VICTORIA UNIVERSITY OF WELLINGTON
Te Whare Wananga o te Upoko o te Ika a Maui

Department of Computer Science

PO Box 600
Wellington
New Zealand

Tel: +64 4 471 5328
Fax: +64 4 495 5232
Internet: Tech.Reports@comp.vuw.ac.nz

Security in an Open Network Environment

Mark Davies and John H. Hine

Technical Report CS-TR-95/16
May 1995

Abstract

The dramatic growth of the Internet has brought the information superhighway to our doorstep. It presents a new concept of public networking, offering new forms of communication and services. Most organizations now accept that a connection to the Internet is inevitable, but are concerned about the new security threats that arise from attaching their private, enterprise network to a public network. This paper will review the nature of threats that arise and discuss the tradeoffs that must be considered in developing a security policy. Design alternatives for firewalls will be discussed, with an emphasis on the strengths and weaknesses of the different designs. A range of available tools will also be considered.

Publishing Information

This technical report was presented at the UniForum NZ '95 Conference at Masterton, NZ in May, 1995. It is reproduced as a technical report because of the limited circulation that conference receives.

Author Information

Mark is the network manager and John the professor in the Department of Computer Science at VUW. Email (Mark.Davies|John.Hine)@comp.vuw.ac.nz

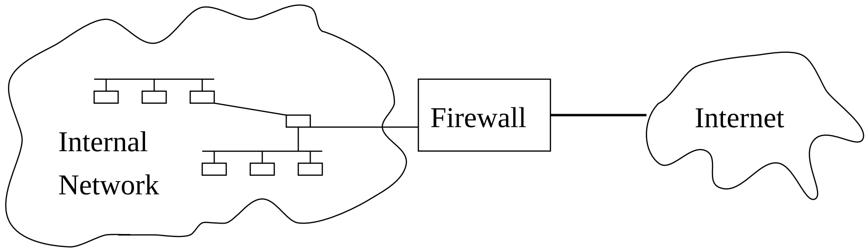

Figure 1: A firewall protects an enterprise's internal network from arbitrary access from the Internet.

Introduction

New Zealand has one of the highest penetrations of Internet hosts in the world. The rapid expansion in Internet connections is causing a variety of organizations to consider whether or not they should connect to the net, the Internet services they are seeking access to, and what services they may offer. Current commercial services are generally restricted to the provision of information to existing customers and clients, however demand for new services is expected to grow.

Most organizations have experience with maintaining the security of a private, enterprise network. The public interface of such networks is generally limited to modem pools and the principle security threat arises from attempts to gain dial-in access. Security in this area has been developed over a number of years and is well understood.

The Internet is a public network, the computer or data equivalent of the telephone system. Connecting an enterprise network to the Internet changes the security situation in a major way. Potential intruders now have a variety of paths of attack and system administrators are faced with a new set of complex security problems.

In this paper we will address the problem of securing an enterprise network from unwanted intrusion from the Internet, principally by the use of a firewall. A firewall acts as a gateway between an internal (enterprise) network and an external network. See figure 1. The goal of the firewall is to support controlled access to required services and to prevent other unauthorized access. Depending on the complexity of its requirements the firewall may be as simple as a single router, or as complex as a separate subnetwork of several devices.

We will not attempt to cover the area of secure and authenticated transactions that will be required to carry out commerce on the Internet. Neither will we go into the full details of how to construct a secure firewall. This is beyond the scope of this paper and readers are referred to the book by Cheswick and Bellovin[CB94]. We will also avoid the murky areas of the legal and ethical issues of monitoring traffic and data. Implementors of firewalls should be aware of the implications of current privacy legislation.

Security Policy

There is general agreement among the top experts in the network security field that absolute security is impossible to achieve. As Gene Spafford puts it:

> *The only system which is truly secure is one which is switched off and unplugged, locked in a titanium lined safe, buried in a concrete bunker, and is surrounded by nerve gas and very highly paid armed guards. Even then, I wouldn't stake my life on it.*

Further, there are aspects of security other than the network connection. A great deal of information can be removed from an organization by carrying a DAT tape through the front door. Network security must be designed as a part of an overall security policy which (1) balances the cost of security against the reduction in risk, and (2) considers all risks to the security of information.

In considering the resources to be devoted to network security an organization should take into account:

1. The requirement for internal access to the Internet. Which Internet services are of benefit to the organization? How great is the benefit? It is much simpler to design a firewall that is an e-mail only gateway than it is to design one which allows full access to all Internet services.

2. The requirement for external access from the Internet to the organization. For example, does the organization use the Internet to present information to the public or to provide product support for its customers? Is the organization considering financial transactions through the Internet interface?

3. The cost of the proposed design, both to implement and maintain. A complex firewall supporting a variety of services will be more difficult to design and implement and will require vigilance of log information to maintain its integrity. A more complex firewall is also more likely to contain a bug.

4. The existence of other, complementary, security measures. Roughly speaking the amount of security against different paths of attack should be balanced. The balance should reflect the perceived risk of attack from each alternative path.

Defining a comprehensive security policy is a first step to a careful definition of requirements. It will define the function required of a gateway between the organization's network and the Internet. It will also specify the degree of security that is to be achieved.

The Attack

While there is always the risk of an opportunistic attack, security planning should assume an attack will be well planned and carefully managed. An organization able to repel such attacks is unlikely to have problems with occasional, opportunistic threats. The following analogy indicates that, in its general form, a network attack is no different than a household break in.

The Household Robbery

1) Case the target. Find the entrances and exits, locate the security devices, and identify the routes to the valuables. This may be accomplished by casual observation or by obtaining a copy of the blueprints.

2) Develop a plan of attack by identifying potential weaknesses and matching these to the available tools.

3) Carry out the break in according to plan. Note anything that might be of use in future robberies.

The Network Attack

1) Case the target. Build a road map of the organization's network. Find the routes to likely targets, identify potential security weaknesses, and locate the important services (DNS, mail, etc.).

2) Develop a plan of attack by identifying potential weaknesses and matching these to the available tools.

3) Attack the target. If (partially) successful plant trojan horses to facilitate a return visit.

4) Disappear into the night.

4) Disappear into the night.

We now examine these steps in more detail.

Reconnaissance

Knowledge of an organization's network is a significant aid to the potential intruder. The names of machines provide potential targets. Network addresses lead to complete maps of the network.

Complete knowledge of an organization's network opens up paths of attack that would not otherwise be available.

1. A routing attack may be mounted using the source routing option of IP. RFC1122[Bra89] requires that the destination of a source routed packet respond using the reverse route. This allows an intruder to construct an IP packet which appears to have originated at a trusted host yet still contains the intruder's system or another compromised host on the path. The target believes it is exchanging packets with a trusted host when, in fact, it is exchanging packets with the intruder. This is one form of *address spoofing*.

2. Certain routing protocols (RIP, OSPF) also have weak security and may believe a bogus attempt to update their routing table. An update that entered a specific host route would likely go unnoticed and could be used to cause packets destined for a particular host to be sent to a compromised host on the same local network.

3. Another trick that may convince the target or a nearby router to incorrectly route a datagram involves using ICMP redirects to insert a host specific route into a routing table. Subsequent datagrams will be routed to the address specified by the ICMP redirect.

4. An alternative point of attack would be the Domain Name System. A number of known weaknesses in this system have been removed, but it is generally assumed that if you must identify a host as trusted or not trusted you are better to use its IP address than its domain name.

5. DNS databases often contain host information (`HINFO`) records that identify the types of system you are running. This can be used by an attacker to identify systems on which known security bugs exist.

There are advantages in preventing the leakage of information across your gateway. Many organizations run mail servers which present a single uniform interface to the outside world. All e-mail appears to have come directly from `username@company.co.nz`. "`Received:`" lines which were generated internally and give both the domain name and IP address of hosts which have handled the message are stripped. A carefully configured DNS service can completely separate the internal and external networks' name databases.

Planning

Given knowledge of a network the intruder will now plan an attack and begin to probe for weaknesses. There may be known weaknesses in particular operating systems or applications. For example, the security problems with Sun's NIS are documented in [HSP92].

Many of the tools that a system administrator may use to check for weaknesses, such as standard accounts without passwords, are also available to the bad guys. Their tools can methodically probe the network in search of these weak points. Needless to say it only takes one weak link for security to be breached.

Protection against such probing comes from knowledge and methodical monitoring. The system administrator should be aware of all potential threats in all systems employed. These include accounts which must be secured or removed, applications which must be carefully configured for security (e.g. mail), the need for good passwords, etc. The administrator should then use tools such as COPS[FS90] and the Internet Security Scanner (ISS) regularly to check that security is being maintained. At VUW we run COPS nightly and ISS regularly.

Attack!

The first goal of most attacks is to capture a copy of the password file. Over the years the techniques for minimizing the risk of a break in through cracking a password have improved significantly. (See the side bar.) However, if an intruder is able to obtain a copy of a password file and remove it for subsequent analysis, it must be assumed that a number of passwords will be cracked. This will compromise at least one, if not a number of hosts and give the attacker an internal platform from which to mount subsequent attacks.

There are a number of tricks that an attacker may use to obtain a copy of the password file. In Unix the password file has traditionally been a "public" file (`/etc/passwd`) that contains information read by a number of programs. It this case it may be possible that TFTP will be happy to transfer it or that it can be accessed by anonymous FTP. Both of these can be prevented by careful administration.

It is quite common for TCP/IP based applications to use text based protocols. The best known of these is SMTP, the Simple Mail Transfer Protocol. It is then possible, using telnet, to directly access the protocol and enter commands. This opens a further route of attack – scan the ports on a targeted host looking for a service that will accept a connection. Then attempt to trick that service into accepting the connection and providing the password file.

A fuller discussion of security problems with the TCP/IP protocol suite will be found in [Bel89].

There are a range of security measures that can be applied here. Firstly, use a shadow password file. Most Unix systems now come with shadow password files. Avoid any systems that don't, for roles that may expose them to the Internet. Use various filters on your firewall (see the next section) to prevent an attacker scanning ports or doing any general form of repetitive probing.

Invasion

Earlier we used an analogy between a household break in and a network attack. This analogy breaks down once the break in/attack succeeds. In a robbery there is usually a well defined target, and the thief quickly departs. Only in special circumstances would the thief worry about hiding the fact that the break in had occurred.

Unfortunately, successful network attacks often lead to a desire to return again—perhaps to search for further information or to use the victim to host a subsequent attack[1]. Having succeeded in breaking into one machine an attacker will then move on to:

- destroy any evidence of the break in,
- export the password file for subsequent analysis if this wasn't done as part of the break in,

[1] Network thieves will work hard to hide their identity. A common technique is to attack new targets using a route that makes it difficult to trace the original source of the traffic. Previously compromised hosts are useful as intermediate nodes in such routes.

- obtain `root` access on the compromised system to gain access to the shadow password file,
- possibly set up a password sniffing tool,
- plant a trojan horse that will enable reentry in case the original break in is discovered and the security hole fixed,
- obtain information about other hosts that trust the compromised host, for example `/etc/hosts.equiv` and user's `.rhosts` files, and/or
- plant a time bomb to cause damage at some time in the future.

The conclusion is that any detected break in must be taken seriously. What may appear to have been a simple, opportunistic break in, perhaps by a school student, could be the beginning of a major invasion. If this occurs the prudent system administrator will rebuild the complete system from an archive that is know to be secure. While this is a major task it is the only safe alternative.

The Firewall Defense

As a brief review, figure 2, reminds us of the basic layers of the TCP/IP architecture. Below IP we have a media access or network layer such as ethernet or HDLC. This layer has its own, protocol specific, addressing. The IP layer provides internet addressing and routing and a datagram based message delivery system. Above IP we find the transport layer which provides end-to-end connections between ports. The principal protocol at this level is TCP, a connection oriented protocol with a wide range of services. The less used alternative is UDP, a simple datagram protocol that does little more than add port addresses to IP datagrams. Various applications run on top of the transport layer.

The major approaches in the design of a firewall correspond to the layers in the architecture. The first approach is to monitor all IP packets flowing through a router at the boundary between the internal network and the Internet and filter out any packets not appropriate for the security policy which has been defined. The second and third approaches do not allow direct IP level communication between nodes on the internal network and those in the outside world, but instead, require that all communication go through a 'proxy' running on a gateway machine that is part of the firewall. This proxy will either be a general purpose transport circuit gateway or an application specific gateway. In general, firewalls operating at higher levels in the protocol stack have more context available to them and are able to successfully operate more fine grained policies than firewalls using lower level protocols.

IP Packet Filter

IP packet filtering, as the name suggests, involves looking at each IP packet as it passes through a router. The protocol information in the IP and transport (TCP or UDP) headers are compared with a set of rules to determine if the packet should be routed normally or filtered out. Most commercial routers provide some filtering capability but mechanisms and actual capabilities differ.

An IP filter is ideally suited to restrict access based on the host or network address. For instance, in figure 3 the gateway router may be configured to:

allow users on network 130.195.5.0 access to the NZ Internet and 130.195.6.0,
allow users on network 130.195.6.0 access to 130.195.5.0, but not to the NZ In-

ternet, and
allow Internet users access to hosts on 130.195.5.0, but not 130.195.6.0.

Unfortunately, you will generally want finer control than this, restricting access according to particular hosts and services.

Filters are normally implemented as a table of rules. Each rule has a set of conditions and an action—'permit' or 'deny'. A packet is compared against each row of the table in turn, until the first which matches all conditions. The associated action is then performed on that packet. Typically these tables have an implicit rule at the end that matches any packet and has an action of 'deny' so if packets are not explicitly matched in the rules they are dropped. Each packet is handled independently.

The following two line table[2] is sufficient to implement the above example from figure 3.

	action	src	s-port	dest	d-port	flags	
1	permit	130.195.5.0/8	*	*	*		traffic from subnet 5
2	permit	*	*	130.195.5.0/8	*		traffic to subnet 5

In this example there are only six possible traffic flows. The table allows four and implicitly denies the two between subnetwork 6 and the NZ Internet.

A slightly more complex example will begin to reveal the weaknesses of the IP filter. Consider a tightening of our previous policy to allow any host on subnetwork 5 to continue to send mail (connect to TCP port 25) directly to external machines, but to restrict external machines to mail only connections to a mail gateway at 130.195.5.2. We have also identified `badhost`, which we will not allow to talk to us at all.

	action	src	s-port	dest	d-port	flags	
1	deny	`badhost`	*	*	*		block badhost
2	permit	130.195.5.0/8	*	*	25		our outgoing SMTP
3	permit	*	25	*	*	ACK	their replies
4	permit	*	*	130.195.5.2/0	25		incoming mail
5	permit	130.195.5.2/0	*	*	*		gateway can do anything

The 'ACK' flag in rule three is necessary to allow external mail hosts, accepting connections from subnetwork 5, to properly acknowledge connections from internal machines. It forces the packet to have the ACK flag enabled in the IP flags header, which indicates the packet is part of an already established connection. Without the ACK flag, outside sites would be allowed to make an arbitrary connection to a host on subnetwork 5, provided the connection was initiated from port 25.

Even this simple example demonstrates the major problem with packet filtering: complexity. Specifying a set of rules to correctly implement your policy is difficult. Real world implementations may add complexity by offering fewer features, for example not being able to specify the source port in rules, or rules only being applied to packets going out a particular interface, not in.

To add to the complexity, protocols like FTP have the actual file transfer occur on a secondary connection that is initiated, on an arbitrary port, by the remote machine. Outgoing X connections have a similar problem, as the server is the local machine and the client the remote host. If you allow connections at all, you have to allow them from arbitrary ports. A final complication is applications that don't use well known ports at all, but rather use

[2] IP address filters allow a mask to be specified that indicates the bits to be considered and those to be ignored—generally to separate a host from a network. In these examples we have represented this by `IP-address/mask`, where the mask indicates the number of trailing bits to ignore.

a random port and rely on some other method for clients to find them. Examples are Sun RPC based applications which use portmapper to locate services and world wide web servers which use arbitrary URL's. A filter with its static tables has little chance of handling these sorts of applications.

Logging is a desirable feature in a firewall. It can assist you in identifying repeated attempted attacks, and take steps to handle the situation. This is another area where filters in commercial routers fail—they tend to apply their rules silently, not logging packets which are denied access.

Circuit Gateway

A circuit gateway is a proxy that relays TCP connections. They are most useful for outgoing connections but can be used to tunnel external connections to specific services as part of the application gateways discussed below.

To establish an outgoing TCP connection the caller connects to a port on a gateway machine that is part of a firewall. The gateway validates the connection in some way and, if the connection is valid, opens an ongoing connection to the true destination. Subsequently, the gateway relays bytes back and forth between the internal and external hosts.

A connection to a circuit gateway may be made transparent to an application by configuring the system to make a connection to a particular port on the gateway that will always cause the connection to be forwarded to a fixed external host and port. However, the more usual case requires a protocol at connection set up, to pass the required destination and validation information to the proxy. This points out the major drawback of this type of gateway: applications need to be modified to be used with it.

Socks [KK92] is a popular circuit gateway implementation that attempts to minimize the problems imposed by modification. Socks provides a set of almost compatible replacements for the standard UNIX sockets system calls: bind, connect, etc. Many applications are starting to appear that have been 'socksified' so no local modification is required.

Packages such as socks address the "reverse connection" problem by causing temporary ports to be created on the gateway. The remote client is notified of the port to use for the connection and the gateway listens for the expected connection. The risk of an attack using the port is minimized by disabling the port if a connection does not occur within a short period of time.

Application Gateway

Firewalls based on application level gateways have the greatest context to work with, can offer the best protection and the highest level of service. Their disadvantage is in the cost; each application must be handled seperately. An application gateway is a proxy that is designed for a particular application and can make use of knowledge about the application to provide finer control over access and detailed logging of all traffic. When a firewall is based on a set of application gateways, only traffic from those specific applications can move between internal hosts and the outside world.

The transparency of an application level gateway depends on the underlying protocol of the application. For instance, a telnet gateway would tend to require a two step process—explicitly telneting to the gateway and then from there to the actual destination. The telnet protocol has no facility for indirection. On the other hand, mail or world wide web proxies are much more transparent to the user, as the protocols cope with indirection through a third party.

In some cases, the application proxy provides other real benefits as well as increased security. With mail the proxy provides a single point at which internal e-mail addresses can be mapped using a consistent scheme for the rest of the world to see, such as `Firstname.Lastname@org.nz`. World wide web proxies provide a point where pages may be cached, reducing the external network traffic.

The TIS toolkit[RA94] is a freely available set of application gateways for some of the common Internet applications. It is designed to be informally verified for correctness as a whole or at the level of individual components. Each component is designed to be independent and as simple as possible consistent with achieving its goal. This allows a fair degree of reasoning about its correctness.

Although not an application gateway as such, TCP wrapper[Ven92] is a program that can be used to add some validation and logging to applications which do not support these features. To use the wrapper you install it as the program `inetd` will invoke whenever a service is requested. The wrapper looks up the IP source address and accepts, rejects and/or logs the connection. Finally, it hands the connection off to the appropriate service.

Designing a Firewall

In this section we present several alternative firewall designs. The designs vary in complexity, function and cost. While full implementation details are beyond the scope of this paper, we hope the presentation of several alternative designs will demonstrate how the various tools (IP filters, circuit gateways and application gateways) can be combined.

Recall from our earlier discussions the importance of identifying the policy you are trying to support. Different policies will require different function in the firewall. (In choosing a firewall design remember that your policy may change!) Because of the obvious weaknesses in an IP filter based firewall we will not consider that option any further.

A Bastion Host

The simplest approach, and the one implemented by most of the commercially available firewall products is the use of a dual homed gateway as shown in figure 4. A gateway sits between the internal network and the rest of the world with an interface on both. The host is configured to **not** route packets between the two (in UNIX systems IP forwarding is switched off). This 'bastion host' is the only machine that is visible to the outside world. With this approach all communication between the internal network and external Internet requires the use of circuit or application gateways on the bastion host. Since routing is disabled on the bastion host an external router is required to route datagrams headed for the Internet.

A typical approach is to install application gateways for incoming connections. These gateways provide the maximum security against external attacks. Internal connections can be handled with either circuit or application level gateways—whichever suits the local environment and services to be supported.

The bastion host approach is reasonably inexpensive, but does suffer from several shortcomings, most of which arise from the need for the gateway to handle both incoming and outgoing connections. Performance is an obvious concern. Generally, the kernel should be reconfigured to assist it in handling the high volume of network traffic. Very careful administration of the DNS is required to both make it available to the gateway and to prevent information leakage to the outside.

There are a few standard principles that should be followed in configuring a bastion host:

- Access to the console should be restricted to the console itself or a separate dial-in facility.
- The only network services allowed are proxies.

- All unnecessary software should be removed to minimize administration and remove possible hosts for trojan horses.

Finally, it should be noted that the bastion host is the one and only line of defense. If it is compromised there is no secondary defense that may slow an intruder enough to allow an administrator to notice the problem.

Dual Gateway

The shortcomings of the bastion host approach have led to more sophisticated and complex designs. Since most of the problems arise from the single gateway handling both incoming and outgoing connections, an obvious approach is to introduce a second gateway host as shown in figure 5. The private network between the two gateways is frequently referred to as a DMZ.

This approach separates responsibilities with the external gateway accepting incoming connections for services such as FTP and SMTP. It also handles DNS for the external world. The internal host offers only a few carefully chosen services to the external host. In particular the internal host does not trust the external host and forms a second line of defense should the external host be compromised.

Although the only host the external gateway can reach is the internal gateway, routing must now be enabled. A careful approach is to configure static routing to avoid protocols such as ARP announcing an address that might be learned (or subverted) by an intruder.

An alternative design for a dual gateway system replaces the DMZ with a router. The router is carefully configured to only accept incoming traffic from the external gateway destined for the internal gateway. Logically, this setup is equivalent. Its disadvantage is the provision of a second point of attack (the router) for the external world. It has performance advantages where outgoing connections are not required to go through a gateway.

Screened Subnet

The screened subnet design shown in figure 6 is another popular approach. The bastion host is placed on a subnet that is protected on each side by routers forming a 'demilitarised zone'. The routers are configured to force all traffic through the bastion host. A potential weakness of this design is the exposure of the second device (the router) to the outside world.

The details of the various implementations of these designs may be found in Cheswick and Bellovin's book, *Firewalls and Internet Security, Repelling the Wily Hacker* [CB94].

Conclusion

No paper can be definitive on the threats and counter measures employed in network security. Network software is large and complex and very difficult to get correct, both in design and implementation. In this paper we have presented a range of threats that may arise from an Internet connection. We have also outlined alternative firewall designs. The more complex firewalls offer greater control over the services allowed. They are also more costly and perhaps more prone to incorrect implementation. The bottom line is to understand the problem thoroughly, plan carefully and monitor continuously!

References

[Bel89] Steven M. Bellovin. Security problems in the tcp/ip protocol suite. *Computer Communication Review*, 19(2):32–48, April 1989.

[Bra89] Robert Braden. Requirements for internet hosts – application and support. Technical report, RFC 1122, October 1989.

[CB94] William R. Cheswick and Steven M. Bellovin. *Firewalls and Internet Security Repelling the Wily Hacker*. Professsional Computing Series. Addison-Wesley, Reading MA, 1994.

[FS90] Dan Farmer and Eugene H. Spafford. The cops security checker system. In *USENIX Conference Proceedings*, pages 165–170, Anaheim, CA, Summer 1990.

[HSP92] David K. Hess, David R. Safford, and Udo W. Pooch. A Unix network protocol security study: Network information service. *Computer Communications Review*, 22(5):24–28, October 1992.

[KK92] David Koblas and Michelle R. Koblas. Socks. In *Proceedings of the Third USENIX UNIX Security Symposium*, pages 14–17, Baltimore, Maryland, September 1992. USENIX.

[MT79] R. Morris and K. Thompson. Password security: A case history. *Communications of the ACM*, 22(11):594–597, November 1979.

[RA94] Marcus J. Ranum and Frederick M Avolio. A toolkit and methods for internet firewalls. In *USENIX Summer 1994 Technical Conference*, Boston, MA, June 1994. USENIX.

[Ven92] Wietse Venema. TCP WRAPPER: Network monitoring, access control and booby traps. In *Proceedings of the Third USENIX UNIX Security Symposium*, pages 85–92, Baltimore, Maryland, September 1992. Usenix.

Passwords

Nearly two decades have passed since the publication of Morris and Thompson's[MT79] classic on passwords. The author's demonstrated that a high level of success could be achieved by building a list of potential passwords from the names and numbers found in a corporate phone directory and automobile registrations found in the parking lot. They also analyzed the work required to break passwords of different lengths from different alphabets, and introduced salting as a technology that significantly increased the work required to break a password.

Despite this well known work it is still worthwhile to stress the need to monitor user's passwords. We don't encourage the assignment of arbitrary passwords to users. This simply encourages them to write them down. However, we do require our users to choose passwords that meet stringent requirements. We specify a minimum length, an appropriate mixture of alphabetic, numeric and special characters and apply a dictionary search before accepting a password.

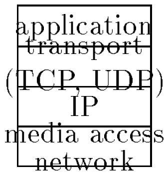

Figure 2: The TCP/IP Architecture

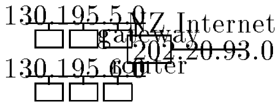

Figure 3: A router which implements IP filtering is ideal for restricting packet flow among networks.

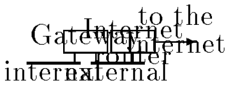

Figure 4: A bastion host firewall.

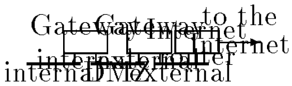

Figure 5: A firewall with dual gateways.

Figure 6: A screened subnet firewall.