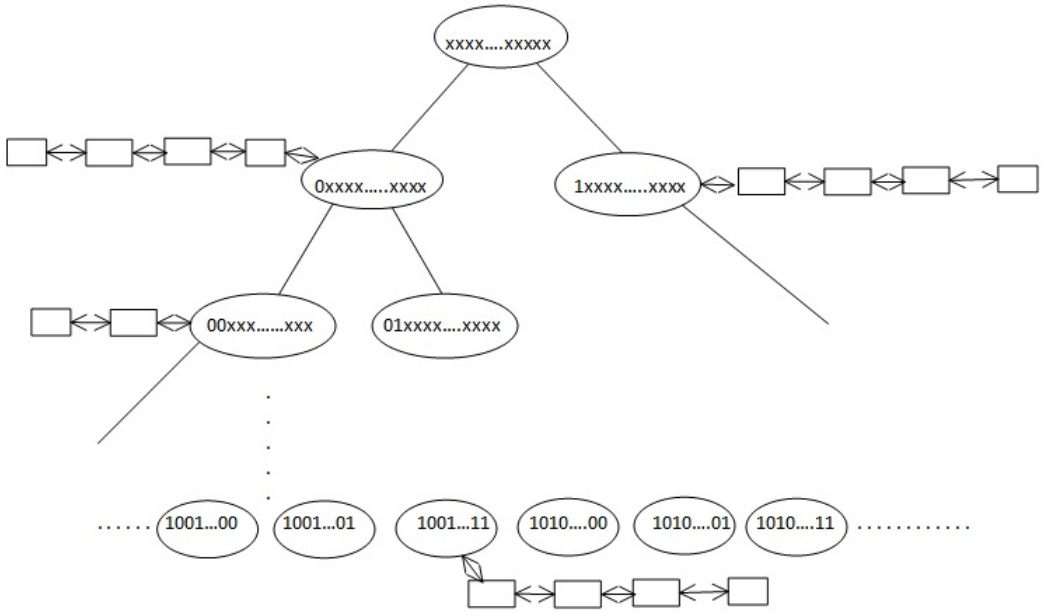

Figure 1: LawTree - The tree organization of the laws specified to the router

6.2.3 Advantages of our data structure:

- For each packet, only the "relevant" laws for that packet are scanned. That is, only laws which refer to IP addresses that include (have same prefix) the packets IP address (directly or by mask) will be scanned.
- Inserting/removing a law into the tree may be a bit costly, but we decided that the most important factor is the speed during filtering operation (i.e. searching within the tree which is done all the time) and not the speed during insertion / removal (which are done periodically by the system administrator).

6.3 Headers

A packet header is the portion of an IP (Internet protocol) packet that precedes its body and contains addressing and other data that is required for it to reach its intended destination.The header's format is specified in the Internet protocol. It normally contains 20 bytes of data, although an option exists within it that allows the addition of more bytes.

In our implementation, we have used the **struct ip** structure as an header to the payload that is sent. ip is a struct (structure) in the C programming language. The ip struct is used as a template to form an IPv4 header in a raw socket. The structure can be found in the default include files of most Unix distributions. It is most commonly located in the *netinet/ip.h* header file.

TCP segments are sent as internet datagrams. A TCP header follows the internet header, supplying information specific to the TCP protocol. This division allows for the existence of host level protocols other than TCP.

In our implementation, we have used the **struct tcphdr** structure as an header for the segment following

Attribute	Description
unsigned int ip_hl:4	umber of 32 -bit words forming the header, usually five
unsigned int ip_v:4	Always set to the value 4 in the current version of IP
uint8_t ip_tos	Usually set to 0, but may indicate particular Quality of Service needs from the network. This helps the router in taking the right routing decisions.
uint16_t ip_len	It includes the IP header and everything that comes after it.
uint16_t ip_id	The source and ID field together will represent the fragments of a unique packet. So each fragment will have a different ID.
uint16_t ip_off	It is a 13 bit field that represents where in the packet, the current fragment starts.
uint8_t ip_ttl	Specifies the number of hops within which the packet should be delivered or else destroyed.
uint8_t ip_p	Specifies the module to which we should hand over the packet (UDP (17) or TCP (6)).
uint16_t ip_sum	The header checksum. Every time anything in the header changes, it needs to be recalculated, or the packet will be discarded by the next router.
struct in_addr ip_src	Source IP address
struct in_addr ip_dst	Destination IP address

Table 2: struct ip

TCP protocol. TCP segment header tcphdr is a struct (structure) in the C programming language. The tcphdr struct is used as a template to form a TCP header in a raw socket. The structure can be found in the default include files of most Unix distributions. It is most commonly located in the *netinet/tcp.h* header file.

The tcphdr struct is unique in that it was written in two different formats, a BSD format and a Linux format. We have used the BSD format, so we add *#define __USE_BSD* and *#define __FAVOR_BSD* at the very top of our definitions.

Attribute	Description
u_short th_sport	the source port number
u_short th_dport	the destination port number
tcp_seq th_seq	The sequence number is used to enumerate the TCP segments. The data in a TCP connection can be contained in any amount of segments (single tcp datagrams), which will be put in order and acknowledged.
tcp_seq th_ack	Every packet that is sent and a valid part of a connection is acknowledged with an empty TCP segment with the ACK flag set
u_int th_x2:4	Variable in 4 byte blocks. The x2 variable is deprecated, it should be set to all binary zeros
th_off:4	The segment offset specifies the length of the TCP header in 32bit/4byte blocks.
u_char th_flags	This field consists of six binary flags - URG, ACK, PUSH, RST, SYN, FIN
u_short th_win	The TCP window - the amount of bytes that can be sent before the data should be acknowledged with an ACK before sending more segments.
u_short th_sum	The checksum of pseudo header, tcp header and payload
u_short th_urp	Urgent pointer. Only used if the urgent flag is set, else zero.

Table 3: struct tcphdr

In our implementation, we have used the **struct udphdr** structure as an header for the segment following

UDP protocol. UDP segment header udphdr is a struct (structure) in the C programming language. The udphdr struct is used as a template to form a UDP header in a raw socket. The structure can be found in the default include files of most Unix distributions. It is most commonly located in the *netinet/udp.h* header file.

Attribute	**Description**
u_short uh_sport	the source port number
u_short uh_dport	the destination port number
short uh_ulen	the udp length
u_short uh_sum	checksum

Table 4: struct udphdr

6.4 A Working Example

Consider the following set of laws specified to the router.

src_addr	**src_port**	**dst_addr**	**dst_port**	**action**	**src_mask_flag**	**dst_mask_flag**	**protocol**
172.24.32.14	5000	172.24.32.15	5001	0	0	0	6
172.24.0.0	5010	172.24.32.16	5011	2	1	0	17
172.24.12.1	5000	172.24.0.0	5001	0	0	1	6

Restricted words -	terrorist	bomb	suicide

Consider a TCP source packet originating from the machine having IP address 172.24.32.14:5000 destined to the machine 172.24.32.15:5001. The datagram is first routed to the firewall. At the router (which is running the firewall), the firewall extracts the required fields from the packet header. It then traverses the LawTree according to the source IP. On traversal, we get a match at law 1. The action demanded by the law is to accept the packet and hence, the packet is forwarded to the destination and a message is sent to the source informing that the datagram has been routed to the destination.

Now consider a UDP source packet originating from the machine having IP address 172.24.32.14:5010 destined to the machine 172.24.32.16:5011. At the router, similar procedure as above is followed, with the exception that the src_mask flag is set. This means while traversing the LawTree an internal node is matched which contains a non-empty list of laws which apply to all the addresses matching the sub-tree rooted at this node. Hence, in this case the datagram is accepted and desired action needs to be taken. The action bit is set 2 which implies the packet is to be dropped and the source is informed.

Now consider a TCP source packet containing the word "terrorist" and originating from the machine having IP address 172.24.12.1:5000 destined to the machine 172.24.32.15:5001. Following the above guidelines, the packet is accepted by the router after packet-level filtering. Now the packet is filtered using content-based filter in which it is dropped as it matches a restricted word "terrorist".

7 Conclusion and Future Work

We have succesfully implementated packet-filtering based firewall using BSD Socket Programming in C. The firewall has been tested in the live local network at IIT Kanpur. Future work includes extension of this implementation to other firewall techniques such as application gateway, circuit-level gateway, proxy server to name a few. Also, this implementation can be extended to protocols other than TCP and UDP.

8 Acknowledgements

We would like to thank the course instructor Dr. Dheeraj Sanghi for giving us this wonderful opportunity and his guidance throughout the course and making this learning experience enjoyable for all. We are also thankful to the course TAs for their constant monitoring and support which enabled us to complete the project.

References

[1] Douglas E. Comer, *Internetworking With TCP / IP Principles, Protocols and Architecture*, Vol I, Prentice-Hall Pvt Ltd.

[2] Larry L. Peterson and Bruce S. Davie *Computer Networks A Systems Approach*, Edition 3, Morgan Kaufmann Publishers.

[3] Wikipedia *Firewall (Computing)*, `http://en.wikipedia.org/wiki/Firewall\_(computing)`

[4] *IP Spoofing with BSD Raw Sockets Interface*, `http://www.enderunix.org/docs/en/rawipspoof/`