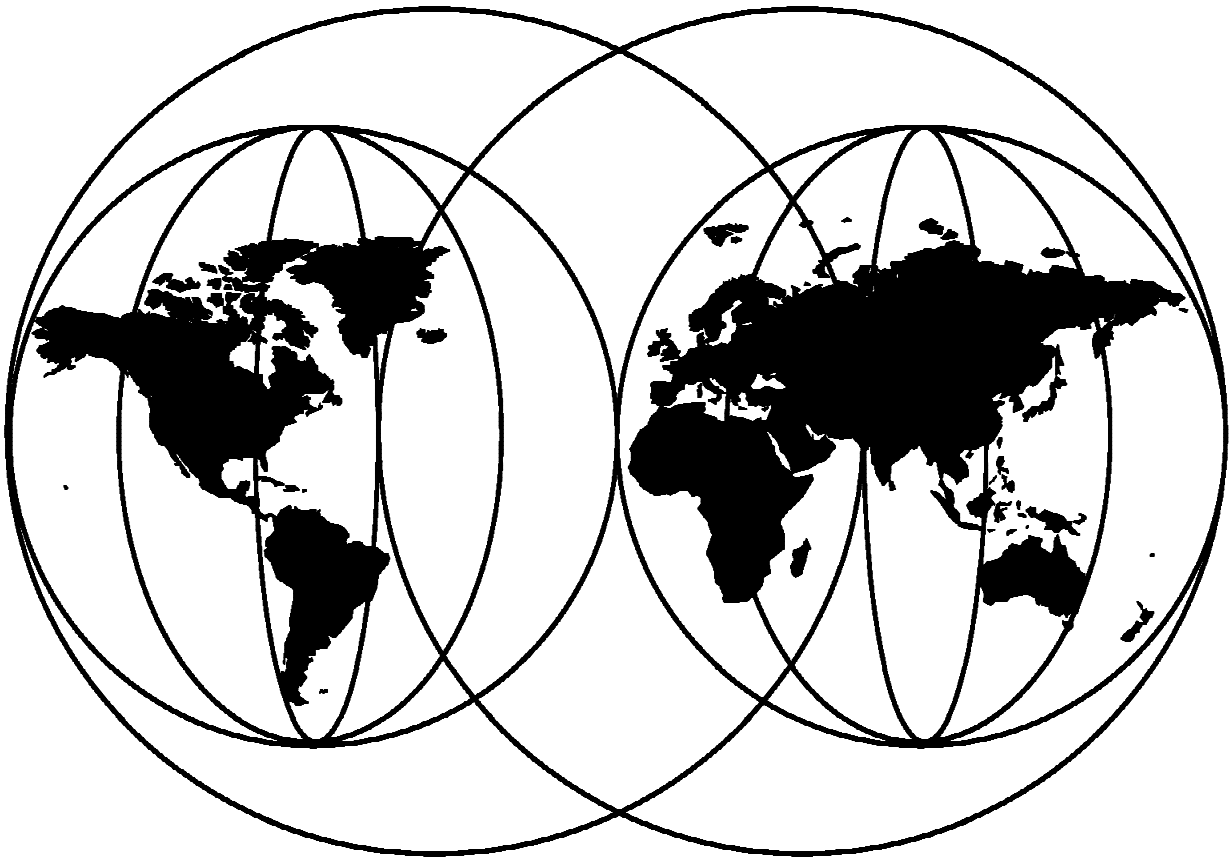
IBM

Protect and Survive Using IBM Firewall 3.1 for AIX

E. Luk, K. Majewski, D. Raxworthy, J. Ferrari



International Technical Support Organization

http://www.redbooks.ibm.com

SG24-2577-02

International Technical Support Organization

SG24-2577-02

Protect and Survive
Using IBM Firewall 3.1 for AIX

February 1998

Take Note!

Before using this information and the product it supports, be sure to read the general information in Appendix G, “Special Notices” on page 301.

Third Edition (February 1998)

This edition applies to Version 1 Release 3 of IBM Firewall for AIX, Program Number 5801-AAR for use with AIX for RISC System/6000.

Comments may be addressed to:
IBM Corporation, International Technical Support Organization
Dept. HZ8 Building 678
P.O. Box 12195
Research Triangle Park, NC 27709-2195

Contents

Preface

This book was written to help you design and configure solutions using the IBM Firewall for AIX. It deals with the issues involved in connecting private IP networks to the Internet. It describes the different kinds of firewall technologies that can be used for such connections, focussing on the implementation of those functions in the IBM Firewall for AIX program product.

There are numerous configuration examples showing ways to set up the filtering, application server and gateway functions of the IBM Firewall for AIX.

This book will help you to establish a solid fence between your network and the outside world.

Some knowledge of TCP/IP protocols is assumed.

The Team That Wrote This Redbook

This redbook was produced by a team of specialists from around the world working at the Systems Management and Networking ITSO Center, Raleigh.

Ellis Luk is a Network Security Specialist at Advantra Pty. Limited in Australia. He holds a PhD degree in Electrical Engineering from the University of New South Wales. He has been working in the UNIX and TCP/IP area for over ten years, specialized in network security in last five years. His areas of expertise include security audit, firewalls and intrusion detection systems.

Klaus Majewski is a Services Specialist with IBM Finland. He has 2 years of experience in Firewall field. He holds a M.Sc. degree in Computer Science from Helsinki University of Technology. His areas of expertise include firewalls, DNS, Web servers, Internet services, network computers.

Douglas Raxworthy is a Managed Operations I/T Specialist from New Zealand. He has 5 years of experience in UNIX, specifically AIX, and 1 year of experience in firewalls. He has worked at IBM for 3 years. His areas of expertise include AIX Administration, Fault Determination and Resolution, AIX Connections and a little Tivoli for good measure.

Jorge Ferrari is a Senior ITSO Specialist in Network Design Tools at the Systems Management and Networking ITSO Center, Raleigh. He writes extensively and teaches IBM classes worldwide on all areas of network design tools. Before joining the ITSO in 1993, he worked in the SNAP/SHOT and Network Design group in Raleigh, NC.

This publication is the result of a residency conducted at the Systems Management and Networking ITSO Center, Raleigh.

Thanks to the following people for the invaluable advice and guidance provided in the production of this document:

Andrew Yeomans, IBM UK

Pascal Rütten, IBM Netherlands

Freddy Alves Vaquero, IBM Venezuela

Rob Macgregor, IBM UK

Valerie Aschman, Stacy Powers, Rex White, Jaime Claypool, Kevin Dunphy
from the Network Security Products Development, IBM RTP

Some of the material in this book was based on a previously published redbook: *Safe Surfing: How to Build a Secure World Wide Web Connection*, SG24-4564. We thank the author of that material:

Andreas Siegert
IBM Germany

Comments Welcome

Your comments are important to us!

We want our redbooks to be as helpful as possible. Please send us your comments about this or other redbooks in one of the following ways:

- Fax the evaluation form found in "ITSO Redbook Evaluation" on page 311 to the fax number shown on the form.
- Use the electronic evaluation form found on the Redbooks Web sites:

  For Internet users `http://www.redbooks.ibm.com`
  For IBM Intranet users `http://w3.itso.ibm.com`

- Send us a note at the following address:

  `redbook@vnet.ibm.com`

Chapter 1. Here There Be Dragons (An Introduction to Firewalls)

In the far-off days of old, maps of the world were simple affairs with the known world at the center, and a lot of blank space around the edge. Map makers inscribed warnings to would-be travellers: "Uncharted" and "Here There Be Dragons". But people *did* travel, to map what lay beyond their horizon and (most importantly) to try to become rich by finding precious cargoes.

In many ways the Internet of the past few years has been like the unknown world of those early maps. Although there are many people in the Internet world, few of them know this world really well and many see it as a source for future growth and revenue. There are dragons lurking here too; crackers who will try to disrupt and destroy. This book deals with some techniques for providing Internet access, while keeping the bad guys out of your own network.

1.1 Commercialization of the Internet

The Internet is a worldwide IP network which links many different organizations. The Internet is not a centralized organization but a collection of different networks from various sources, governmental, educational and commercial. There is a single administration for managing IP addresses and naming domains, the *Internet Assigned Numbers Authority*, but no other central administrative function. Internet routing is done by many Internet providers, government departments and private service companies who establish connections among themselves and build the base of the network. Organizations connected to the Internet are usually bound to one provider and so may communicate with any other connected organization across the inter-provider routes.

In the past, the Internet was suited to data processing and scientific people, using unfriendly interfaces such as Telnet or FTP. Today, everybody who knows how to use a keyboard and click a mouse may use a very friendly Web browser client to navigate the World Wide Web. The Web has had a profound effect on both the volume of traffic carried by the Internet and also on traffic patterns, since by following hypertext links, a user may rapidly establish connections with many different server machines. This means that:

1. There are many more sessions being set up, as a single client is likely to communicate with many more servers than in the past.
2. There are many more client machines, because the interface is familiar and easy to use and many organizations have rushed to provide Internet access to almost anyone.
3. There are many more servers, partly because many companies and organizations see the Internet as a way to disseminate information and partly because low cost service providers have allowed individuals and small organizations to provide their own Web sites.

So successful has the World Wide Web been, that to many people today it *is* the Internet.

The Internet today is metamorphosing from a vehicle for academic discussion and the sharing of research data, into a global information resource for private and commercial use.

Like the merchant-adventurers of old, many companies feel that the world of the Internet promises opportunities, in several forms:

- Inter-Company Communications

  The use of e-mail has radically changed the way that companies operate internally, by speeding communications, increasing the reach of corporate announcements and introducing a new egalitarianism (that is, people will send E-mail to managers and directors to whom they would not dream of telephoning or writing). Mail access via the Internet brings the same sort of benefits to inter-enterprise communications as well as a way for individuals to communicate.

- Marketing, Sales and Support

  The Internet offers a huge marketplace for promoting products and services to a targetted audience. Some companies have flooded the net in a crass and insensitive way, generating a lot of anger among the Internet community. However, as the Internet becomes increasingly commercialized and with the advent of new access methods such as the World Wide Web, product promotion and support has become an increasingly important portion of network traffic.

- Productivity and Competitive Edge

  No matter what subject you are interested in, you will find information about it *somewhere* in the Internet. Providing rapid access to information sources can give a major productivity boost. Also access to data allows employees to make more informed decisions and to better explore business opportunities.

- Conducting Business

  Until recently, business on the Internet was inhibited by the absence of good security standards. Several developments have addressed this problem. Probably the most far-reaching is the global availability of the Secure Sockets Layer (SSL) by Netscape Corporation. This allows Web transactions to take place with good privacy and authentication, using public-key encryption techniques. Many other security technologies have been widely used for some time, such as Pretty Good Privacy (PGP) for secure mail transfer, but unlike SSL they required some technical knowledge from the user. SSL is a generic solution to the problem of ensuring privacy, but it is not an answer to everything. For example, the Secure Electronic Transactions (SET) standard will enable secure credit card payments to become a reality. In parallel with these new technologies, there has been an easing of the U.S. cryptography export restrictions.

  It is clear that there will be a lot more commercial activity with security developments such as these that allow safe transfer of private data, credit card numbers, etc.

This commercial interest in the Internet is fuelled by and also a cause of the explosive growth in Internet hosts and traffic (see Figure 1 on page 3).

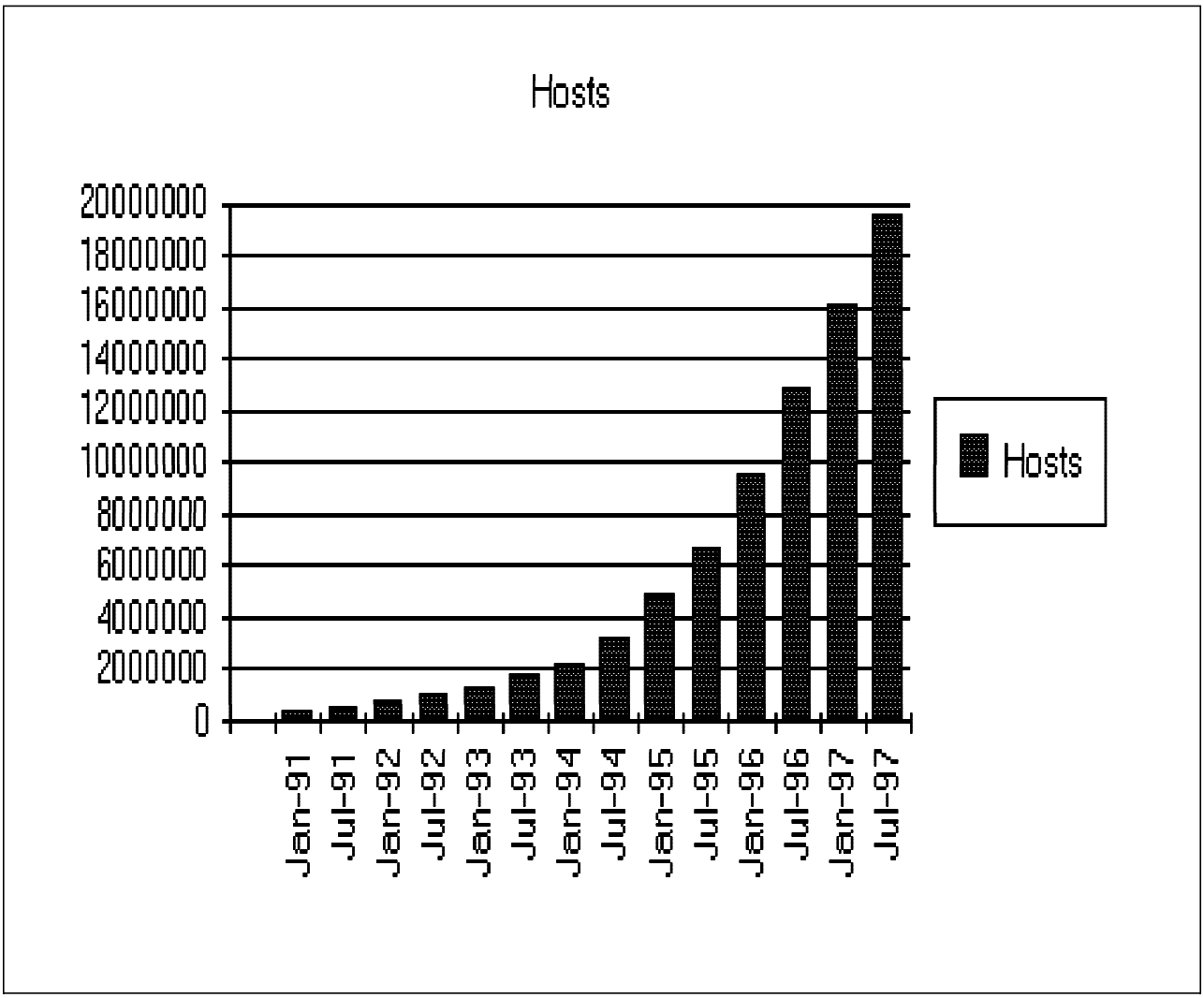


Figure 1. Internet Host Growth. This information is published by Network Wizards and is available on the Internet at http://www.nw.com/. Note that this is only the number of "visible" hosts. Many hosts are hidden by firewalls so the real figures are likely to be much greater. This number of hosts also translates into an even larger number of users, because each host may have multiple users.

So much for the opportunities; what about those dragons? There has been much press comment about crackers on the Internet, some of which has over-sensationalized their exploits. The threat that crackers have posed to date is less severe than the popular image. Most of the publicized incidents have been perpetrated as a means to grab attention and not with any real malicious intent. However, the ingenuity of the crackers in defeating system defenses has *not* been overstated. The really frightening prospect, therefore, is if one of these wily crackers has some personal or commercial reason to do you harm.

1.2 What Does "Security" Mean?

There is always a trade-off to be made between making a computer secure and the function it can provide. In the extreme case, the most secure computer is one that is turned off. With networked computers the problem is compounded since the communication channel itself is open to attack. We can characterize attacks in two ways:

1. Passive attacks: Tapping or tracing the communications. These are very difficult to detect. You should assume that someone is eavesdropping on every communication you send across the Internet.
2. Active attacks: That is, a cracker is trying to take over your machines. Even if you are certain that your own machines have not been compromised, you cannot be certain about the machines at the other end of the connection. Realistically you have to extend your circle of trust to some of those machines, or else you will not use the Internet at all.

It seems that once you start getting paranoid about computer security you can reach a point where nothing seems safe anymore. Is this justifiable? After all, we don't (normally) worry about people tapping our telephone conversations or reading our mail, and we happily send credit card numbers, private messages, gossip and scandal using those media. The difference with the Internet is that the carrier is not a regulated, well-defined entity. In fact you have no idea whose computers your message passes through on the way to its destination.

1.3 Reducing Your Exposure

This book concentrates on protecting the points of entry, where the Internet meets your private network. However, we will also consider authentication and encryption of data passing across the Internet, using secure tunnels and alternatives for secure terminal emulation.

The any-to-any connectivity of the Internet can, alone, give you many security problems. You will need to protect your own private data and also protect access to the machines inside your private network against abusive external use.

The first step to achieving this is to limit the number of points at which the private network is connected to the Internet. The best configuration is where the private network is connected by just one gateway. If you only have this unique path, you have gained a *choke point*, where you can exercise control over which traffic to allow into and out of the Internet. We call this gateway a firewall. When considering the details of your firewall it is important to keep in mind that the whole concept relies on the fact that it is the only gateway. If an attacker can find an alternative, uncontrolled, access point your firewall effort has been wasted. Every point of entry must be subjected to the same rigorous standards of security.

Depending on what your security requirements are and how much money you have, you can develop different firewall strategies. The most important advice is to keep the strategy simple.

1.4 The Firewall Concept

To understand how a firewall works, consider this example: Imagine a building where you want to restrict access and to control people who enter in. You define in the architecture of the building a single lobby as the only entrance point. In this lobby, you have some receptionists to welcome, some security guards to watch over, some video cameras to record, and some badge readers to authenticate people who enter the building.

This works very well to control a private building. But imagine that a non-authorized person succeeds in entering, no matter how. To protect the building against any actions from this person is more difficult. However, if you supervise his or her movements you at least have a chance to detect any suspicious behavior and repair any damage.

When you are defining your firewall strategy, you may think it is sufficient to prohibit everything that presents a risk for the organization and allow the rest. However, because of new attack methods, you may not be able to prevent every attack and, as in the case of the building, you need to monitor for signs that somehow your defences have been breached. Generally, it is much more damaging and costly to recover from a break-in than to prevent it in the first place.

In the case of the firewall, the classic solution uses a screening router. Although that is not sufficient today to ensure security, it is still the starting point for firewall defenses. A better strategy is to permit only the applications you have tested and have confidence in. If you follow this strategy, you have to exhaustively define the list of services you must run on your firewall. Each service is characterized by the direction of the connection (from in to out, or out to in), the list of users authorized, the list of machines where a connection can be issued, and perhaps the range of time of day you authorize this service.

If you want to refresh your knowledge about TCP/IP

In the following sections we assume a fair understanding of IP protocols, addressing, and router configurations. If you want to increase your understanding of TCP/IP, we recommend you read *TCP/IP Tutorial and Technical Overview*, GG24-3376. Also, if you encounter words or phrases that you do not understand, you may find an explanation in the glossary.

1.4.1 Screening Filter

The first and most commonly used strategy is to separate the private IP network from the Internet by inserting a router between them, as shown in Figure 2.

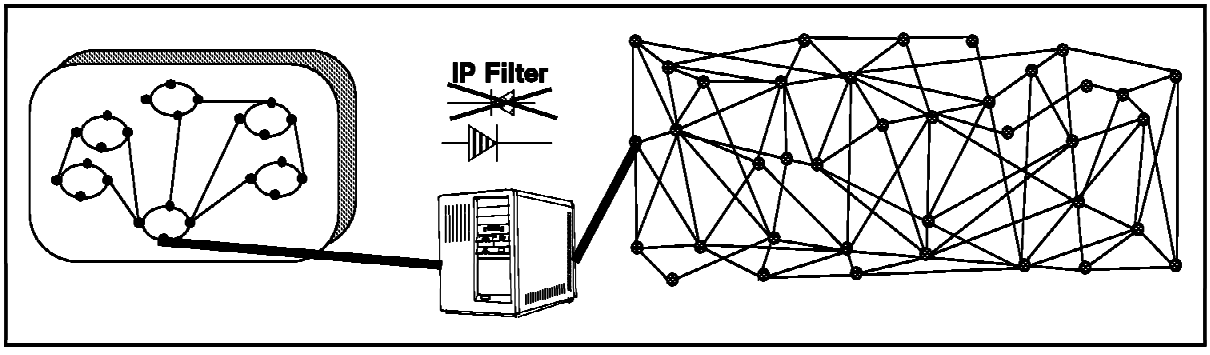


Figure 2. Screening Filter

This router filters all IP packets passing through and is called a screening filter. This way you can prevent access to machines or to ports in the private network and also do the reverse; prevent an inside machine from accessing the Internet. But if you do this, there is no way to control what's happening at the application layer. That is, you may want to allow one type of traffic across the gateway but

not another. You could manage this at the application host itself, but the more machines on which you have to impose controls, the less control you have. Nonetheless a screening filter is a very useful tool to use in conjunction with other tools as a security building block.

1.4.2 Bastion

A bastion is a machine placed between the secure and nonsecure network where the IP forwarding is broken, which means no IP packet can go through this machine. As the routing is broken, the only place from which you can access both networks is the bastion itself. Therefore, only users who have an account on the bastion, with a double identification (one for the bastion and one for the remote host), can use services on both the networks, as shown in Figure 3.

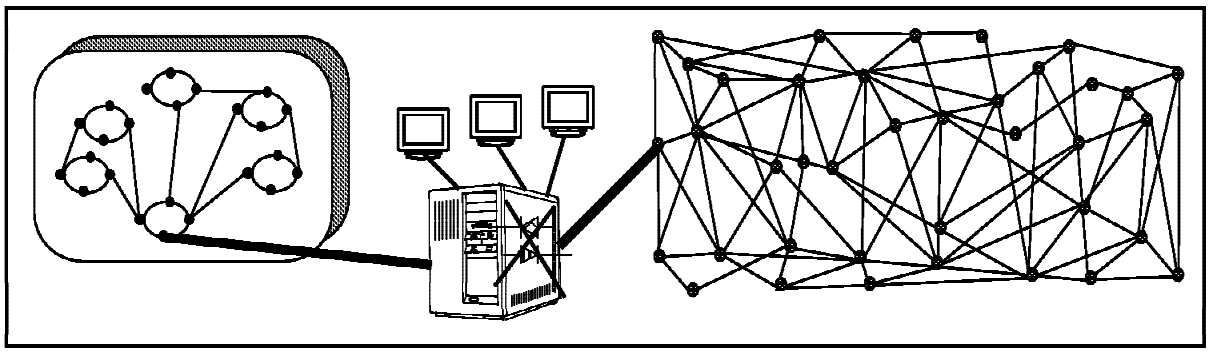


Figure 3. Bastion

This has some disadvantages, because the bastion may have to support many users. It is important to enforce good password control here since if a cracker manages to break into a user ID he or she can then impersonate the user and get into the private network. Besides this security point, supporting a great number of users will require a big machine. To avoid having users logged in to this machine and to reduce load on the machine, more general-purpose bastion applications exist, such as the SOCKS server (see 2.1.3, “SOCKS Server” on page 15).

1.4.3 Dual-Homed Gateway

One good solution is to combine a screening filter and a bastion, as shown in Figure 4 on page 7.

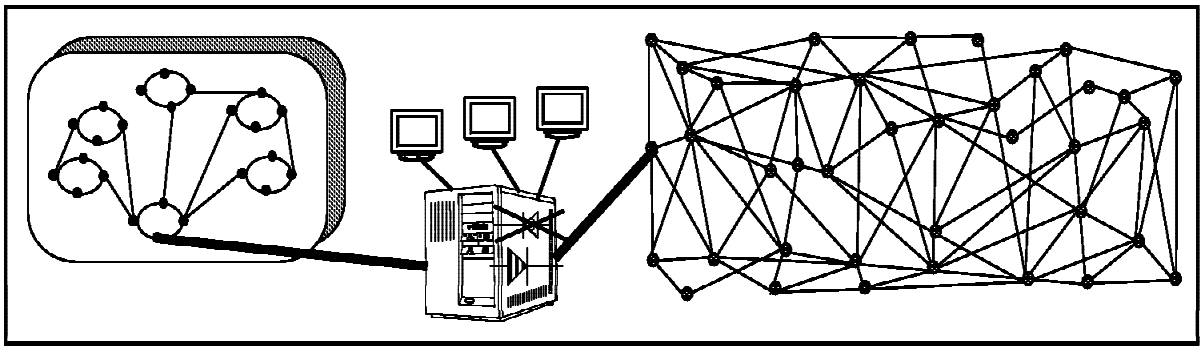


Figure 4. Dual-Homed Gateway

In this case, you can protect the dual-homed gateway from external attacks with filtering. For example, if you forbid external access to the telnet daemon, you reduce the threat of an external attack. If you have some nomadic machines that are hosted outside but need to connect to hosts inside the private network, you can limit the exposure by using a proxy server and perhaps using smart card authentication techniques.

The problem with this configuration is that the firewall machine can become very complex, so if a cracker does break into it, it may take some time to track him or her down. For example, there are a great number of IP ports used by so-called well-known services. Some of these are, in fact, not well-known at all and crackers regularly use them as a back door into a computer. So it is important to block as many such ports as you can, without impacting the services that you do want to work.

1.4.3.1 Bastion Behind a Screening Filter

A better solution is to use the same solution as above but use two machines as shown in Figure 5.

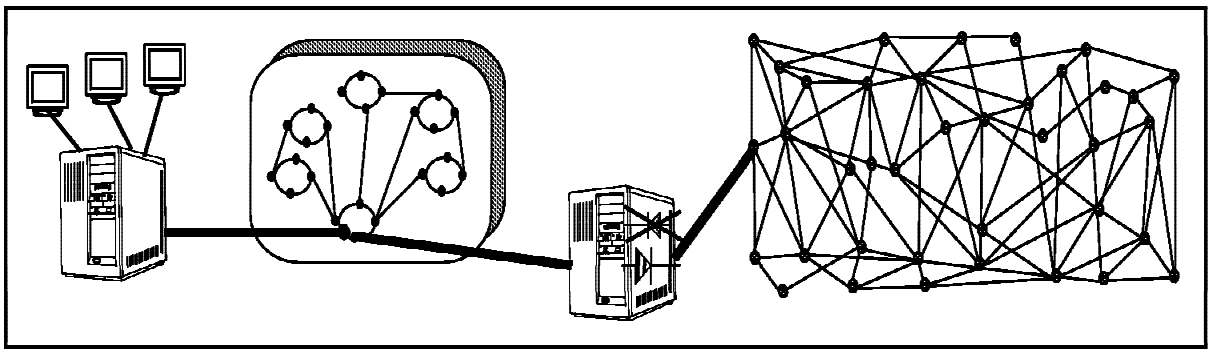


Figure 5. Bastion Behind a Screening Filter

In this configuration, the bastion is protected from external attack by the screening filter. This one is a very simple router without a daemon inside. This implies it is very hard to break into this machine. In this manner, you could hide the structure of your private organization, and protect a complex daemon such as sendmail.

1.4.3.2 Screened Subnet

A further development of this is to use the subnetwork between the screening filter and the bastion as a site for application services. This is increasingly common, as organizations want to provide machines that are widely available (such as Web servers) but still have strong protection for their private network. The screening filter provides some protection for the service machines, without unduly limiting access. A possible example of this is shown in Figure 6. This network is composed of two screening filters and one or several bastions. When you start considering this sort of solution, the cost becomes a major factor since, for reasons of integrity, each component in the design should ideally be a dedicated machine.

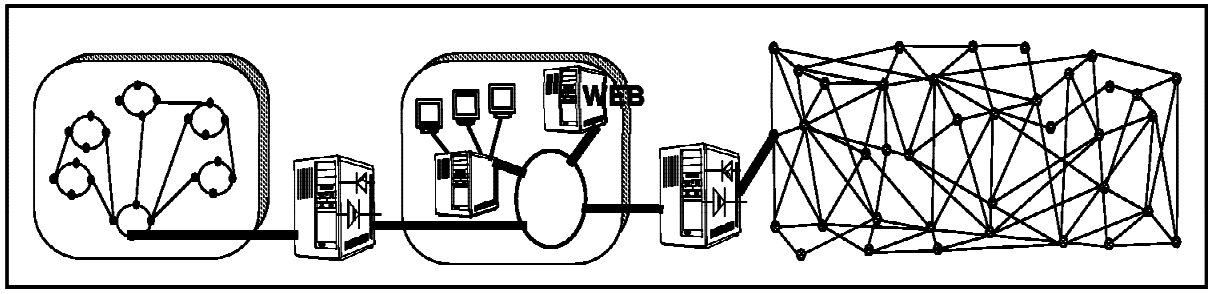


Figure 6. Screened Subnet

The architecture is very simple and each machine on this subnet performs only simple task(s), depending on the number of bastions you have. It is often referred to as a *demilitarized zone* (DMZ).

Cheaper and another common solution to provide DMZ is to use three adapters on the firewall. DMZ is still in the area where secure traffic never flows and we can limit activities in the DMZ with the firewall. Figure 7 shows you the idea.

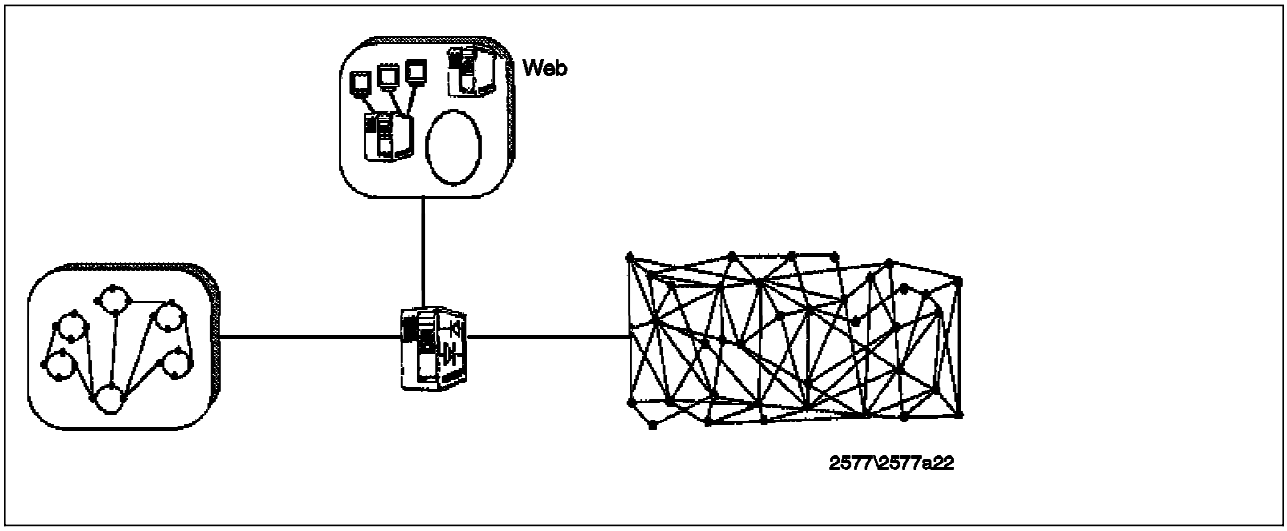


Figure 7. Screened Subnet Implemented with Three Adapter Firewall

1.5 Firewall Objectives and Firewall Rules

We have seen that there are a number of ways to configure a firewall, depending on the size of your organization and what you are trying to achieve. Before we look in detail at how to use the IBM Firewall 3.1 to achieve some of these configurations, let us state some high-level objectives and rules.

There are a number of objectives that are common to all firewall cases:

- You want to only allow traffic to flow that you have determined is safe and in your interest.
- You want to give away a minimum of information about your private network.
- You want to be able to keep track of firewall activity and be notified of suspicious behavior.

These common objectives translate into a number of rules that you should always keep in mind:

- *Anything that is not explicitly permitted should, by default, be denied*.

  What this means is that when you set up your firewall you should be able to state exactly what traffic you want to pass through it. It should not be possible for any other traffic to pass.

- *You should keep outside users out of your internal network wherever possible*.

  Even if you are providing a legitimate service for outsiders to use, you should not trust them. If possible such services should be placed outside the firewall (possibly within a DMZ), isolated from your internal systems.

- *You should do thorough auditing and logging*.

  You should assume the worst, that at some time your systems will be compromised by a cracker. At this point you need good logging functions to allow you to detect the cracker, retrace his movements, and prevent further damage.

1.5.1 Beyond the Firewall: Filtering Content

Firewall technology must wage a continuous battle against the ingenuity of the cracker. This means continuing to be vigilant in preventing misuse of legitimate IP network services and new security holes.

One area that has become part of this battlefield in recent times is the potential for exploitation of "smart" client function. As the Web browser becomes the complete do-everything client, Web-based applications start to rely on client side execution of programs, using techniques such as Java, ActiveX controls and other plug-in function.

1.6 Firewall and High Availability

Firewall is a very critical component of the network and it should be operational 24 hours per day. IBM Firewall 3.1 can be combined together with another IBM product called High Availability Cluster Multi-Processing (HACMP). Together they will provide a high availability firewall. The basic idea behind high availability is to have two or more machines providing the same kind of services. All the machines are sending heartbeat signals to each other. When one

machine stops sending the heartbeat signal, the other machines know it is in trouble and take over its services. The failing machine will shut down itself. Look at Appendix E, “IBM Firewall Related Products” on page 285 for more information about HACMP.

Load balancing between firewalls can be handled with an IBM product called Interactive Network Dispatcher. The Network Dispatcher has many different algorithms to divide the network traffic between several firewalls. It will also provide some level of high availability. When one of the firewalls is down, the Network Dispatcher will guide the network traffic to available firewalls. Look at Appendix E, “IBM Firewall Related Products” on page 285 for more information about Network Dispatcher.

Chapter 2. Introducing IBM Firewall 3.1

Firewalls have existed since the 1980's and continue to evolve to allow both better user function and increased security for private networks. The IBM Firewall for AIX was first released on December 1994. It was based on IBM technology that has been actively securing IBM assets since 1987. Subsequently, IBM Firewall has been updated to meet the increasing concern of inter-network security, and on July 1997, the IBM Firewall Version 3.1 was released.

The IBM Firewall has changed names through the years; a brief history of these changes would be:

- 1987: Internal release
- December 1994: 5765-473 IBM NetSP Secured Network Gateway V1.2
- October 1995: 5765-626 IBM Internet Connection Secured Network Gateway V2.1
- July 1996: 5756-626 IBM Internet Connection Secured Network Gateway V2.2
- June 1997 5765-C16 IBM Firewall V3.1 for AIX

You can think of IBM Firewall as a tool box you use to implement the functions of different firewall architectures, both screening filter and bastion. Once you choose your architecture and your security strategy, you can build the practical implementation using the tools it provides.

IBM Firewall for AIX is built on standard system hardware and a standard commercial operating system. Although it starts out as a standard base system, by the time the firewall code is installed it has been hardened to such an extent that it becomes a fortress on the network.

There are two user interfaces for configuring IBM Firewall on AIX:

- A graphical user interface (GUI) which is a Java application invoked from a Web browser.
- A set of menus under the AIX systems management interface tool (SMIT).

We recommend that you use the GUI for firewall configuration because it provides a better visualization of the firewall setup. This will reduce the chance of misconfiguration. In general we will use the GUI for the examples in this book.

IBM Firewall provides comprehensive logging of all significant events, such as administrative changes and attempts to breach filter rules. The logging method used by IBM Firewall is the UNIX syslog daemon, so it is very easy to process event messages (you will find examples of this in Chapter 14, “Logging” on page 251).

2.1 IBM Firewall Components

Because the firewall is, at heart, an IP gateway, it divides the world into two or more networks, composed of one or more nonsecure networks and one or more secure networks. The nonsecure network is, for instance, the Internet. The secure networks are usually your corporate IP networks. The tools that the product provides for you are:

- IP filters
- Proxy servers
- SOCKS server
- Specific services such as domain name service and mail handling
- A secure IP tunnel

2.1.1 IP Filters

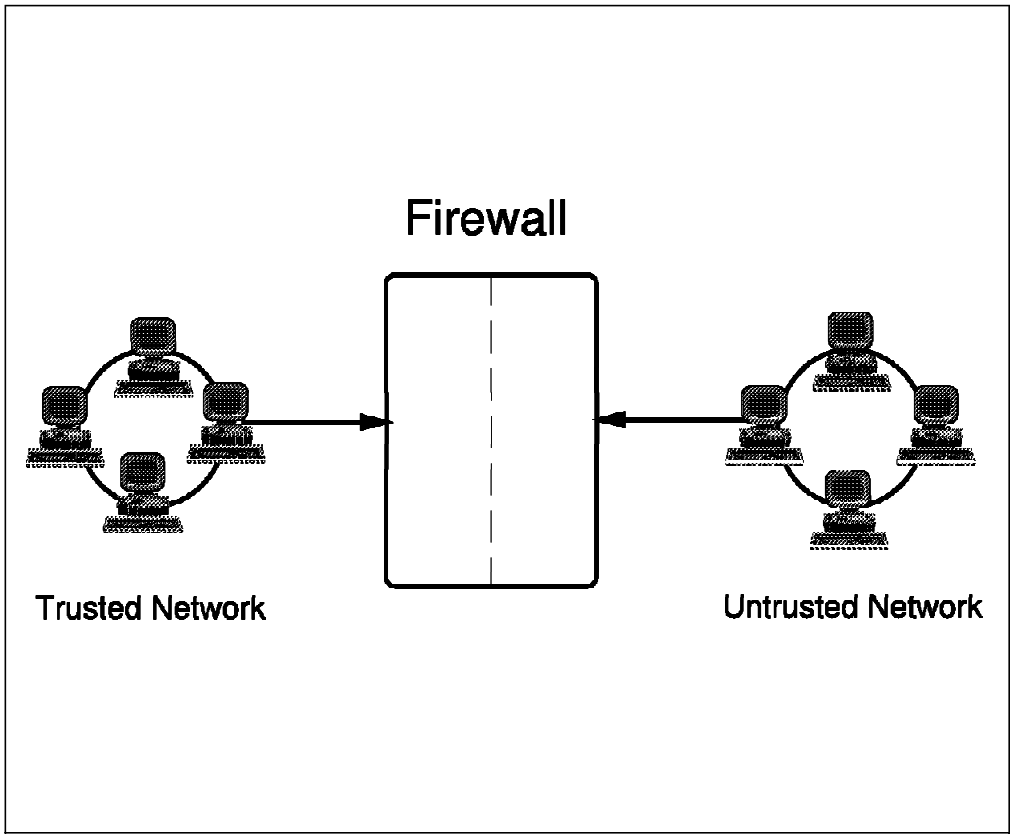


Figure 8. Firewall with IP Filtering

IP filters are tools to filter packets at the session level, based on IP address, direction, and packet type (TCP or UDP ports, or ICMP ID). The filter rules work with the IP gateway function, so the machine is required to have two or more network interfaces, each in a separate IP network or subnetwork. At least one adapter is declared nonsecure and the other(s) declared secure. The filters act at *all* of these adapters. That is to say, the filters do not only control what can flow from the secure side to the nonsecure side of the gateway but also what can flow in and out of each adapter individually. This is an important distinction, because it means that filters are an essential part of any firewall configuration, even if all traffic through the firewall is handled by a proxy application instead of

passing straight through. As we said in 1.5, “Firewall Objectives and Firewall Rules” on page 9, our prime design objective is to give away as little useful information as possible about our network without impacting the legitimate traffic. IP filters are a key component of this.

2.1.1.1 Implementation

In IBM Firewall you do not normally directly select the IP filters that you want. Instead, you define the connections that you want the firewall to support and the configuration dialog generates the appropriate filters. The result is a series of entries in the /etc/security/fwfilters.cfg configuration file where each line defines a filter using the following criteria:

- Source IP address and mask
- Destination IP address and mask
- Type of IP protocol
- Source and destination service addresses (for example, TCP/UDP port numbers)
- Direction of the IP packet
- Network interface
- Whether routed through the firewall or not
- Whether the IP packet is fragmented or not

These criteria are described in detail in 3.1, “Fundamentals of IP Packets” on page 23.

Whenever a packet arrives at the IBM Firewall it is compared with each line in the filter file in turn, until one is found that matches it. At that point the processing stops and the filter directive (*permit* or *deny*) is applied.

2.1.1.2 Objective of the Filters

IP filtering provides the basic protection mechanism for the firewall, allowing you to determine what traffic passes across it based on IP session details, thereby protecting the secure network from outsiders that use unsophisticated techniques (such as scanning for secure servers) or even the most sophisticated techniques (such as IP address spoofing). You should think of the filtering facility as the base on which the other tools are constructed. It provides the infrastructure in which they operate and denies access to all but the determined cracker.

2.1.2 Proxy Servers

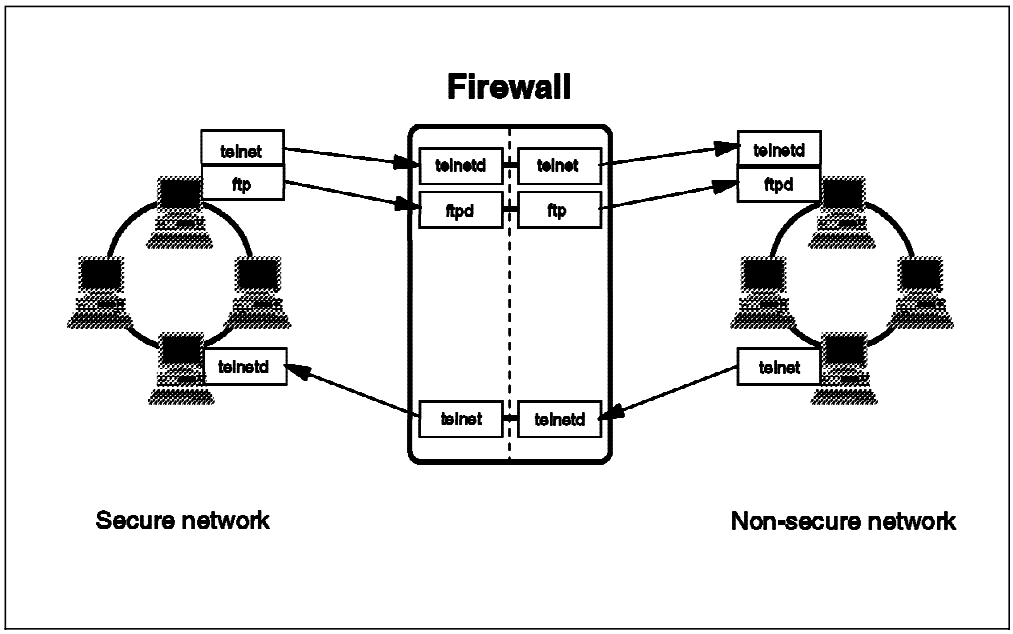


Figure 9. Firewall with Proxy Server

Proxy servers provide network security at the application level (that is why they are called application-level gateways as well). For a user to make connections through a proxy server firewall, the user is required to connect to the proxy first, and then request the proxy to connect to the destination. Since connections are terminated at the firewall, the proxy server can examine the data of the connection before sending the data to the destination. In this way, the proxy server can provide access control based on the application data, such as authenticated user name. The difference between proxy server and IP filter is that connections through proxy servers do not require/involve packet routing at all.

For example, users use FTP client program to access Internet sites. They have to connect and authenticate (using a password) to the proxy. Having successfully accessed the proxy (and thus authenticated themselves) they now have to again issue the command to reach the desired machine on the Internet.

This method is usable also from the nonsecure network to the secure network, but this raises other important security problems. If you use the Internet to connect to the bastion, you have to enter your login name and password to be identified. But, as we have said, you can't know what machines your session may pass through and some cracker may be looking for login names and passwords, by wire tapping or using IP trace commands. If they catch your ID, they may impersonate you and thus get into the organization with your identity. (Of course, this would only get them to the firewall; they would need to trace the second command to get past it.) In this scenario, with the proxy server, you can use a more sophisticated tool of authentication, such as a security identity card. This mechanism generates a unique key which is not reusable for another connection. Two security identification cards are supported by IBM Firewall: the SecureNet card from Axent and the SecureID card from Security Dynamics.

Even sessions established using such strong authentication techniques can be attacked using hijacking, so the only totally secure way to allow incoming Telnet connections is to encrypt the session between the end user and the firewall, as

explained in 6.22, “Secure Terminal Emulation” on page 133 and Chapter 7, “Secure IP Tunnel” on page 137.

2.1.2.1 Implementation

In IBM Firewall 3.1, the proxy services available are Telnet, FTP and HTTP. The Telnet and FTP proxy servers, by default, require users to authenticate themselves at the firewall. Users who access the gateway in this way get a very restricted operating environment. It allows you to establish sessions onward into the nonsecure network, using commands such as telnet. They do *not* allow you to issue any commands to look at or modify the firewall itself, for example you cannot do `ls`, `cat`, `echo`, etc.

However, Telnet and FTP proxies can be operated in the *transparent* mode. Under this mode, users can use the proxies but do not need to be authenticated at the firewall, so long as their connections are originated from the secure network. This reduces the proxy user administration on the firewall, and provides a more user-friendly interface to firewall users.

The HTTP proxy provides a more efficient way to handle the HTTP protocol and provides better logging information (such as URL) for firewall administration. It should be noted that the HTTP proxy does not support user authentication. Therefore we recommend that you use the IP filter to allow access from the secure network only.

2.1.2.2 Objectives of Proxy Applications

When you connect via a proxy server, the TCP/IP connections are broken at the firewall, so the scope for compromising the secure network are reduced. Compared with IP filtering, proxy servers can provide more comprehensive logging based on the application data of the connections. For example, the name of the files transferred through the FTP proxy is logged. When using Telnet and FTP, users have to authenticate themselves if they are connected from the nonsecure network. As long as their password remains secure, the firewall itself is safe.

Once connected, the appearance and the behavior are unchanged. But, this method needs to have a double connection: one from the client machine and one from the firewall machine. This is time consuming and will have an impact on performance.

One of the major advantages of the proxy approach is that you do not need a special version of the client program on the client machine. Therefore, once you have installed your firewall, every user recorded in the firewall can have access to the nonsecure network without any additional software installation.

2.1.3 SOCKS Server

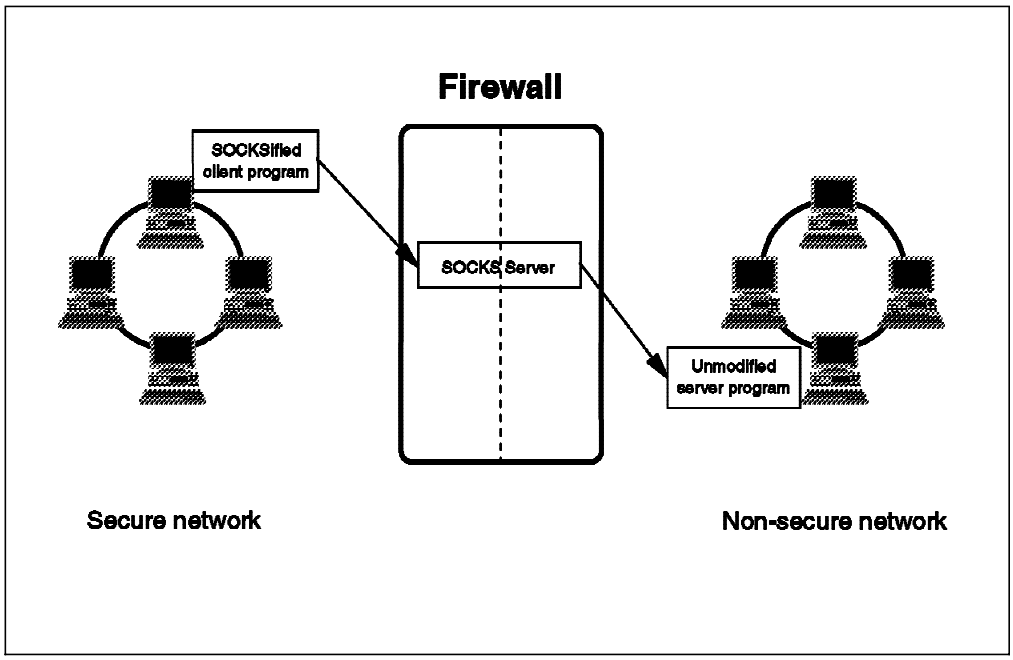


Figure 10. Firewall with SOCKS Application-Layer Gateway

SOCKS is a standard for circuit-level gateways. It does not require the overhead of a more conventional proxy server where a user has to consciously connect to the firewall first before requesting the second connection to the destination.

The SOCKS server results in a similar bastion configuration, since the session is broken at the firewall. The user starts a client application with the destination server IP address. Instead of directly starting a session with the destination server, the client initiates a session to the SOCKS server on the IBM Firewall host. The SOCKS server then validates that the source address and user ID are permitted to establish onward connection into the nonsecure network, and then creates the second session.

SOCKS needs to have new versions of the client code (called SOCKSified clients) and a separate set of configuration profiles on the firewall. However, the server machine does not need modification; indeed it is unaware that the session is being relayed by the SOCKS server. For a comprehensive list of socksified clients see 8.11, “Using SOCKS Services” on page 195 and E.1, “Aventail AutoSOCKS” on page 288.

2.1.3.1 Implementation

The SOCKS daemon uses a configuration file, /etc/sockd.conf, to allow or deny connections through the bastion.

Each line of this file defines a rule that controls access through the firewall. The following information is present:

- User ID or list of user IDs
- Address of client and mask
- Address of remote server and mask
- Operation field and port to define what service or what range of services
- Command to execute (very useful to log or to alert)

Authentication of the user ID is optional, but if you choose to use it, the SOCKS server calls the identd server on the client machine. The ident server answers with the identity of the calling user. This procedure is described in more detail in 8.8, “Using the SOCKS Server” on page 190. Once a connection is established, the SOCKS server acts as an *application-level router* appearing as a client to the real application server and as a server to the client.

2.1.3.2 Objectives of the SOCKS Server

For outbound sessions (that is, from a secure client to a nonsecure server) SOCKS has the same objectives as a proxy application server, that is to break the session at the firewall and provide a secure door for access control. It has the advantage of simplicity for the user, at a cost of a little extra administrative work. SOCKS server is not intended to handle inbound sessions, since it does not provide for secure password delivery, and the identd user ID checking could possibly be subverted by a cracker. SOCKS server is also, arguably, likely to be more secure than a proxy server, since it is a relatively small piece of code written solely with security in mind.

2.1.4 Domain Name Service

Access to the domain name records for the secure network is of great assistance to crackers, since it gives them a list of hosts to attack. A subverted DNS server can also provide an access route for a cracker; see 16.1.1.8, “8. The Inverse DNS Tree Can Be Used for Name-Spoofing” on page 264.

From the outside view, the name server on the firewall only knows itself and never gives information on naming inside the private IP network. From the inside view, this name server knows the Internet network and is very useful for accessing any machine on the Internet by its name

2.1.4.1 Implementation

Configuration is performed through either the GUI or SMIT. You have to specify the following:

- Nonsecure domain name
- Secure domain name
- Nonsecure name server
- Secure name server

This generates all the necessary configuration files for the firewall domain name server. It is still necessary to configure the name server in the secure network to use the firewall service, however. We show some examples of this in Chapter 11, “Domain Name Service” on page 215.

The operation of DNS on the firewall relies on three features:

1. The *forwarders* function, so that the name server inside the secure network can receive information about hosts outside its domain from the firewall name server, but the reverse cannot happen.
2. The caching capability which allows the firewall name server to get name information from the nonsecure network without pre-definition.
3. The fact that name resolution requests can be directed to any name server, whether or not the host from which the request is coming is a name server itself. This allows the firewall to be able to resolve names inside the secure

network, without giving those names away to hosts in the nonsecure network.

Figure 11 shows how name resolution requests flow for different combinations of requester and node.

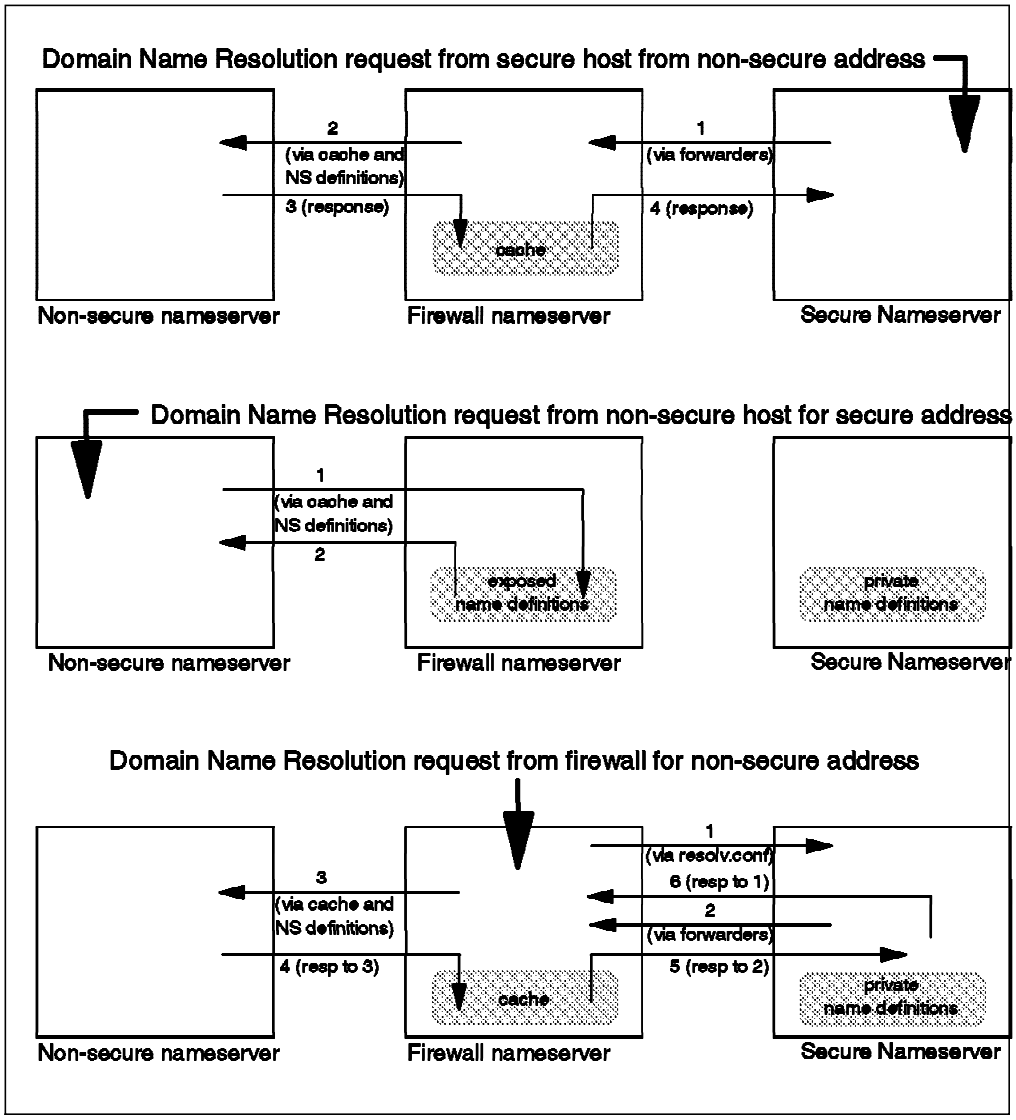


Figure 11. Name Resolution Flows

Notice that only the names and addresses that we want to reveal (and have defined in the firewall name server) are available to an external host. Notice also that name requests originating on the firewall itself are treated as those from any secure network host, since /etc/resolv.conf refers to the secure network name server.

2.1.4.2 Objectives of the DNS Server

Running the DNS server on the firewall has the dual advantage of preventing name resolution requests flowing across the gateway and hiding secure network hosts from the nonsecure world.

2.1.5 Secure Mail Handling

Mail is one of the primary reasons why an organization would want to access the Internet. IP mail is transmitted via the SMTP protocol, which is a simple client/server architecture allowing store and forward or direct delivery.

Normally, you want to have relatively free access in and out of the secure network for mail traffic. Unfortunately, however, the standard mail daemon, sendmail, has proved to be a fertile breeding ground for bugs over the years, many of which have been exploited by crackers. In previous versions of IBM Firewall, the AIX sendmail daemon is used as a mail relay, but this sendmail daemon has been replaced. Now the IBM Firewall supports its own secure mail gateway called SafeMail. It can be configured to forward mail to a mail gateway within the secure network.

2.1.5.1 Implementation

The implementation assumes that there is a secure mail gateway within the secure network. The main reason why you would have such a gateway is for reasons of control, for example:

- A single point to handle misrouted files
- A common definition for all your distributed nodes
- A logical point to perform transformations to other mail formats
- A single point to handle aliasing.

  For example, you may want your users to have personal mail handlers that are not related to the machines to which they log in.

You have an option in the system administration folder, the secure mail server, to configure the IBM Firewall secure mail gateway. If you do so, the firewall mail relay is automatically started, forwarding traffic between the gateway and the outside world.

From the point of view of hosts in the nonsecure network, the only mail gateway they see is the firewall. This is advertised by the addition of an MX record to the firewall name server definition. Hosts inside the secure network can be defined either to route internet traffic to the firewall directly, or to route to the internal mail gateway, which can then forward mail to the firewall. Some examples of SafeMail are shown in Chapter 12, “Mail Handling” on page 231.

2.1.5.2 Objectives of the Mail Relay

As for the proxy, SOCKS and DNS servers, this relay means direct sessions do not have to be carried across the firewall gateway. It also hides the internal mail gateway from the nonsecure network. Only the firewall secure mail server is advertised outside the secure network, which is much more resistant to attack than the real mail gateway.

2.1.6 Secure IP Tunnel

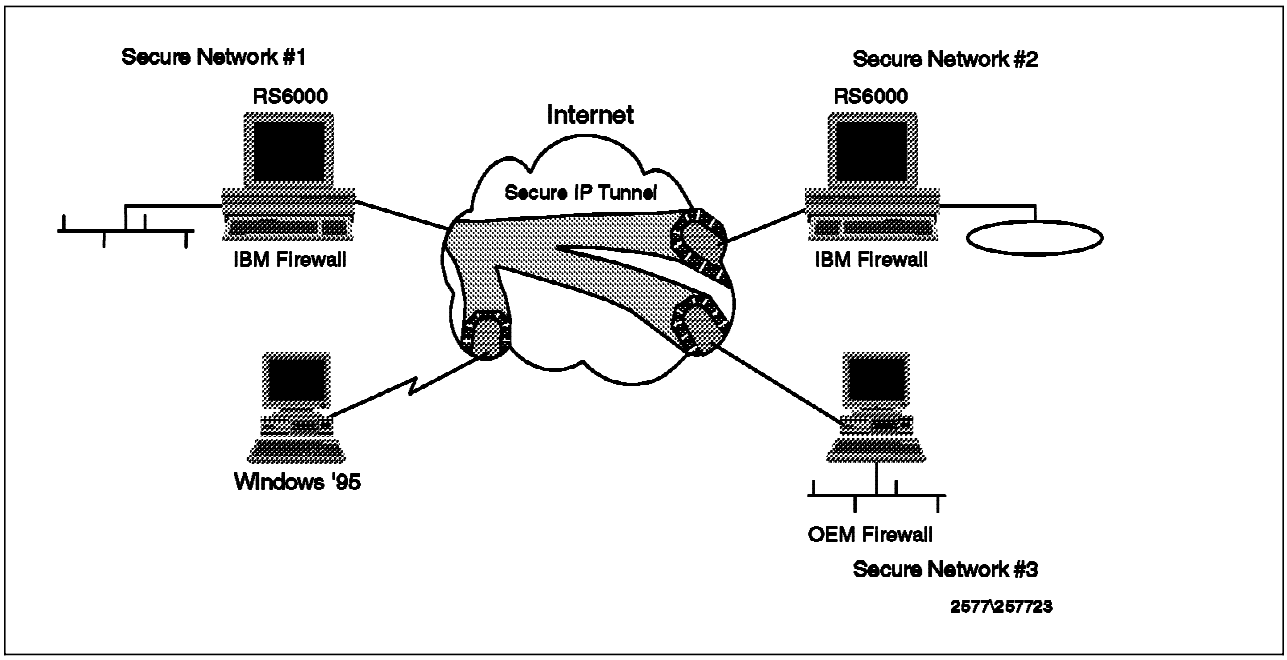


Figure 12. Secure IP Tunnel between Firewall and Tunnel Partner

The secure IP tunnel is a mechanism provided by IBM Firewall that permits a private communications channel to be created between the secure network and an external trusted host over an intervening public network such as the Internet. The external trusted host can be another firewall (IBM or other vendor), a Windows 95 PC or an AIX machine.

The firewall and the external trusted host establish a connection between them and they encrypt and authenticate traffic passing between the private networks.

2.1.6.1 Implementation

In order to configure the secure IP tunnel, you will have to add tunnel definitions and specific filtering rules to tell the IBM Firewall how to encrypt/authenticate traffic.

Chapter 7, “Secure IP Tunnel” on page 137 shows how secure IP tunnel operates between IBM Firewall and different external trusted hosts (for example, Windows 95 secure remote client and AIX IPSec client).

2.1.6.2 Objectives of the Secure IP Tunnel

Outsiders can eavesdrop on all traffic that goes through the public network. The secure IP tunnel allows you to obscure the real data being sent between the IBM Firewall and the external trusted host. It also allows you to be assured of the identity of the session partners and the authenticity of the messages.

2.1.7 Combining the Tools

We have now briefly introduced the tools provided by the IBM Firewall 3.1. In reality, you may not need to use all of the tools available, but it is likely that you *will* need to use tools in combination. For example, most of the server functions need to be protected by a set of IP filters to prevent them from being bypassed.

2.2 Tier Pricing

The IBM Firewall 3.1 is offered in three different options:

- Up to 50 users
- Up to 250 users
- Unlimited users

Of course, each option has a different price; it is called tier pricing.

The IBM Firewall 3.1 determines the number of users by tracking all the packets from the secure network, and registering the IP addresses of the secure hosts in a table, until the number of different IP addresses reaches the limit. After that, the IBM Firewall logs all packets from the secure network that have new IP addresses (not in the table), but those packets are not processed.

As you can see, only the IP addresses of the secure network count for establishing the number of users, not the IP addresses of the nonsecure side.

The entries in the table are not deleted. This means that if you bought a 50 users license, only the first 50 secure IP addresses that send packets to or through the firewall will be registered in the table. This table is deleted only at reboot. The only way to modify the limit of users is to upgrade the firewall software to a different tier.

Installing a new version is disruptive; the user must uninstall the existing firewall version and then install the new version. The existing configuration files can be saved for the new installation (i.e., using installp -ug FW will leave all the existing configuration files intact).

Chapter 3. Getting to Grips with IP Packets

As we have described, a firewall needs to intercept every IP packet that it receives and then process it according to the policy that you define. In this chapter we discuss the packet characteristics that the firewall can use to make processing decisions.

3.1 Fundamentals of IP Packets

The IBM Firewall machine operates between two or more IP networks, sometimes acting as a router (passing packets between the secure and nonsecure sides), sometimes providing a proxy server function to break the session in two. IBM Firewall uses IP filters to control which packets are passed and which are blocked on each side.

The information it uses to decide whether to block or pass a packet is largely contained in the packet headers. Some of the filtering criteria are:

- The source and destination IP address
- The direction of flow
- The IP protocol (ICMP, TCP, UDP or other protocols)
- The interface where the packet is detected (secure or nonsecure)

Before we show detailed examples of how to set up the firewall controls, we will consider the characteristics of the different IP protocols to gain a better understanding of what the firewall looks for.

3.1.1 An Introduction to IP Packets

Normally, a firewall does not look at the content of a packet, but only at the header. An IP packet consists of a formatted header, followed by the packet payload. The payload itself may include a further header containing session-level protocol information (for example, a TCP or UDP header).

Figure 13 shows the format of the IP packet header.

<table>
<tr><td>Version</td><td>Length</td><td>Type of Service</td><td colspan="2">Total Length</td></tr>
<tr><td colspan="3">Identification</td><td>Flags</td><td>Fragment Offset</td></tr>
<tr><td>TTL</td><td colspan="2">Protocol</td><td colspan="2">Header Checksum</td></tr>
<tr><td colspan="5">Source IP Address</td></tr>
<tr><td colspan="5">Destination IP Address</td></tr>
<tr><td colspan="3">Options</td><td colspan="2"></td></tr>
</table>

Figure 13. IP Packet Header Format

The following are the important fields in the IP header:

- The source address

- The destination address
- The fragmentation indicator (in the flag field)
- The protocol ID

The source address, destination address and protocol ID are used by the firewall filters to define which machines can access which particular service.

The fragmentation indicator is used to instruct IBM Firewall on how to handle fragmented packets. Different types of networks support different maximum packet sizes, so sometimes a router has to break a packet into smaller fragments to pass it from one network to another. The packet-filtering firewalls have to be aware of packet fragmentation because only the first fragment contains the header information of higher-layer protocols, such as UDP and TCP. Later fragments in a packet could override header fields, such as the source and destination port.

The IP specifications allow packets of very small sizes. The minimum packet size that can be sent according to RFC 791 is 68 octets. The problem here is that this packet size is not enough to carry the complete information for upper layer protocols. This leads to an attack technique called the *tiny fragment attack*.

The reassembly algorithm contains a mechanism by which later fragments can overwrite the data portions of previous fragments. An attacker could create a series of packets in which the first fragment will be allowed by the filter, but later fragments will overwrite relevant information (such as TCP source and destination ports). In this way the filtering rules can be bypassed if you allow fragmented packets. This is called the *overlapping fragment attack*.

Ideally you should configure your firewall to only support nonfragmented packets. See *RFC 1858 Security Considerations for IP Fragment Filtering* for a complete discussion about this point.

3.1.2 An Introduction to ICMP Packets

ICMP is a protocol designed to communicate errors and information between hosts that are processing IP datagrams. You can find the specification of ICMP in RFC 792. It is used for purposes like informing that a host is unreachable or that a sender is sending packets too fast.

The ICMP messages that most people are familiar with are the ones that are generated by the ping command, but in fact there are many different types of ICMP messages. Ping generates an ICMP *echo request* message and expects to receive an *echo reply* message in response. Echo request is a relatively safe message, but any of the ICMP messages can be used by an outsider in order to gain some knowledge of your network or to directly attack your system. Also, like every protocol that you allow, ICMP messages can be used to overwhelm your systems in a *denial of service* attack.

Each ICMP message consists of a type plus a code, both of which are small integer values. Unlike the higher layer protocols, such as TCP or UDP, there is not a source port nor a destination port, just the message type and code.

Every ICMP message has the following format:

Type	Code	Checksum
unused		
Internet Header + 64 bits of Original Data Datagrams		

When configuring firewall filters you *could* disable all ICMP messages in both directions, if you don't care about the different types of message. This may make it difficult for you and your users to troubleshoot access problems, but will be safer and simpler for you. You also have to consider that some ICMP messages are used by network management applications (principally echo and address mask).

We now look at each of the ICMP message types. For each message type we describe ways in which it could be abused by an attacker and suggest a suitable filtering policy. We show examples of the firewall connection definitions to implement our suggestions in 6.20, “Filtering Specific ICMP Messages” on page 130. There is a summary of all the ICMP message types and codes, including RFC information where appropriate, in Appendix B, “Summary of ICMP Messages Types” on page 273.

3.1.2.1 Echo and Echo Reply Messages

Type	Description	Code
8	Echo	0 - No code
0	Echo reply	0 - No code

Description: The echo (also called echo request) message is used to check if a host is up or down. When a host receives the request, it sends back an echo reply message. These messages are usually generated by a ping command, but may also be generated by a network management station that is polling the nodes of a network.

Firewall Considerations: Echo request can be used by an outsider to map your network. We suggest you allow the outgoing echo request and incoming echo reply. Disable the incoming echo request and outgoing echo reply.

You could consider enabling this facility to some key hosts, such as the router of your network provider. You might allow incoming pings to the nonsecure adapter of the SNG.

3.1.2.2 Destination Unreachable Message

Type	Description	Code
3	Destination unreachable	0 - Net Unreachable 1 - Host Unreachable 2 - Protocol Unreachable 3 - Port Unreachable 4 - Fragmentation Needed and DF Set 5 - Source Route Failed 6 - Destination Network Unknown 7 - Destination Host Unknown 8 - Source Host Isolated 9 - Communication with Destination Network is Administratively Prohibited 10 - Communication with Destination Host is Administratively Prohibited 11 - Destination Network Unreachable for Type of Service 12 - Destination Host Unreachable for Type of Service 13 - Communication Administratively Prohibited by Filtering 14 - Host Precedence Violation 15 - Precedence Cutoff in Effect

Description: These messages are generated by hosts or intermediate routers in order to notify that a session cannot be established.

Firewall Considerations: An outsider can force nodes of your network to generate these packets in order to obtain knowledge of your network; for example, they can use a port scanner to learn which services you are providing. If you reply with a port unreachable, they will know that you are not providing this service (this type of information can also be gathered for TCP services by using stealth scanning).

You should receive these messages, as they may provide useful information for troubleshooting. You should only send them through the secure interface, because if you send them through the nonsecure interface, it will help outsiders to map the services that you are offering.

3.1.2.3 Source Quench Message

Type	Description	Code
4	Source Quench	0 - No code

Description: This message is generated by a host or a router when it wants the sender to slow down the rate at which it is sending packets. The IP stack passes this packet to the upper layers, and they are responsible for slowing the rate down.

Firewall Considerations: This message could be used by an attacker (probably combined with IP spoofing) in order to make a very effective denial of service attack. Unfortunately it is more often a legitimate message, so if you decide to

filter it, you may cause problems due to lost packets. We suggest you allow it to be sent and received, but also log the received messages for later analysis.

3.1.2.4 Redirect Message

Type	Description	Code
5	Redirect	0 - Redirect Datagrams for the Network 1 - Redirect Datagrams for the Host 2 - Redirect Datagrams for the Type of Service and Network 3 - Redirect Datagrams for the Type of Service and Host

Description: This message is generated by a router when it receives a packet from a host and forwards the packet to another router that is on the same network as the host from which it received the packet (not the original sender, the last hop sender). This message is intended to modify the routing table of the receiver so that the router does not have to do unnecessary work. The example in Figure 14 helps to explain how this works:

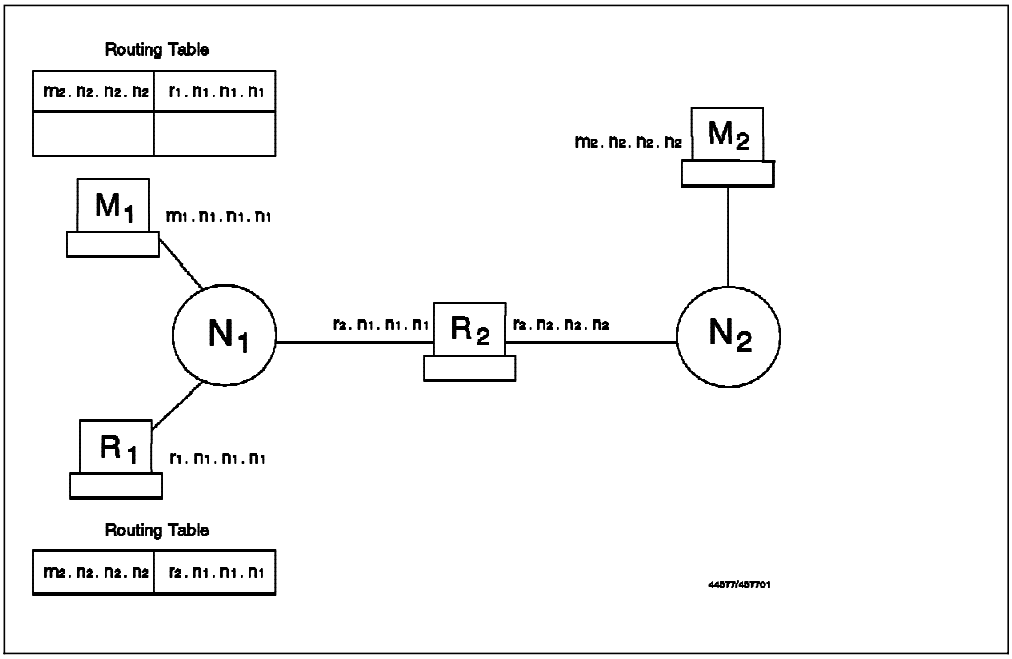


Figure 14. ICMP Redirect, Configuration

Suppose that machine M1 in network N1 wants to send a packet to machine M2 in network N2. As it is not directly connected to network N2, it looks at its own routing table in order to find who it must send the packet to. So it sends the packet to router R1 (see Figure 15 on page 28).

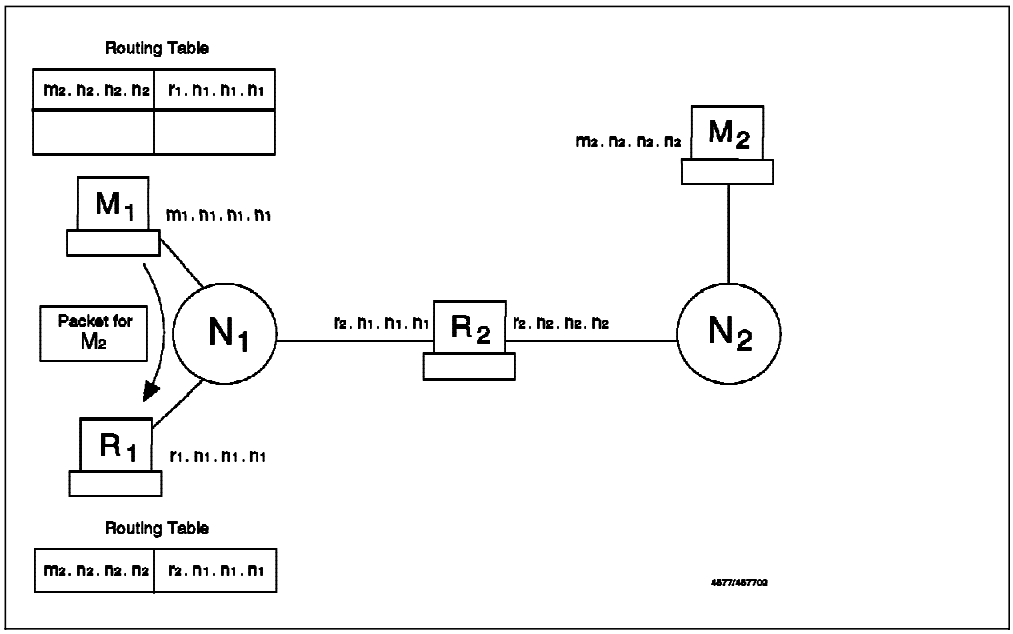


Figure 15. ICMP Redirect, M1 Sends Packet to R1

Router R1 receives the packet. As it is not directly connected to network M2, it looks at its own routing table in order to find who it must send the packet to. It forwards the packet to router R2 (see Figure 16).

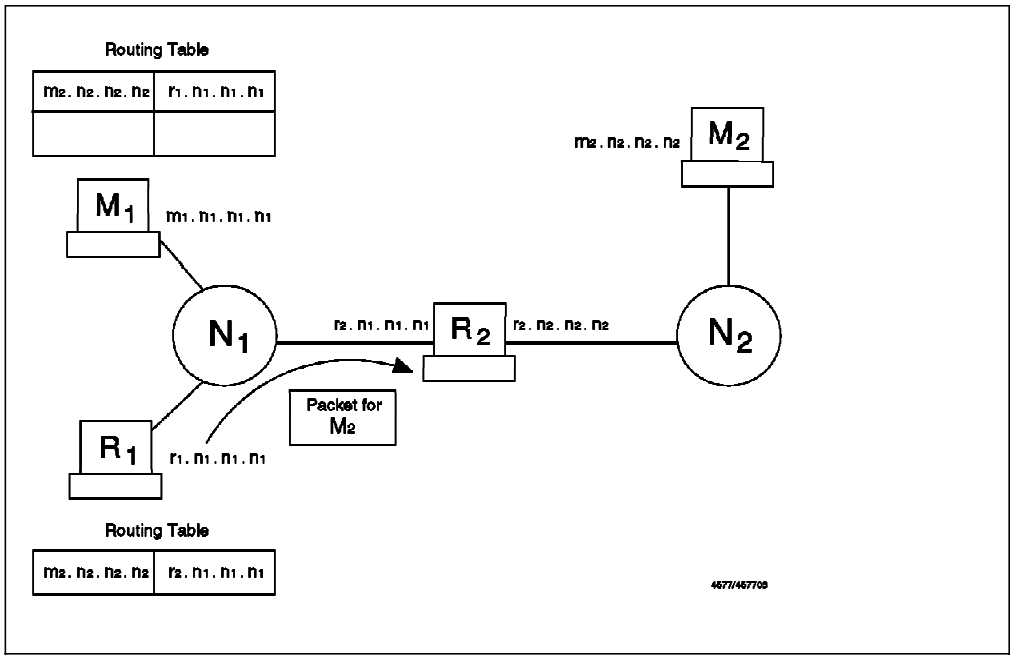


Figure 16. ICMP Redirect, R1 Sends Packet to R2

Router R1 realizes that it sent the packet through the same interface on which it was received. So instead of R1 receiving messages for M2 from M1 and then resending them to router R2, it sends a redirect message to M1 telling it to use R2 as the router in order to reach M2 (see Figure 17 on page 29).

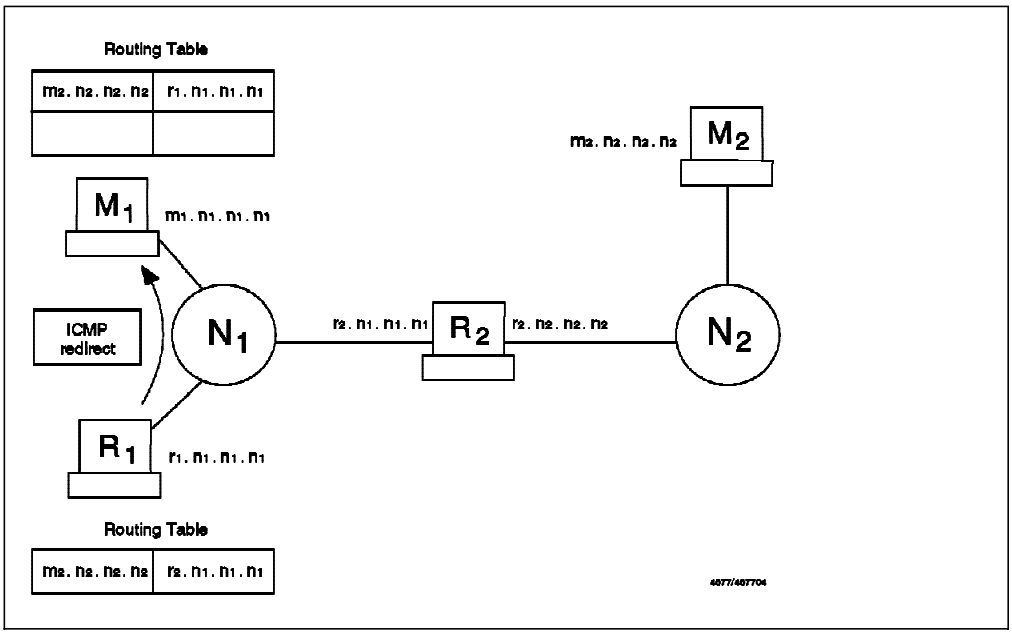


Figure 17. ICMP Redirect, R1 Sends Redirect Message to M1

Machine M1 receives the ICMP redirect message from R1 and updates its routing table in order to be more efficient (see Figure 18).

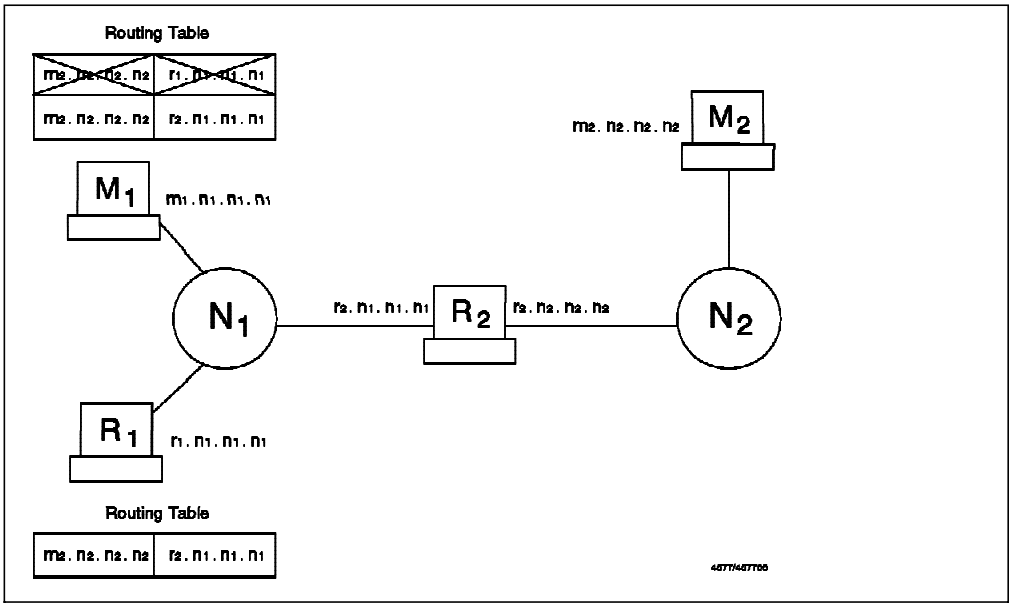


Figure 18. ICMP Redirect, Dynamically Updated Routing Table in M1

Firewall Considerations: Redirect has, as described, a very specific legal use. However it can be abused by a cracker to subvert the routing table and thereby allow IP address spoofing. Redirect is not supposed to cross a router (the packet is only sent when the sender and both routers are on the same physical network). It may be legal to receive this in the firewall directly if your routing tables are not properly set up. For the same reason, you might allow the firewall to send this type of message.

Our recommendation is to send and log this packet, but not to receive it, as your routing tables should be determined only by you. It is also recommended to

notify the owners of the machines to which you sent redirects so that they can correct their routing tables.

3.1.2.5 Time Exceeded Message

Type	Description	Code
11	Time exceeded	0 - Time To Live Exceeded in Transit 1 - Fragment Reassembly Time Exceeded

Description: Time to live exceeded is generated by a router when it has to forward a packet with a time to live (TTL) value of zero. Fragment reassembly time exceeded is generated by a host when it does not receive all the fragments needed to reassemble a packet.

Firewall Considerations: Enable this for incoming packets so your hosts can perform error recovery. For outgoing packets, allow all fragment reassembly time exceeded messages but not the TTL exceeded messages.

The reason that we recommend blocking TTL exceeded messages from going from the secure network to the nonsecure network is that an attacker can use a tool called traceroute to find out which hosts are the routers in your network. This tool manipulates the TTL option of a UDP packet, in order to receive an ICMP TTL exceeded message in response (see 6.15, “Traceroute” on page 118). Blocking the outgoing TTL messages will help you hide your network structure.

3.1.2.6 Parameter Problem Message

Type	Description	Code
12	Parameter problem	0 - Pointer Indicates the Error 1 - Missing a Required Option (RFC 1108) 2 - Bad Length

Description: This message is generated when a host that is processing a packet finds a problem in the header parameters that forces the packet to be discarded.

Firewall Considerations: An outsider will gain no information with this packet, so allow it to flow in both directions in order to report problems.

3.1.2.7 Time Stamp and Time Stamp Reply Message

Type	Description	Code
13	Time stamp message	0 - No code
14	Time stamp reply message	0 - No code

Description: The time stamp message is used to know the time in milliseconds since midnight. It receives as an answer a time stamp reply message.

Firewall Considerations: This protocol may be used by an attacker as a mapping tool (an alternative to ping). We didn't find any reason for allowing it.

3.1.2.8 Information Request Message

Type	Description	Code
15	Information request message	0 - No code
16	Information reply message	0 - No code

Description: This message is used by a host that is booted across the network to learn in which IP network it is located. It sends an information request packet with both the source and target fields set to zero. The replying host will send the reply with the complete address specified, so the host will now know which IP address it must use.

These messages are obsoleted by new protocols, like RARP, BOOTP and DHCP. Also RFC 1122 says that a host should not implement this protocol.

Firewall Considerations: This message is for local networks only, so it does not need to cross a router. The IBM Firewall should not generate requests, because it knows its IP interfaces, and certainly there is some better place to generate the replies than your firewall, so block it.

3.1.2.9 Address Mask Request and Address Mask Reply

Type	Description	Code
17	Address mask request	0 - No code
18	Address mask reply	0 - No code

Description: The address mask request message is sent when a node wants to know the address mask of an interface. It expects to receive as an answer an address mask reply message containing the mask of that interface.

Firewall Considerations: This message can be used by outsiders to learn the topology of your network. There were also cases in which a TCP/IP stack took inappropiate actions when it received an unsolicited address mask reply. The address mask request message may be generated by a network management station, such as Netview for AIX. AIX 4.1.4 by default will not answer this request unless you explicitly enable this using the `no` command icmpaddressmask option. Do not allow them in any direction.

3.1.2.10 Router Advertisement and Router Solicitation Message

Type	Description	Code
9	Router advertisement	0 - No code
10	Router solicitation	0 - No code

Description: These messages are used by hosts in order to dynamically discover the routers in a network. It is specified in RFC 1256, and the current status of the protocol is elective (as listed at the time of writing in the latest RFC of Internet Official Protocol Standards, RFC 1880). When the host boots, it sends a router solicitation message in order to discover the neighboring routers. The routers reply with router advertisement messages. The router also advertises periodically its routes in an unsolicited way.

Firewall Considerations: These messages are supposed to be for local networks only. They may be received by your firewall, but you should not trust any information they give you. Block these messages.

3.1.2.11 Domain Name Request and Domain Name Reply Messages

Type	Description	Code
37	Domain name request	0 - No code
38	Domain name reply	0 - No code

Description: These messages are used by hosts in order to learn the domain associated with an address. The host sends a domain name request message and receives as an answer a domain name reply. It is specified in RFC 1788, and the current status of the protocol is experimental.

The idea of this protocol is to substitute the IN-ADDR domain defined in the domain name server (the one that is used in order to translate IP addresses to domain names). Using this protocol, each host will be responsible for the translation of its own IP addresses. The RFC requires every host to implement an ICMP domain name server and also suggests that every host should implement an application for sending the ICMP domain request.

Firewall Considerations: Block it, because it is not currently used.

3.1.2.12 Traceroute Message

Type	Description	Code
30	Traceroute	0 - Outbound Packet successfully forwarded 1 - No route for Outbound Packet; packet discarded

Description: This message is used in order to implement traceroute (a useful network debugging tool) in a more efficient way. It is specified in RFC 1393, and the current status of the protocol is experimental.

The implementation has two parts:

- A new IP option
- The new ICMP traceroute packet

When a host wants to discover the path to a node, it sends a packet (for example, an ICMP echo request) with the new IP option. Then every router that forwards the packet will also send an ICMP traceroute message to the sender, informing it whether the packet was succesfully forwarded or if it was discarded.

Firewall Considerations: Incoming, (used to trace routes from the secure network to the unsecure network) this packet can be allowed. If you want to hide your internal network structure (you probably should), the outgoing packet must be blocked.

3.1.3 An Introduction to TCP Packets

TCP is the transport layer protocol that is used by most IP applications. For example, ftp, telnet, smtp (mail) and http (World Wide Web) are all higher-layer protocols that use the TCP transport layer. TCP is defined in RFC 793.

TCP provides the application with a reliable end-to-end connection. It takes care of retransmission, lost and duplicate packets and reordering of packets. When a host establishes a connection to another machine using TCP/IP, both hosts will be able to use the same connection in order to send information (that is, it is a two way channel).

Figure 19 shows the format of the TCP header.

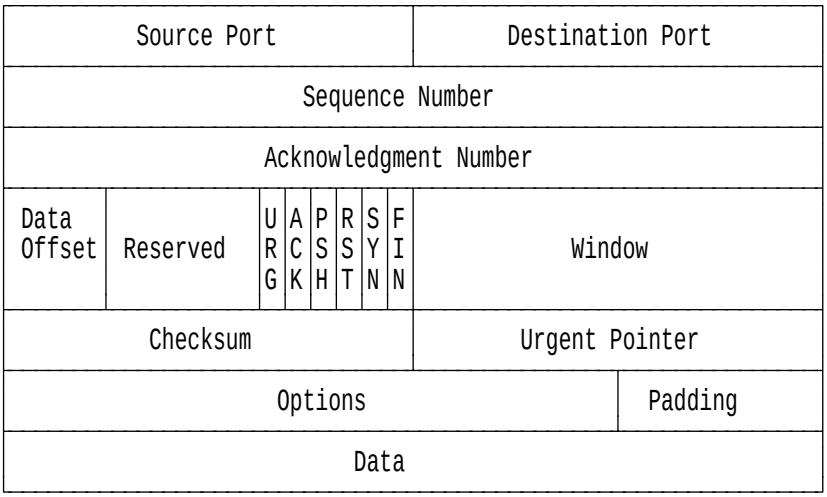


Figure 19. The TCP Packet Header

From the firewall point of view, the most important parts of the TCP packet are the source port, destination port and ACK bit. The source port and the destination port are used to identify which process is using a TCP connection. A TCP/IP connection is uniquely defined by:

```
< Source Address, Source Port, Destination Address, Destination Port >
```

There are some particular ports that are reserved for specific applications, while others are dynamically allocated for the processes that need them. The reserved ports are normally referred to as *well-known ports*. For example, port number 23 is reserved for incoming Telnet connections. Port numbers below 1023 are usually described as *privileged*, meaning that an application needs root authority to use them. This is only a convention, so you cannot base your security policy on it, but one side-effect is that the client end of a TCP session

normally uses a port number above 1023. We enforce this by policy in our firewall rules.

Let us now see how a TCP connection is established. A TCP session is initiated by a three-way synchronization sequence as shown in Figure 20.

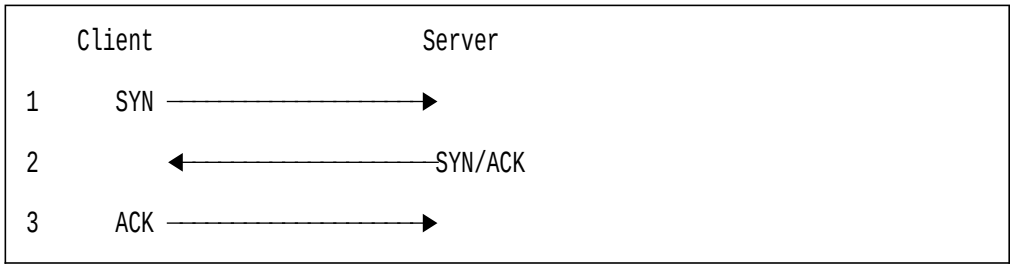


Figure 20. TCP/IP Synchronization Sequence

The acknowledgement (ACK) flag is used by one end of a session to tell the other end that its previous packet was received. The result is that the only packet in the TCP/IP session without the ACK flag set is the SYN packet that creates the session in the first place. So when a connection is created, the first packet does not have the ACK flag set, but all the following packets will have it. Firewall rules use this to control the direction in which a session can flow. If we want to prevent someone from creating connections from the outside (unsecure network) to the inside, we can specify a rule with a protocol specification of *tcp/ack*. This will block the first packet, thereby preventing the establishment of a connection.

This is an effective way to prevent unwanted sessions from being established from outside the secure network. However, for complete security, you should aim to use proxy servers or SOCKS wherever possible.

3.1.4 Use and Abuse of TCP Ports

If a service normally uses a well-known protocol, that does not mean that it can not use another port. For example, the Telnet server usually uses port 23, but nothing prevents it to be run on another port, for example, port 5234.

This must be considered because it might be used to circumvent the firewall restrictions, either by an outsider or an insider. Often, holes in the firewall security are not directly created by attackers, but by unhappy insiders who consider the firewall to be unnecessarily restrictive. An insider that wants to provide an outside access that is not permitted may use a nonstandard port in order to do it. For example, if you prevent your users from providing HTTP servers but allow connections from outside to nonprivileged ports, a user can provide HTTP access using port 5234.

3.1.4.1 Source Porting

An outside privileged port might be used by an outsider to circumvent your security policy. If, for example, you allow outside access from tcp/20 (a port usually used by an FTP server for data transfer), an outsider may use this port in order to run another service, for example, a Telnet client. Use of the tcp/ack protocol flag can prevent incoming connections, but if you allow an incoming connection from a particular port (as is needed if you want to provide FTP access for your users without implementing a proxy), you will open a security hole in your firewall. We will come back to this point when we describe firewall definitions for FTP services in 6.4, “FTP. File Transfer Protocol” on page 86.

3.1.4.2 Stealth Scanning

We have seen how a firewall can control the direction of TCP connections using the ACK bit of the packets. When you want to disable a connection from one direction or the other, you just block the first packet (the one that does not contain the ACK bit), preventing the establishment of the connection.

If an outsider is trying to scan your network in order to discover which machines you have and which services you provide, they will use a port scanner. Usually port scanners try to open a connection to the port. If you use the ACK bit checking in the firewall, this will block the attack.

However, it is possible to scan a network without sending any packet with the SYN bit on. In order to do this, the attacker sends a packet that *looks like* something from the middle of a legitimate session, that is, with the ACK bit on. If the destination port is active, the host will realize that it is not part of a session in progress and send a reset response. If the port is not active, there will be no response. Other types of TCP packets may be used to perform similar types of scanning, such as a packet with SYN:FIN, ACK in the header or one with the flag field set to 0. All of these packets will be rejected, but the fact that they *are* rejected provides some information about the target machine. This is called *stealth scanning*.

If you want to allow IP forwarding on the firewall and rely on the SYN control, you must be aware that your network might be scanned using these techniques.

Should you care about this? After all, the attacker cannot establish a useful session this way. The danger is, that it provides a way of mapping the contents of your secure network. This knowledge may not be directly useful, but it can become useful if combined with some other back door access. The lesson is this: wherever possible your firewall should be completely dual-homed and not allow any IP routing.

3.1.5 An Introduction to UDP Packets

UDP, like TCP, is a transport layer protocol, but it is less widely used by applications. Common applications using UDP include the domain name service (DNS) and the simple network management protocol (SNMP). UDP is defined in RFC 768.

Unlike TCP, UDP does not provide the application with a reliable end-to-end connection. Once a UDP packet has been sent, the sender has no knowledge about whether it has arrived or not. It is therefore up to the application to provide acknowledgment and sequence control, if required. UDP is connectionless. That is, each message is a separate entity with no expectation of responses or subsequent request messages. Applications will often mimic the operation of a connection-oriented protocol, for example, a client may use a dynamically allocated port to send a message, and then listen on that same port for a response.

Figure 21 on page 36 shows the format of the UDP header.

Source Port	Destination Port
Length	Checksum

Figure 21. The UDP Packet Header

From the firewall point of view, the only important parts of the UDP packet are the source port and destination port. The source port and the destination port are used to identify which process is using a UDP connection. A UDP/IP connection is uniquely defined by:

```
< Source Address, Source Port, Destination Address, Destination Port  >
```

As in the case of TCP, certain well-known ports are reserved for specific applications. For example, DNS uses port 53, and SNMP uses ports 161 and 162.

Because of its connectionless nature, UDP does not have the three-way synchronization sequence of TCP (see Figure 20 on page 34). This means that we do not have the ability to create rules based on the direction in which a session is established. As a result you should avoid routing UDP sessions through the firewall directly. If you do allow them through, you should only allow specific pairs of known end points.

Chapter 4. Installation

The installation of IBM Firewall 3.1 is a two-stage process:

1. Install and configure AIX operating system.
2. Install and configure IBM Firewall 3.1.

When installing IBM Firewall 3.1 please keep the following points in mind:

- Keep It Simple and Secure (KISS).
- Physically secure your system in a locked area.
- Make a checklist of things you change so you can periodically check to make sure those settings are still the same.
- Consider how secure the network structure is between your firewall and the secure network.
- Run the minimum number of services necessary (KISS).
- User IDs should be kept to a minimum and set up using IBM Firewall 3.1.
- Remove any compilers, assembler or any other computer language that allows system calls.
- Remove tracing tools, for example tcpdump and iptrace.
- Use the audit and logging functions to monitor the system.

4.1 Requirements

The software and hardware requirements for IBM Firewall 3.1 include the following:

- AIX Version 4.1.5 or later (we recommend 4.2.1)
- A Uniprocessor RISC System/6000 supported by AIX 4.1.5 or above
- At least two network interfaces
- At least 64 MB physical memory

The firewall software will require approximately 50 MB of disk space. You should also expect to use at least 50-200 MB for log files.

4.2 Install AIX

AIX is a multiuser, multipurpose operating system. It offers a wide variety of services that are not needed when installing a firewall.

We recommend that you install AIX from scratch for IBM Firewall 3.1 and aim to install a minimal system. A fresh install of AIX will ensure you don't have any other software installed except the minimum you will need to run IBM Firewall 3.1.

Take the option to install the Trusted Computing Base, as it can only be installed during AIX installation. This will give you the option of later using it to add a further level of security to your system.

4.2.1 Post-AIX Installation

Once AIX was installed, we performed a number of tasks before installing IBM Firewall 3.1.

4.2.1.1 Additional Filesets

To allow us to run DNS, diagnose performance issues, and back up the system, we installed the following filesets:

bos.net.tcp.server	For DNS software
bos.acct	System performance & accounting
bos.sysmgt.sysbr	AIX Backup utilities

Use `lslpp -l fileset.name` to see if these are installed. Use `smit install_latest` to install them.

Handy AIX Install Tip

Use the preview option in smit before every install/patch. It will help you identify problems before you actually begin modifying things.

4.2.1.2 Install AIX Fixes

Install the current set of AIX fixes. You can obtain these fixes by contacting your local IBM Service Center or download them using the fixdist tool available at http://service.boulder.ibm.com/rs6000

We installed the following fixes for AIX 4.2.1:

bos.net.tcp.server	4.2.1.5
bos.net.tcp.client	4.2.1.9
bos.up	4.2.1.6
bos.net.ppp	4.2.1.1
bos.rte.security	4.2.1.5
bos.rte.libc	4.2.1.5
bos.net.tcp.smit	4.2.1.1

Note: These are the patches that were available in November 1997.

IBM also distributes security fixes quarterly in a package. Download IX72520, or search for a more recent fix package.

Use `smit update_all` to install fixes.

SNMP Support

To enable SNMP support in IBM Firewall 3.1 you will need to install the SystemView agent filesets:

SystemView Agent for AIX	1.4.2.0
SystemView Agent for AIX SNMP Mapper	1.4.2.0

4.2.1.3 Cleanup

We performed several tasks to help simplify and secure the system.

1. Enable AIX skulker in cron to clean up /tmp and /var/tmp nightly:

   ```
   crontab -e
   ```

2. Check for world-writeable files. These files can be updated or deleted by any user, making them an area of concern. Search for them with the command:

   ```
   find / -perm -0002 \( -type -o -type d \) -print
   ```

   Only /tmp and some of the directories under /var should be world writeable.

3. Create a valid dump device to ensure a valid system dump is made if the system crashes.

   a. Determine dump device size requirement with `sysdumpdev -e`.

   b. Create a logical volume in the root volume group for a dedicated dump space. We created a 5 x 4 MB dump logical volume called hd7 for our dump space:

      ```
      mklv -y hd7 -t sysdump  -a e  rootvg 5
      ```

   c. Assign the dump device you created as the primary dump area with:

      ```
      sysdumpdev -P -p /dev/hd7
      ```

   d. Assign the base paging area as the secondary dump area (just in case) with:

      ```
      sysdumpdev -P -s /dev/hd6
      ```

4. Uninstall as much of CDE as you can as it will be disabled by the firewall hardening process. We removed these filesets:

   - X11.Dt.bitmaps
   - X11.msg.en_US.Dt.helpmin
   - X11.Dt.ToolTalk
   - X11.loc.en_US.Dt.rte
   - X11.Dt.rte

5. We kept X-Windows (X11 filesets) installed. However we recommend that you remove X-Windows and use the Remote Configuration tool on another system to administer the firewall. The X-Windows software is a good example of software you can do without on the firewall and should remove to keep it simple and secure.

6. We also suggest to move root's home directory to /home.root (could use /home/root instead). This means that any special profiles, smit.* files, and any other files used in testing and installing can be kept away from the main / directory, which means a) the system is tidier, and easier to clean our junk safely, and b) you can restrict the permissions on /home.root just in case anyone does get in. Use /home.root (in / partition) just in case of errors when other logical volumes fail to mount correctly; other people prefer to use /home/root (in /home partition) to ensure that root's temporary files, especially smit.*, cannot fill the root partition.

4.2.1.4 Filesystems

As the system typically will need to spool mail and log plenty of syslog output, it pays to keep /var big enough right from the start. Larger sites might have 1 GB or more of spool space. Of course the root file system should be big enough to not accidentally run into a full root file system problem. In our case we expanded the / filesystem to 8 MB and /var to 200 MB.

Log files from smit (`smit.log` `smit.script`) can erode space in the root file system. Keep an eye on them, or link to somewhere else, for example /tmp (don't forget skulker!).

Use `smit chfs` to increase the size of the filesystems. We used `chfs -a size=16384 /` to increase root to 8MB and `chfs -a size=409600 /var` to increase var.

Note: A separate filesystem to store log files will help avoid problems with the log filesystem filling up.

Keep an eye on the amount of paging space your system uses with `lsps -a`.

4.2.2 Users

During the installation of IBM Firewall 3.1, all users other than root, daemon, bin, adm and nobody will be removed. The root account will be disabled for remote logins.

For all user IDs in the system that are not used for regular logins, you should define a mail alias that transfers the mail to a local administrator. Otherwise, mail could pile up accidentally in a mailbox without anyone ever noticing it.

Note: You can further optimize the performance of your system using *Understanding IBM RS/6000 Performance and Sizing*, SC24-4810-00.

4.3 Firewall Installation

Installing IBM Firewall 3.1 is a straightforward AIX installp function and we installed the firewall using SMIT in about 10 minutes.

After the installp process is complete it is *necessary to reboot the machine*. The installp messages tell you to do this, but it is easy to overlook them.

4.3.1 Install the Code

Before you begin the installation of the firewall software, move the `smit.log` and `smit.script` files aside. The firewall installation will log all its activity to these files. You will be able to use these files to list the changes made to the system by the install process.

Start the installation program at the AIX command line with `smitty install_latest`. You need to define the installation media (CDROM or Local Disk) and select the components that you need to install. These are:

- 3.1.1.0 Base IBM Firewall
- 3.1.1.0 IBM Firewall Common Libraries and Catalogs
- 3.1.1.0 IBM Firewall Remote Configuration Client
- 3.1.1.0 IBM Firewall Report Generation Utilities

- 3.0.0.0 Netscape Navigator

We tried to install only the Base and Common Libraries, but ended up installing the lot as the others are requisite filesets of the base.

The install options we initially selected can be seen in Figure 22.

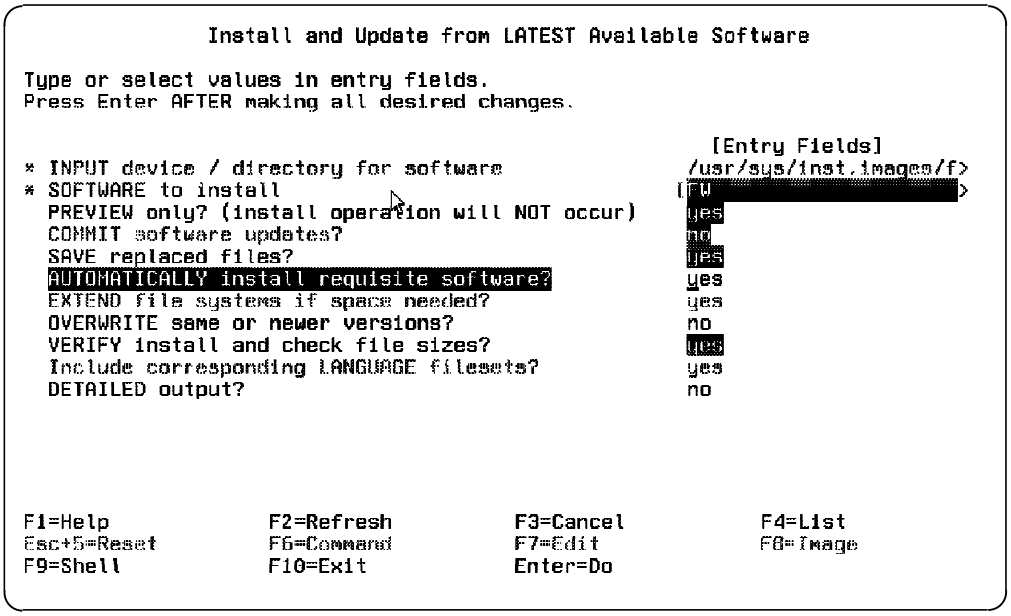


Figure 22. Preview Install of IBM Firewall 3.1 Using smitty

After a successful preview we changed the preview option to no and restarted the installation.

We have a second RS/6000 we will use to administer the firewall. On this system we installed the IBM Firewall 3.1 Remote Configuration Client filesets:

- 3.1.1.0 IBM Firewall Common Libraries and Catalogs
- 3.1.1.0 IBM Firewall Remote Configuration Client
- 3.1.1.0 IBM Firewall Report Generation Utilities
- 3.0.0.0 Netscape Navigator

4.3.2 Firewall Hardening

A major part of the IBM Firewall 3.1 installation is a process called hardening. System resources that might compromise security are disabled to help further secure the system. The IBM Firewall 3.1 hardening process does the following:

- If the AIX Common Desktop Environment (CDE) is installed on the system, the installation process disables it.
- All unnecessary programs are removed from inittab (for example, nfs and printer daemons).
- Startup entries are removed from /etc/rc.tcpip.

  The remaining entries are:

  – syslogd

  – named

  - inetd
- Startup entries are added to /etc/rc.tcpip for IBM Firewall 3.1 components.
- All unnecessary functions are removed from /etc/inetd.conf.

  The remaining functions in /etc/inetd.conf are:

  - ssld
  - Proxy ftp
  - Proxy telnet
  - Remote configuration server ibmfwrcs
- All users are deleted except root, daemon, bin, adm, nobody, and any previous IBM Firewall users.
- Groups ecs, uucp, and usr are deleted.
- All files and directories not owned by any user are deleted.
- Nonsecure applications are disabled.

  These nonsecure applications were disabled on our system:

  - /usr/bin/tftp
  - /usr/bin/utftp
  - /usr/sbin/tftpd
  - /usr/bin/uucp
  - /usr/sbin/uucpd
  - /usr/bin/rcp
  - /usr/bin/rlogin
  - /usr/sbin/rlogind
  - /usr/bin/rsh
  - /usr/sbin/rshd
- Any previous IBM Firewall users are migrated to this new version.
- The file system integrity checker database is generated.

4.3.3 Post Firewall Installation

After installing IBM Firewall 3.1 we need to complete a few more steps.

4.3.3.1 Install IBM Firewall 3.1 Fixes

To ensure your firewall performs properly it's a good idea to regularly check for and install patches. After installation, we applied the following patches specific to IBM Firewall 3.1:

- 3.1.1.1 Base IBM Firewall (UR48521)
- 3.1.1.1 IBM Firewall Common Libs and Cats (UR48521)
- 3.1.1.1 IBM Firewall Remote Config Client (UR48521)

4.3.3.2 IP Forwarding

IBM Firewall 3.1 adds a line to /etc/rc.net which enables ipforwarding. This and other *no* options will be honored by IBM Firewall 3.1 unless contradicted by a rule. IBM Firewall 3.1 already turns off IP source routing so you don't need to turn it off by changing this line.

4.3.3.3 Setting the Secure Network Adapter

Before you can proceed any further you have to tell IBM Firewall which of its interfaces is connected to the secure network. You configure the secure interface from SMIT by selecting **IBM Firewall**, then **System Administration**, then **Secure Interface**, and **Add**, then select the interface you would like to be secure.

Note: The fast path to this menu is `smit fw_set_secure_adapter`.

4.3.3.4 Enable Remote Configuration

Will you be administering your firewall from the console or with the remote configuration client on another machine in the secure network?

The recommended method is to configure the firewall remotely with the remote configuration client.

Local configuration is possible with SMIT, but the GUI is easier to use. If you want to use the GUI locally you will need X-Windows, which really shouldn't be installed on a firewall.

To allow the remote configuration client to connect to the firewall, we made two changes to the /etc/security/rcsfile.cfg file:

1. Replace `local=yes` with `local=no`.
2. Replace `encr=none` with `encr=ssl`.

You also have to install the remote client configuration software on a machine with a Java-enabled browser in your secure network.

Don't Forget!

If you enable the default rules in IBM Firewall 3.1 you won't be able to remotely access the machine until you modify the rules to allow remote access.

4.3.3.5 Generate an SSL Key

You will need an SSL Key to securely communicate with IBM Firewall 3.1 using the Remote Configuration Client.

Complete instructions for generating an SSL Key with the `mkkf` command are in Chapter 5, "Using the Make Key File Utility (MKKF)" of *IBM Firewall For AIX Reference Version 3.1.1*, SC31-8418-00.

Note: The default key file is called fwkey.kyr. If you want to use a different keyfile name, make sure you change /etc/security/rcsfile.cfg to point to your new keyfile name.

4.3.3.6 Create an Administrator User

The three levels of user security in IBM Firewall 3.1 are:

1. Root
2. Firewall Administrator
3. Firewall User

You can only create a user a security level below yourself. For example only root can create a Firewall Administrator.

To make initial creation of users simpler we enabled remote login for the root user with the command `chuser rlogins=true root`. This will allow us to log in as root from the remote configuration client. We will turn this ability off later using `chuser rlogins=false root`. Allowing remote logins for root is a very bad idea, but not an unreasonable step during installation.

Note: You can create a firewall administrator user account with `smit fw_add_usr` on the console as root if you wish to avoid enabling remote root login.

To create an administrator user we followed several steps.

First we logged onto our administration RS/6000, then set our DISPLAY environment variable for X-Windows. We then started the IBM Firewall 3.1 Configuration Client with the command `fwconfig`.

The system presented us with the logon screen shown in Figure 23.

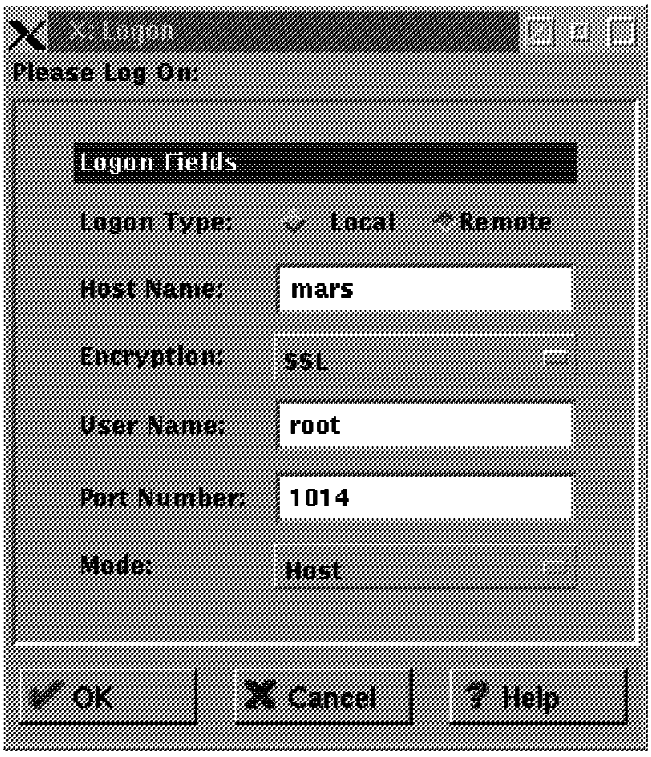


Figure 23. Remote Logon

Note: If you would like more information on the options on the following panels use the online help or refer to Chapter 10, "Administering Users at the Firewall" of *IBM Firewall for AIX Users Guide*, GC31-8419-00.

After authenticating yourself to the firewall you should be presented with the main IBM Firewall 3.1 Netscape screen, shown in Figure 24 on page 45.

Figure 24. Remote Configuration Client Main Screen

From here we selected **Users** to get to the User Administration screen shown in Figure 25 on page 46.

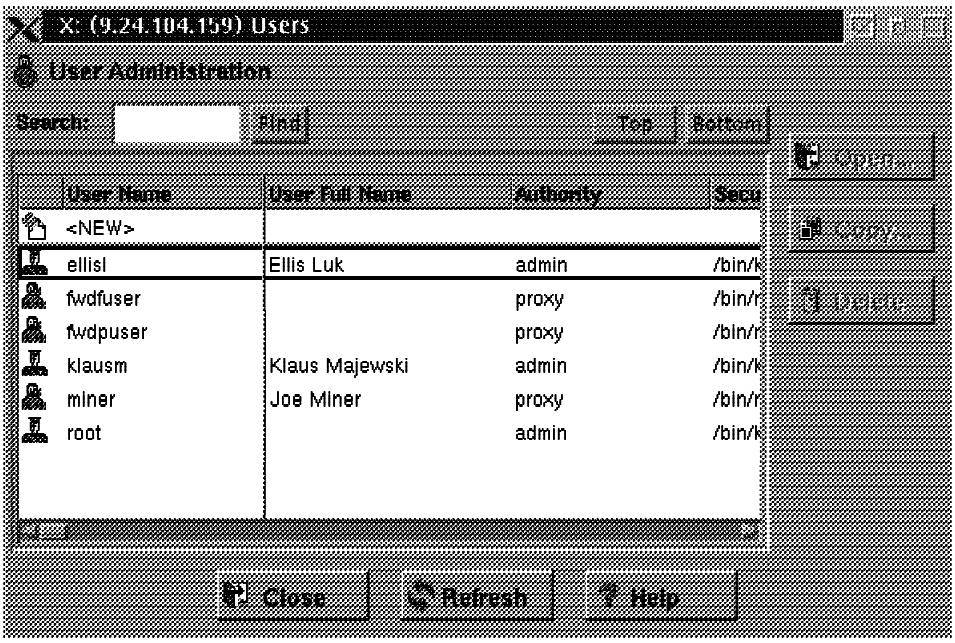


Figure 25. List User Screen

As you can see we have already created a couple of administrator accounts.

We will now create one more administrator account by selecting **<NEW>**.

The Add User panel shown in Figure 26 on page 47 is used to create administrator and proxy users. As we are root we will select **Firewall Administrator** as the authority level for our new user.

Next we changed the Secure Interface Shell to **/bin/ksh** so our administrator would be able to fully access the command line. We left the Nonsecure Interface Shell as **/bin/restrict.sh** because we don't want to allow access to the firewall from the nonsecure interface.

For the Authentication settings we deny any nonsecure access. As we would like the administrator to have broad access to the system, we allowed password authenticated access for all secure side activities.

You can modify these settings later to be more restrictive after you have completed your installation if you like.

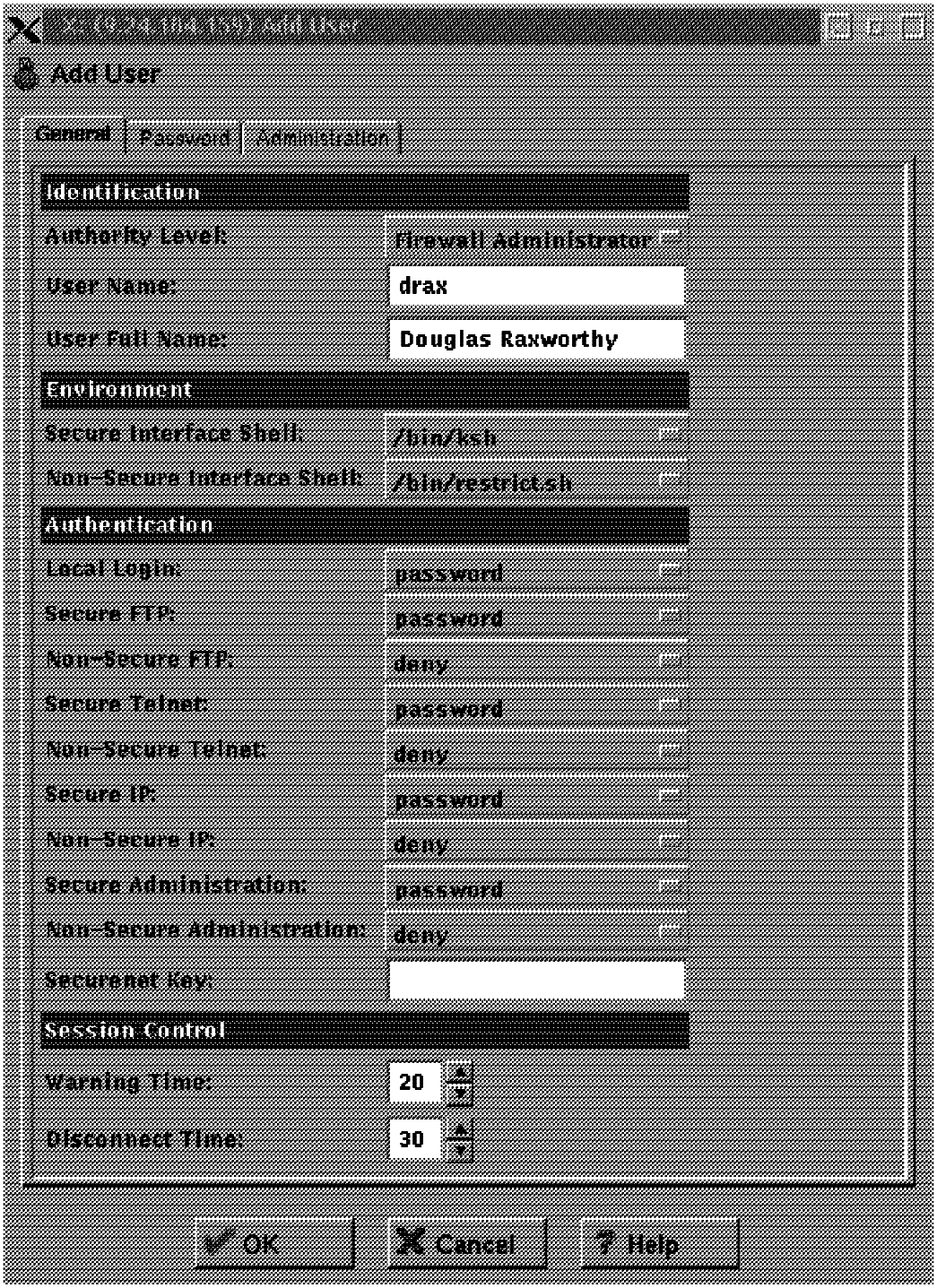


Figure 26. Add User

We selected the **Password** tab to see the Set Password panel seen in Figure 27 on page 48.

Here we set the password for the new user. The default password settings were sufficient for our needs.

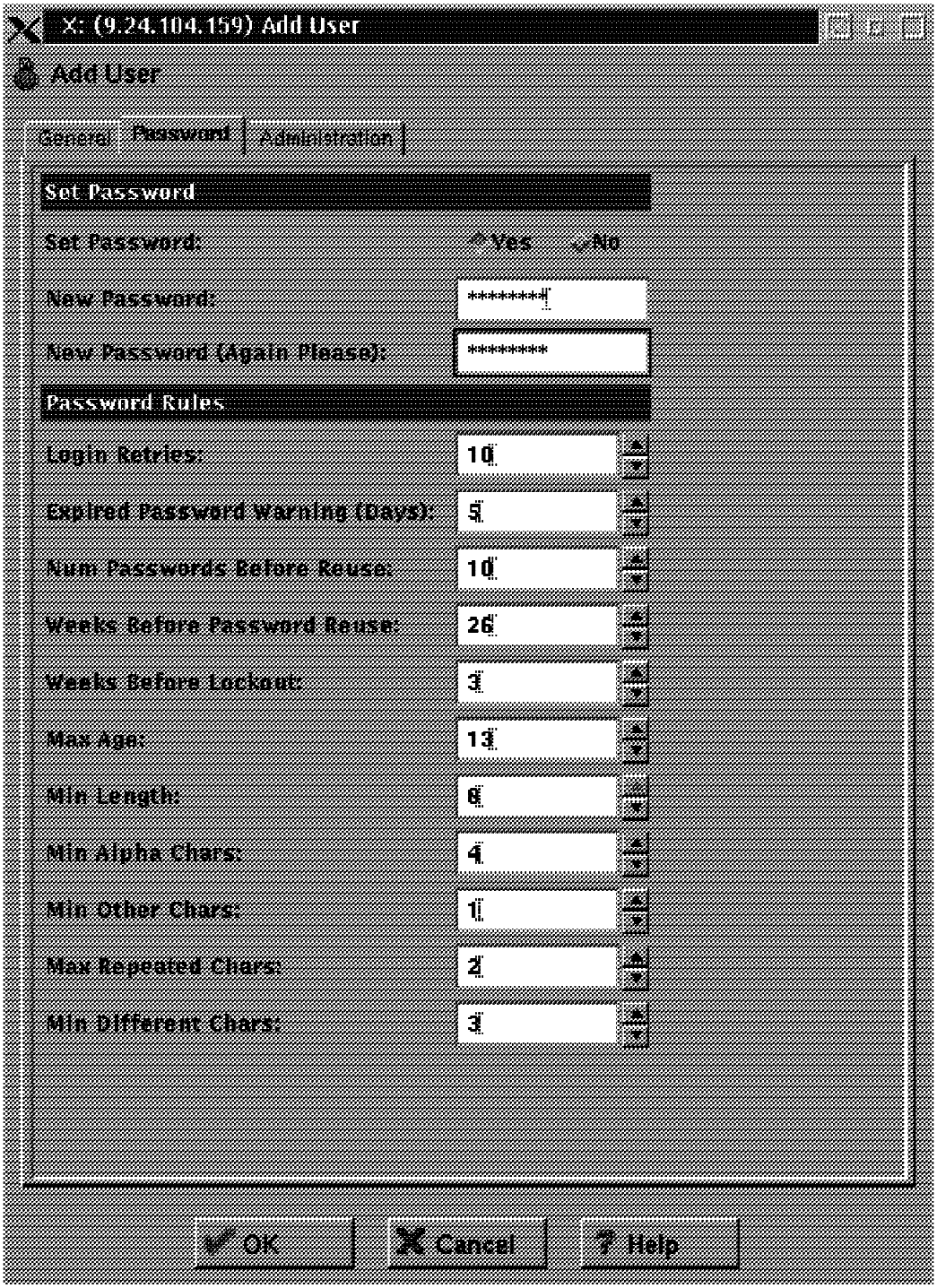


Figure 27. Defining the Password for a New User

The final tab in the Add User screen is **Administration** (see Figure 28 on page 49). We selected this but were happy with the defaults here also.

This is where you can begin to assign different jobs to different administrators. The disabled options become enabled if you select **Enterprise**.

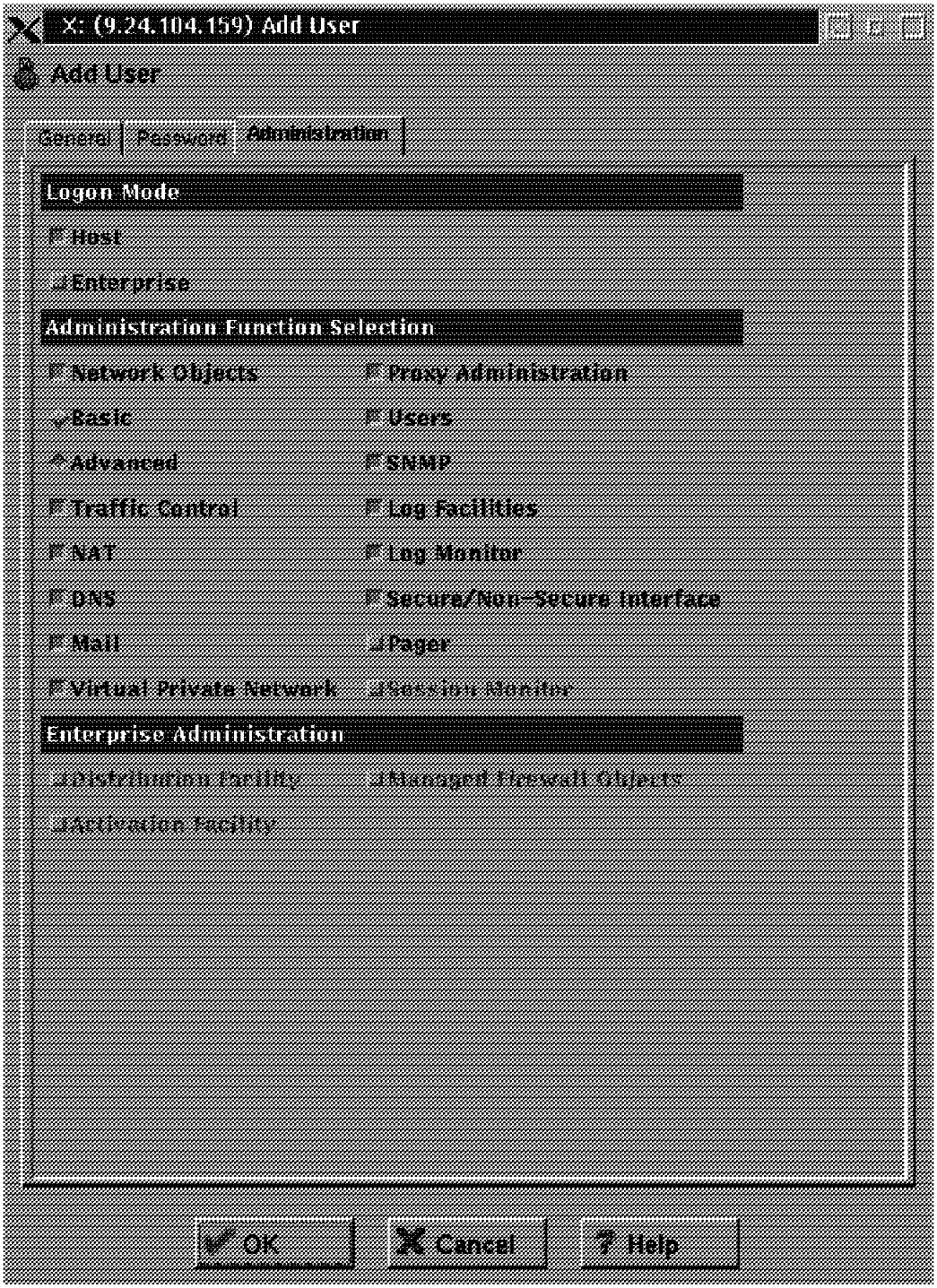


Figure 28. Administration Functions

Don't Forget

A newly created user will need to set their password. The initial password can be set in IBM Firewall 3.1 but the user must still set a new password before they will be able to log on with the Remote Client.

4.3.4 Checking Network Security

After having set up the firewall, how do you check its security? There are many ways to do checks; you will have to find a combination of the tools and commands available that suit your needs. We discuss using testing programs to probe for weaknesses later in this book using a variety of tools. See Chapter 13, “Testing Your Configuration” on page 243.

4.3.5 Checking System Security

If you have followed all of our recommendations on system setup there should not be too many additional things to check.

The key to your firewall's protection is its logs. This is how it warns you of attacks and other problems. In Chapter 14, “Logging” on page 251, we discuss ways to extract information from your log files, and other useful tools that can help you.

4.3.6 IBM Firewall 3.1 Configuration Replication

If you want to replicate or back up your IBM Firewall 3.1 configuration, all the files that the firewall modifies can be obtained from the file:

```
/etc/security/fwconfig.map
```

The only exceptions are any *.modem files in /etc/security.

Note: Don't forget your user information is managed by AIX.

Unfortunately we did not have sufficient time to run a replication test and must caution you not to try this without thorough testing.

A basic outline of the steps involved follows:

- Install IBM Firewall 3.1 on machine A
- Don't create any proxy or administration users
- Back up the files referenced by fwconfig.map
- Install IBM Firewall 3.1 on machine B
- Restore files backed up in /etc/security

In theory the same principle applies to an Enterprise Managed Firewall. The only difference is that the /etc/security files for managed firewalls are stored in a separate directory for each firewall.

Again we advise caution: fully test any replicated configuration before implementing in production.

4.3.7 Command Line Proxy User Generation

The various command line utilities (which will be expanded upon in a PTF currently under test) can allow you to generate proxy users from a list. An example of a script file using the *fwuser* command, which can be used to define proxy users when a file containing the user's data is already available, follows:

```
#-------------------------------------------------------------------

#!/bin/ksh

function genusr
{
  # Generate Proxy User Function
  /usr/bin/fwuser cmd=add username="${USERID}" \
    fullname="${NAME1} ${NAME2}" \
    password=yes \
    pwdvalue="${PASS}" \
    secftp="${SECFTP}" \
    secauth="${SECAUTH}" \
    remauth="${REMAUTH}" \
    remip="${REMIP}"
}

cat user.lst \
  ] awk -F: '{printf("%s %s %s %s %s %s %s %s\n",$1,$2,$3,$4,$5,$6,$7,$8)}' \
  ] while read USERID NAME1 NAME2 PASS SECFTP SECAUTH REMAUTH REMIP
do

genusr

done


exit
```

The file used with the above script is the following:

```
# Sample User List File
pepe:Jose Garcia:raton:password:password:password:password
alicer:Alice Rogers:forgot:password:password:sdi:password
johns:John Silver:donno:password:password:deny:deny
juanp:Juan Plata:platino:password:deny:deny:deny
```

This file contains the user ID, the user name, the password, and the authentication method for FTP from the secure side (secftp), telnet from the secure side (secauth), telnet from the nonsecure side (remauth) and remote IPsec client from the nonsecure side (remip).

We selected a few parameters of the fwuser command; you can modify the script to accept other parameters.

Chapter 5. IBM Firewall 3.1 Rule Base

The control mechanism of the firewall is defined by a rule base. Here we describe the contents of the rule base and work through some simple examples of defining rule base objects. In the next chapter we look at a number of services that you may want your firewall to handle and describe the rule base objects you will need to define.

5.1 Rule Base

The IBM Firewall uses a rule base to determine whether IP packets are passed or blocked at the firewall. The rule base contains a number of components, all related to each other. The main components are called *connections*, each of which exactly defines a specific type of IP traffic to be allowed or prohibited between two network components. Connections are built of one or more *services*. A service is a set of *rules*. A connection also defines the network endpoints (source and destination) to which the services apply. These are defined as *objects*, or *groups* of objects. Figure 29 shows the complete structure.

The recommended way to configure the rule base is to use the browser-based GUI. It is also possible to set up the rule base by using SMIT, but because it is easy to follow the same route in SMIT, we will only describe the GUI configuration.

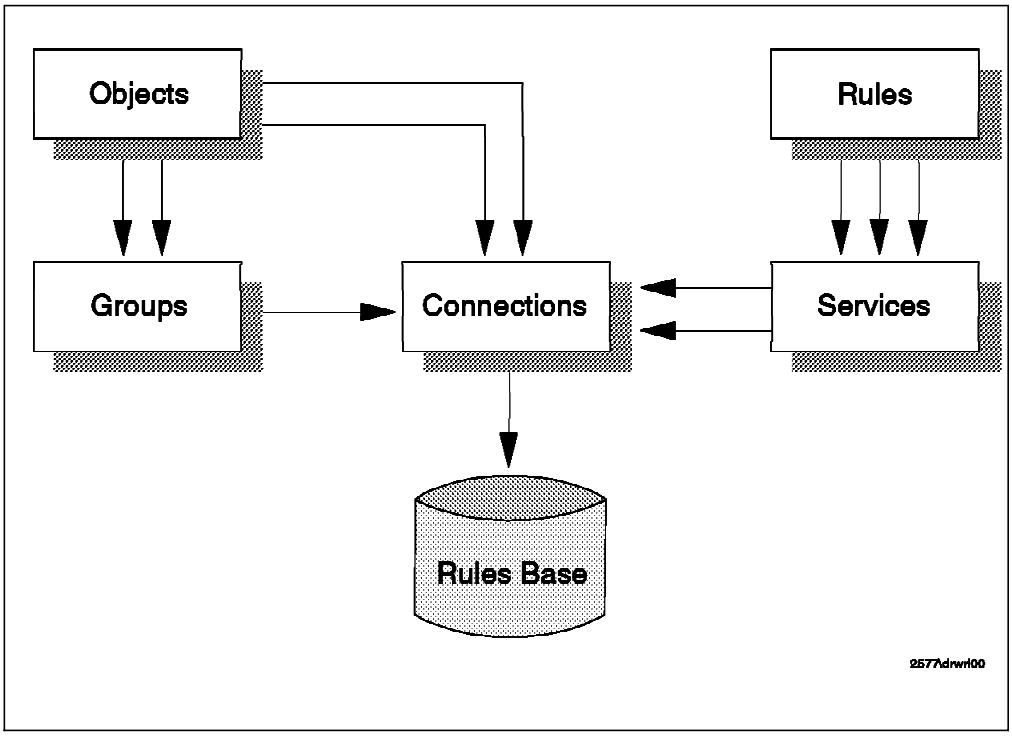


Figure 29. Rule Base Structure

5.1.1 Connections

A connection defines the IP traffic that is allowed or denied between two network components. A connection is built of a source and a destination component, that are connected by a service component, as shown in Figure 30. A service defines the type of IP traffic that is permitted or denied between the source and destination. The source and destination in a connection are each defined by an *object* or a *group* of objects.

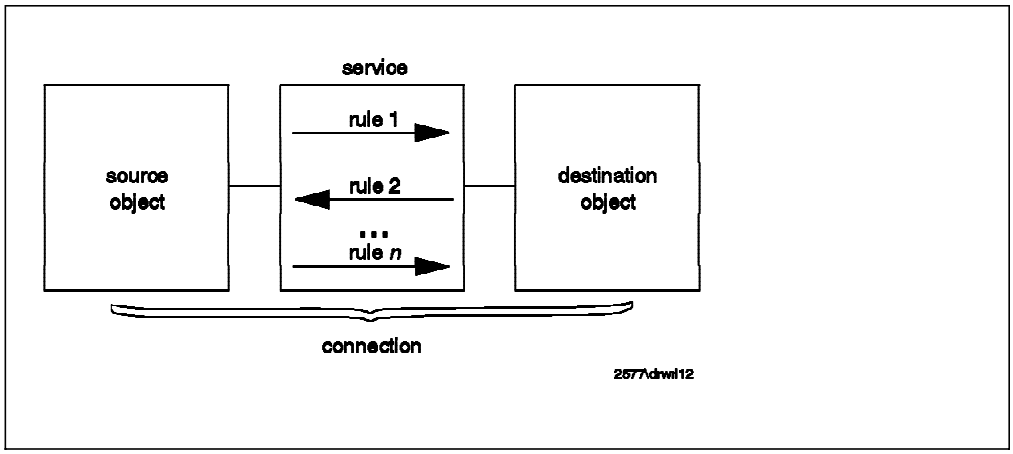


Figure 30. Connection

For example, imagine you have a connection that permits Telnet between a client in the secure network and the proxy server on the firewall. The service in this case is Telnet. To be precise, it is a session from an unprivileged client port to TCP port 23. The source object in this case is any IP address in the secure network, and the destination object is the firewall.

We normally think of a connection definition as something that permits a connection to happen. However they can also be defined to block the defined service. It is most important that the firewall *only* allows the services for which you have permit connections defined. To ensure this happens, it applies a default “block everything” rule to any packet that does not match any rule in a connection definition. Objects, groups and services are described in detail in the following paragraphs.

5.1.2 Objects and Groups

An object is a representation of a network component. It is defined by an IP address and an address mask, so it is possible for one object to represent a whole range of network addresses. A group is a collection of one or more objects. Possible objects are:

- Host: A node in your network with mask 255.255.255.255.
- Network: A set of IP Addresses with a specific mask.
- Firewall: The firewall interface with mask 255.255.255.255.
- Router: A unique IP address with mask 255.255.255.255.
- Interface: A network adapter with mask 255.255.255.255.
- VPN (Virtual Private Network): Network outside of the tunnel.
- User: A remote client without IP address or subnet mask defined.

When you want to define a new object or group, you must select the option **Network Object**, at the left of the main page, of the GUI (see Figure 24 on page 45). For convenience, you can also enter the dialog directly if you find you need to add a new source or destination object when creating a new connection.

The variables in an object definition are:

Table 1. Object Field Meanings

Description	**Meaning**
Object Type	One of the possible objects named previously.
Object Name	This is the name of the object. You should try to apply a name convention to keywords used in this name to make it easier to find them again within a list. Don't use the pipe symbol], simple quote ' or double quote ".
Description	This describes the object.
User Name	Use this if you select object type **user**.
Filter Lifetime	The lifetime of the object rule.
IP Address	The specific IP Address or the range of IP addresses for this object.
Subnet Mask	Depends on the type of object you are defining. The subnet mask automatically changes, but you can override it if needed.

The only default object is The World, an object that is matched by any IP address.

5.1.3 Services

A service defines the type of IP traffic that is permitted or denied between a source and destination object. For example, you could construct a service to permit Telnet, or a service to deny Ping.

A service is built of one or more rules. IBM Firewall provides you with a large collection of commonly required rules and when building a service you can usually find the rules you need predefined. If you don't find the rule that you need, you have to create an extra rule before you define the service. You also have the ability to move rules up or down in the service, to create a specific order in the rules.

In the service configuration window you have the following fields:

Table 2 (Page 1 of 2). Service Field Meanings

Description	**Meaning**
Service Name	This is the name of the service. As for objects, you should use a name convention for keywords used in this name, to simplify searching.
Description	This describes the function of the service.

Table 2 (Page 2 of 2). Service Field Meanings	
Description	**Meaning**
Service Composition	You must add the rules that you need for this service, and you can move rules up or down to establish the correct order of the rules in the service. Order may be important, because some rules contained in a service may be more restrictive than others. If the less restrictive rule is at the top of the rule list, the packet may never be tested against the more restrictive rule. The other element of the service composition section is the *flow button*. This defines whether the rule applies for packets going from the source to the destination object, or to returning packets (those going from destination to source). Very often a service contains an even number of rules, in pairs, with one of the pair controlling the flow in one direction and the other controlling the reverse direction.
Override Log Control	Has the value *yes*, *no* or *no override*. No override (the default) will let the settings in the rules apply. If you select yes a log record will be generated for every packet that matches the rule, regardless of the log control setting in the rules that make up the service. This is useful for debugging, but you will not normally want to log so intensively. If you select no, no log records will be generated regardless of the settings in the rules.
Override Frag. Control	Allows you to override fragmentation settings in the rules. • **no override** - default, rule settings apply • **yes** - match any IP packet • **no** - match only non-fragment packets • **only** - match only non-fragment headers and fragments without a header, do not match non-fragments • **headers** - match only non-fragments and fragment headers, do not match non-fragments
Override Tunnel ID	If you enter a tunnel ID into this field the session will be passed through the specified secure tunnel.
Time Controls	With this feature you can activate and deactivate the service dependent on the time, date or day of week. You have the following fields available: • Control By Time of Day: begin and end time selection in a day. • Control By Days: begin and end day selection for days of the week, or calender days. • Time Control Action: specify if service is active/inactive during specified period.

When you configure a service you do not specify the objects (that is, network addresses) between which it operates; you define the objects when you place the service in a connection definition. However, you do need to know what type of objects a rule applies to, because you have to define the direction of flow for each rule within the service definition. For example, a service that defines a TCP session from a client to a proxy server on the firewall will only operate as intended if it is included in a connection whose destination object is a firewall IP address.

The flow is indicated by an arrow at the left of the rules, as shown in Figure 31 on page 57.

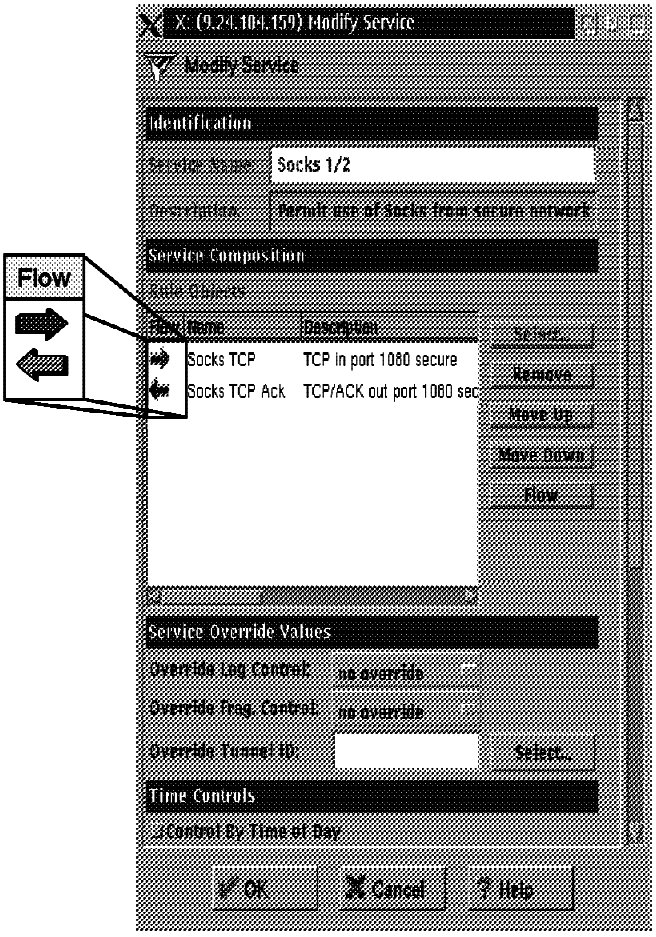


Figure 31. The Flow Indicator

If the rule has a green arrow (arrow points to the right) the filter defined by the rule applies to packets flowing from the source to the destination object. If the rule has a blue arrow (arrow points to the left) the source object and destination object are swapped, so the rule applies to flows from destination to source. For example:

Table 3. Green Arrow	
Source Object	The World
Destination Object	Secure Network
Rule	Permit tcp gt 1023 eq 80
Resulting Filter	Permit The World Secure Network tcp gt 1023 eq 80

Table 4. Blue Arrow	
Source Object	The World
Destination Object	Secure Network
Rule	Permit tcp gt 1023 eq 80
Resulting Filter	Permit Secure Network The World tcp gt 1023 eq 80

5.1.4 Rules

We have already discussed how rules are combined within services which are, in turn, embedded within connection definitions. Let us now look at rules in more detail.

There are two types of rules: deny rules and permit rules. They define which IP packets are blocked (denied) or allowed to pass (permitted) at which firewall interface. The packets can be filtered on all characteristics described in Chapter 3, “Getting to Grips with IP Packets” on page 23. The following fields must be entered when defining a rule:

Table 5 (Page 1 of 2). Filter Rule Field Meanings

Description	Meaning
Rule Name	This is the name of the rule. Use a name convention for keywords used in this name, so rule names will look consistent. This makes it easier to search for them within a list.
Description	This describes the function of the rule.
Action	Has the value *permit* or *deny*. Any IP packet that matches the other fields in the filter definition will either be passed or blocked depending on the value of this field. You can also specify a protocol by number or name.
Source Port/ ICMP Type	The first field specifies the type of operation, the second the desired port number (for ICMP packets it is the ICMP Type of the message). The port operation field is an arithmetic operator field which can have values of: any, equal to, not equal to, less than, greater than, less than or equal to, greater than or equal to. The operator is applied to the desired port field, so, for example, if the two fields were *greater than 1023*, we would only match packets with a source port number of 1024 or higher.
Destination Port/ ICMP Code	This pair of fields is used in the same way as the source port fields to define which destination port(s) we want the filter to match. For ICMP packets, it refers to the ICMP Code field.
Interfaces Settings	This defines which interface the packet is flowing through: • secure • nonsecure • specific (a specific interface can be defined, for example, when there is more than one secure interface) • both (the rule applies when the packet flows through any of the firewall interfaces)
Direction/ Control	In some cases the firewall may act as a router, in which case packets flow through it. In other cases the packets may go to an application on the firewall machine itself (such as a proxy server). This field defines whether the packet has a destination or source of the firewall, or whether the destination and source are both addresses other than the firewall (in which case the firewall is behaving as an IP router). Possible values are: • local (coming to or from the firewall itself) • route (going through the firewall) • both (the rule applies in all cases)

Table 5 (Page 2 of 2). Filter Rule Field Meanings	
Description	**Meaning**
Direction	Defines whether the packet is coming into or going out of the adapter where the rule is applied. Remember that the rule can apply at any of the firewall adapters, controlled by the Interface definition (above). Possible values for the direction are: • inbound • outbound • both (rule applies whichever way the packet is going)
Log Control	This field defines if the packet should be logged or not. The default log control setting for permitted packets (those that pass the rule) is *no* and for denied packets is *yes*. It is important to log intensively on a firewall, because you cannot tell in advance which piece of seemingly unimportant log data will reveal an attack. However, logging every successfully transmitted packet is usually more than you need. Options are: • no • yes
Fragmentation Control	The possibilities are: • yes - matches header, fragments and nonfragmented packets • no - matches only nonfragmented packets • only - matches only fragment headers and fragments without a header • headers - match only non-fragments and fragment headers, do not match non-fragments
Tunnel ID	Identifies the tunnel through which the packet must be sent.

5.1.5 Names and Descriptions Convention

In this section we explain the reasons why we suggest you should use a convention when you define a name for a new object, rule, service or connection and its description.

Three characteristics are important when you design a name or description:

- Identify

  When you add a new definition (connection, service, rule or object) to the rule base it joins the large list of pre-loaded definitions. You need to be able to identify easily the purpose and function of the new definition. A short name and a good description can help you to do it.

- Reuse

  When you need to define a connection or service, it is most easy to create this service using existing rule templates. So if you create the rules with easy names, you can more easily select the appropriate rules for new services.

- Change

  Reuse of definitions is certainly a good thing, but caution is needed. Sometimes it may be better if you create a new rule for a special service,

instead of re-using an existing rule, because when you change a rule used by multiple services, the change will affect all the services, which may not be what you want.

We give some suggested rules for a naming convention which you may want to follow, or use as a basis for your own. For the names we use only 2 or 3 words and for the description we explain the name better. We place the most descriptive word first in the name, so that all of the definitions that are related to each other appear together when they are presented in an alphabetical list. For example:

Table 6. Names and Description Convention

	Name	**Description**
Object	Firewall secure	This is the secure firewall interface
Rule	TCP-80 permit inbound	Permit TCP port 80 secure inbound
Service and Connections	HTTP outbound permit	Permit HTTP from secure network to nonsecure network

Remember don't use the pipe symbol |, quote ', or double quote " in the names and description. Also, remember that you can see any of the rule elements as fields in the rules list; you just need to click the button labelled with the name of the field that you need (so, for example, in Figure 39 on page 66 we could add the *Directon* field to the rule list display by clicking on the button at the top of the list).

5.2 Rule Base Design

To set up the rule base for your firewall in a structured way it is important that you have a clear picture of your network infrastructure and the services that you want to provide. In this way it is easy to configure your firewall and maintain a consistent set of connections, objects, services and rules.

When you want to implement your connections there are basically two possible types of connections. The first type, and most easy to implement, is a standard connection. This connection can be built with predefined services. The following services are predefined:

Telnet
FTP
HTTP
SOCKS
SSL
SMTP
RealAudio
Identd
SNMP
Ping
DNS
SecureID
Tunnels
Remote Logging
Firewall Configuration

The other type of connection cannot be built of predefined services and you must define your own service instead. When defining the service you will probably have to define your own rules as well. Note that some protocols have more than one predefined service and that sometimes a predefined service does not contain the exact rules you need.

We will first describe how to set up a standard connection (one made up of predefined services and rules) and after that a non-standard connection.

5.2.1 Standard Connections

In this part we explain how to define a connection in the IBM Firewall. The example we use is a Telnet connection between the secure network and the firewall secure interface. This is something you are very likely to need to set up, to allow an administrator to log in to the firewall for maintenance purposes.

First, start the configuration GUI using the fwconfig command and then select the connection option from the navigator pane on the left of the display. A list of existing connections will appear, as shown in Figure 32. In the list select **NEW** and click on **Open** to create a new connection.

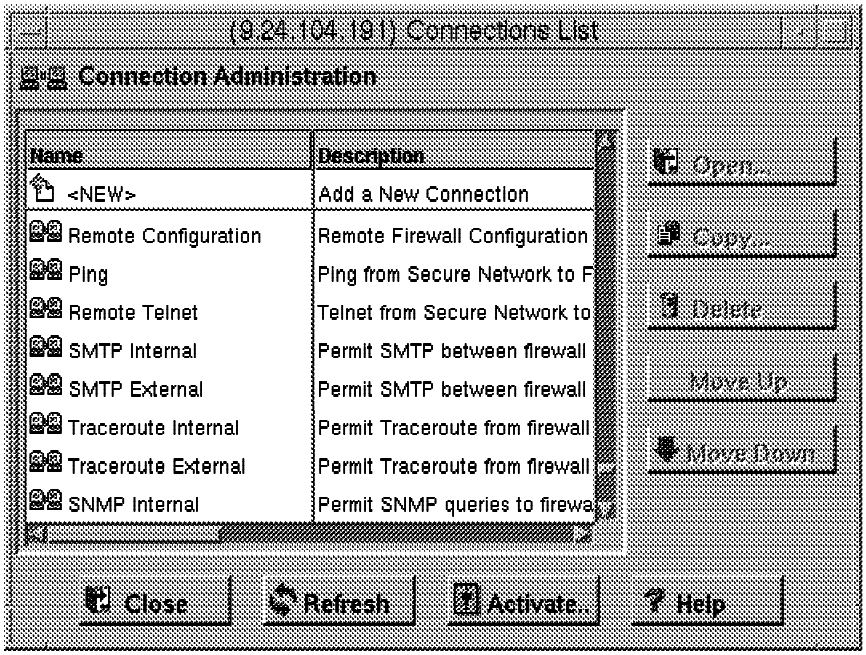


Figure 32. Connection List

Figure 33 on page 62 shows the new connection screen, where you must define all the parameters. First, enter the name and the description of this connection. Remember to use a convention for all the names, as this will make future definitions and modifications easier.

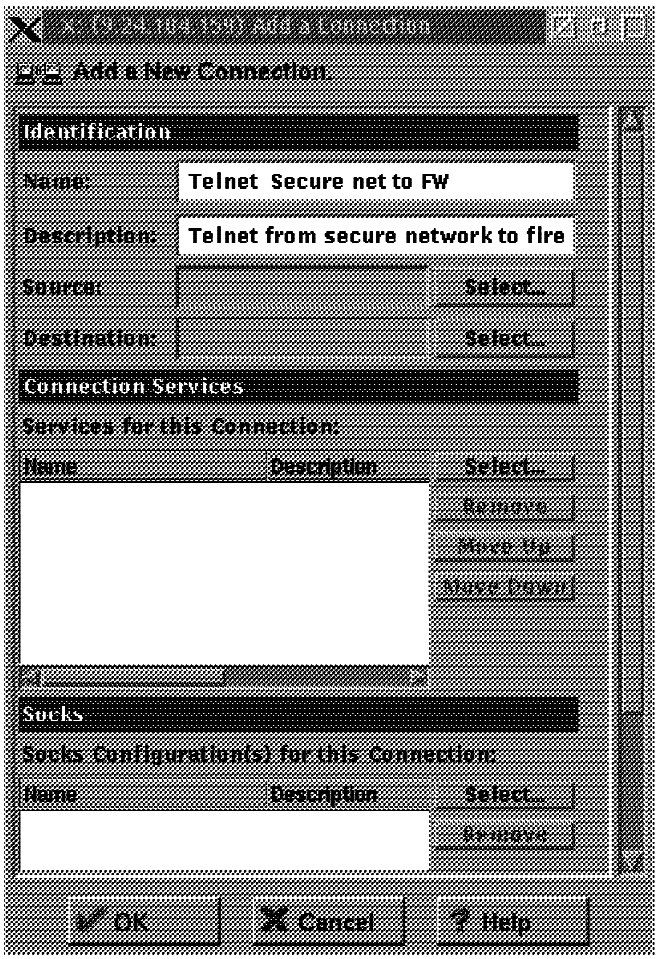


Figure 33. Add New Connection

Secondly you need to define the source object and destination object. Click on **Select** to select each one from the object list. If you have not already defined the object, you can select **New** to define it. The only object predefined is The World, so we will have to define both the source and destination objects to construct our example. Figure 34 on page 63 shows the definition for our source object, representing any address in the secure network. After you have defined the object click **OK**, then select the new object in the object list and click **OK** to place the object in the source object field. This procedure must be repeated for the destination object, the firewall itself. Figure 35 on page 63 shows our definition for this object.

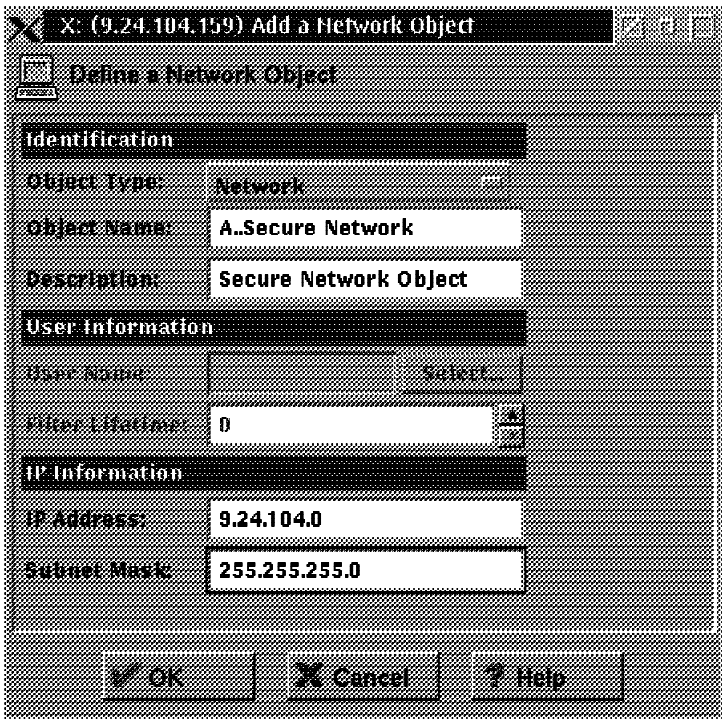


Figure 34. Source Object

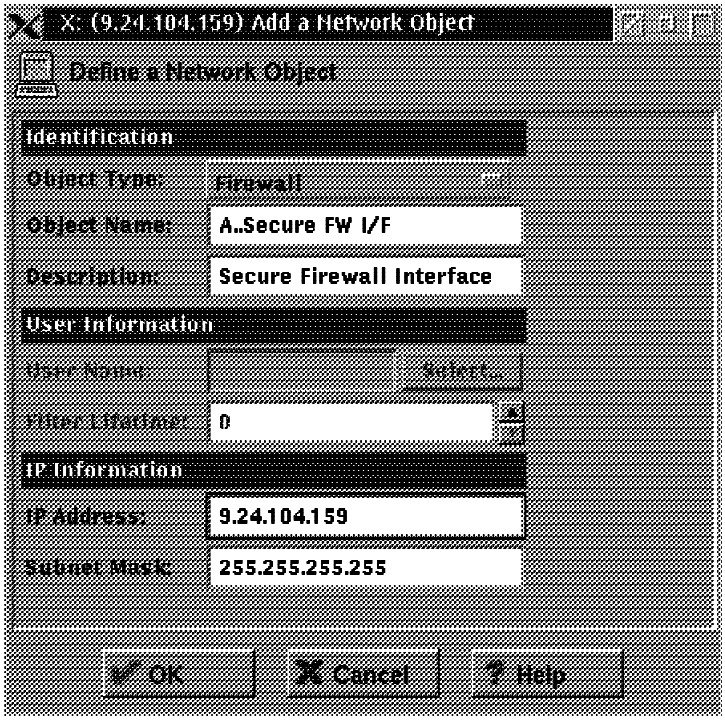


Figure 35. Destination Object

Finally you need to select the service between these objects. Click on **Select** and a list of all the defined services will appear. In this case we are using a standard service, so select the line named "Permit Proxy Telnet Outbound" and click on **OK**. Figure 36 on page 64 shows the services list.

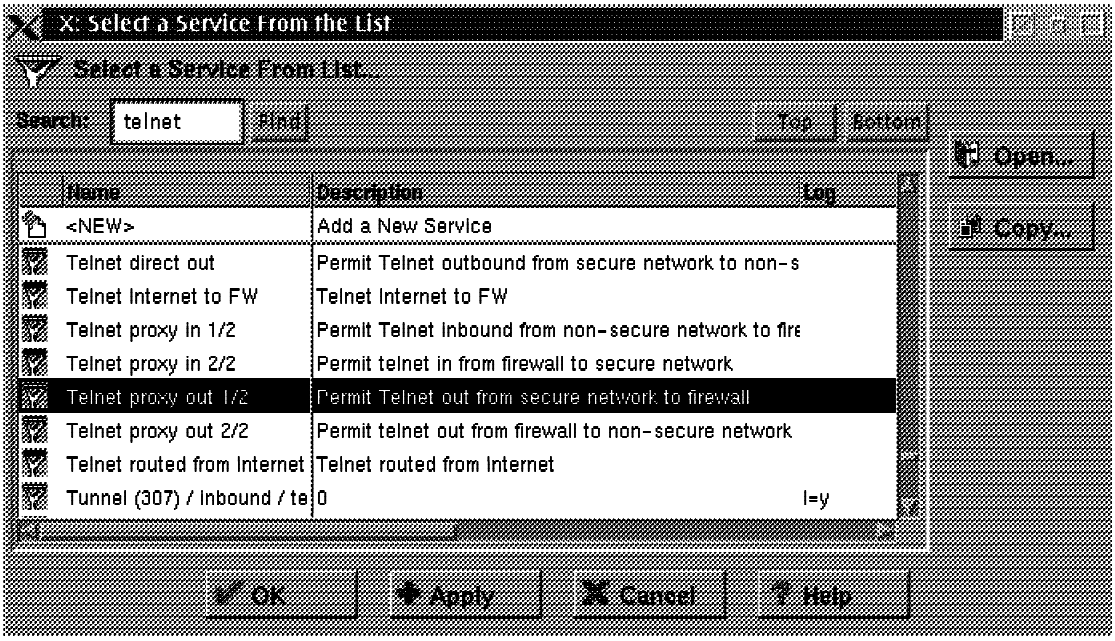


Figure 36. Services List

Figure 37 on page 65 shows the final result. Click on **OK** to save the connection definition.

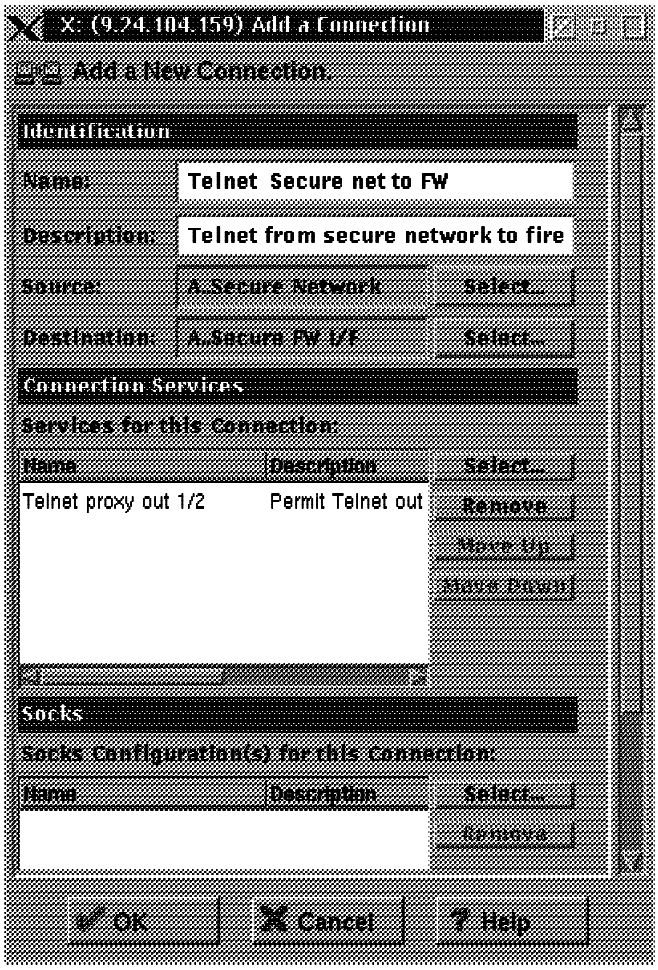


Figure 37. Connection

Now you need to activate the rule base and the filters file will be rebuilt. We describe this in 5.2.3, “Rule Base Activation” on page 69.

5.2.2 Non-Standard Connections

A non-standard connection is one that cannot be built from predefined services. For example, imagine you have a new application (we call it "CUST") which has a proxy server running on your firewall. It listens on TCP port 400 and you want to be able to access it from the secure network. This is visualized in Figure 38 on page 66.

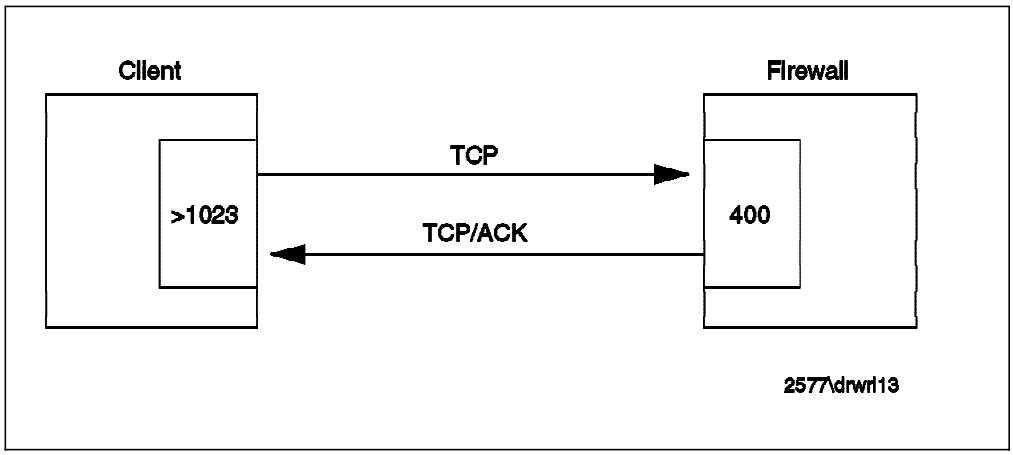


Figure 38. Non-Standard Connection

To be able to build this connection we are going to create a service that is called: "Permit CUST". First, we have to decide whether we need new rules for this service. Therefore you have to know which rules already exist, by checking the list of rules. Do this by selecting **Traffic Control** then **Connection Templates**, and **Rules** from the initial GUI navigator panel (see Figure 24 on page 45).

In this example we need a rule that permits inbound TCP packets on port 400 of the secure interface. This does not exist so we must create a new rule.

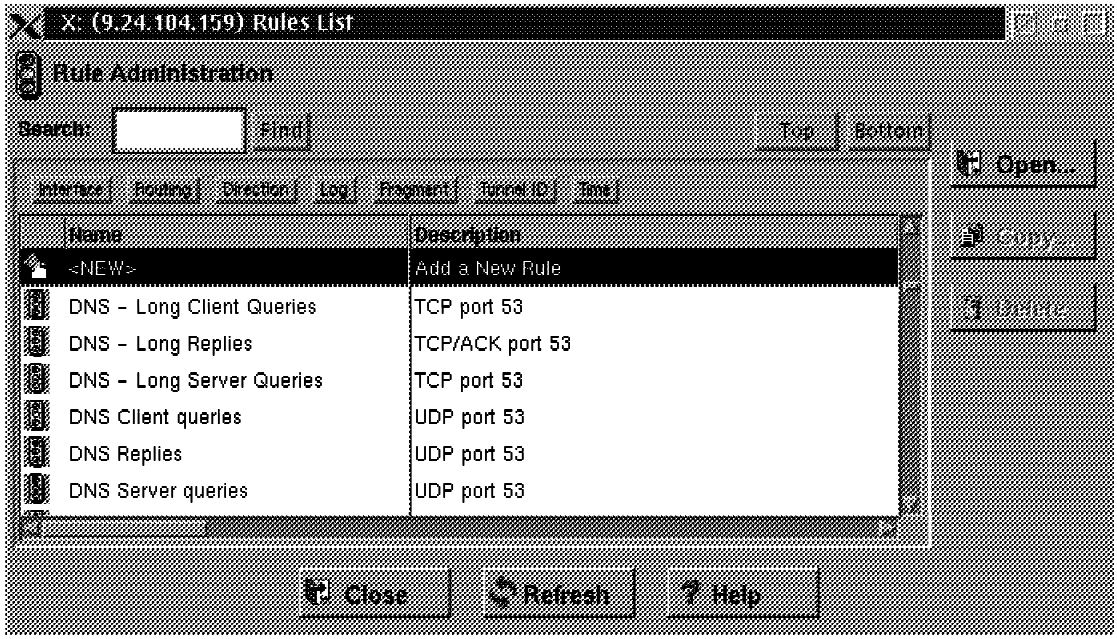


Figure 39. Define a New Rule

It is very important to assign a clear name to a rule. For example, do not use the name of a source or destination in the rule name, because they are independent of the rule. A good name may be: "Permit CUST Inbound 1". By giving rules clear names, it is also easier to reuse your rules. In the rule list double click on **<NEW>** as shown in Figure 39. Fill in the parameters for the new rule, as shown in Figure 40 on page 67. Notice that we have been very specific in defining the rule: it will only allow packets for our CUST application to pass if they appear inbound on the secure side of the firewall.

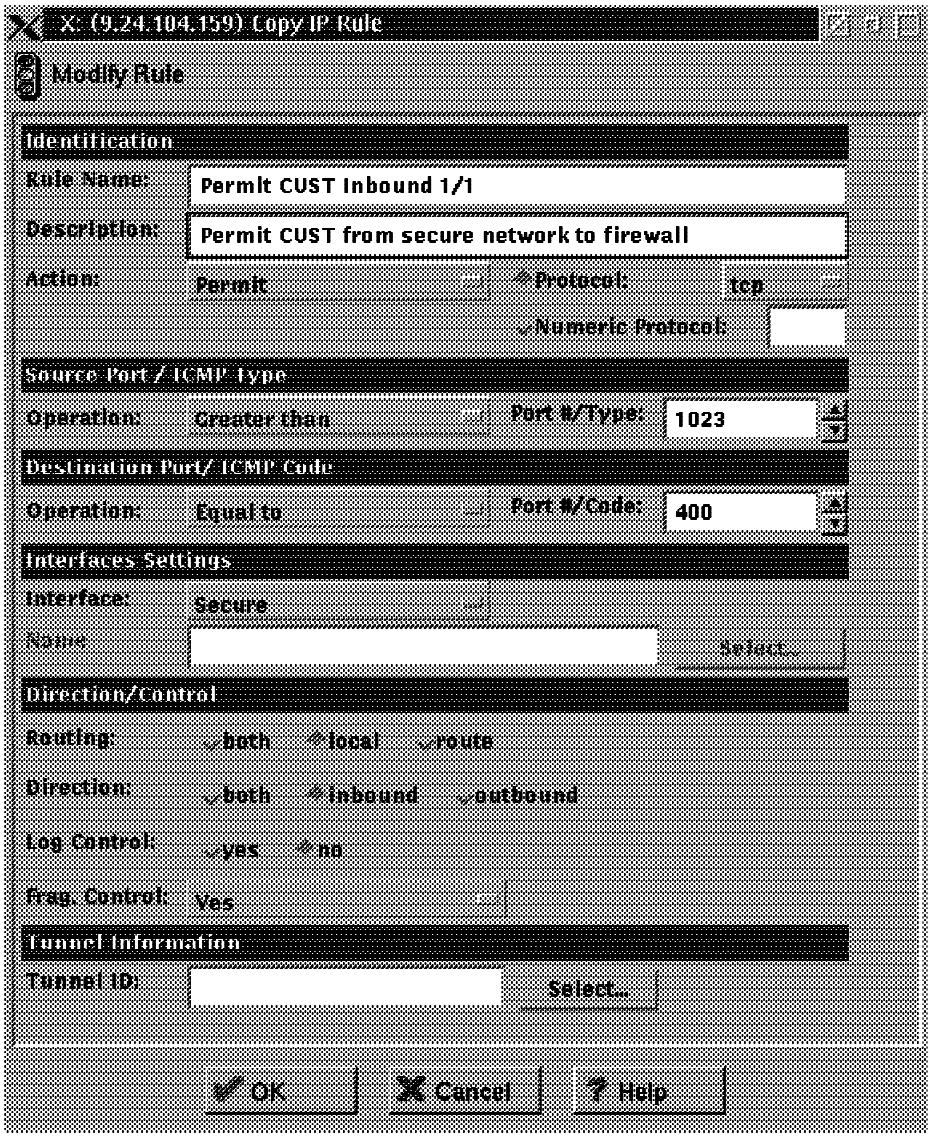


Figure 40. Parameters for "Permit CUST Inbound 1/2"

This rule only deals with one direction, client to server. We also need to create a rule for the response packets from the server to the client. The construction of this rule is visualized in Figure 41 on page 68. The differences from the first rule are:

- The protocol is now TCP/ACK.
- Source and destination criteria are swapped.
- The direction is now outbound.

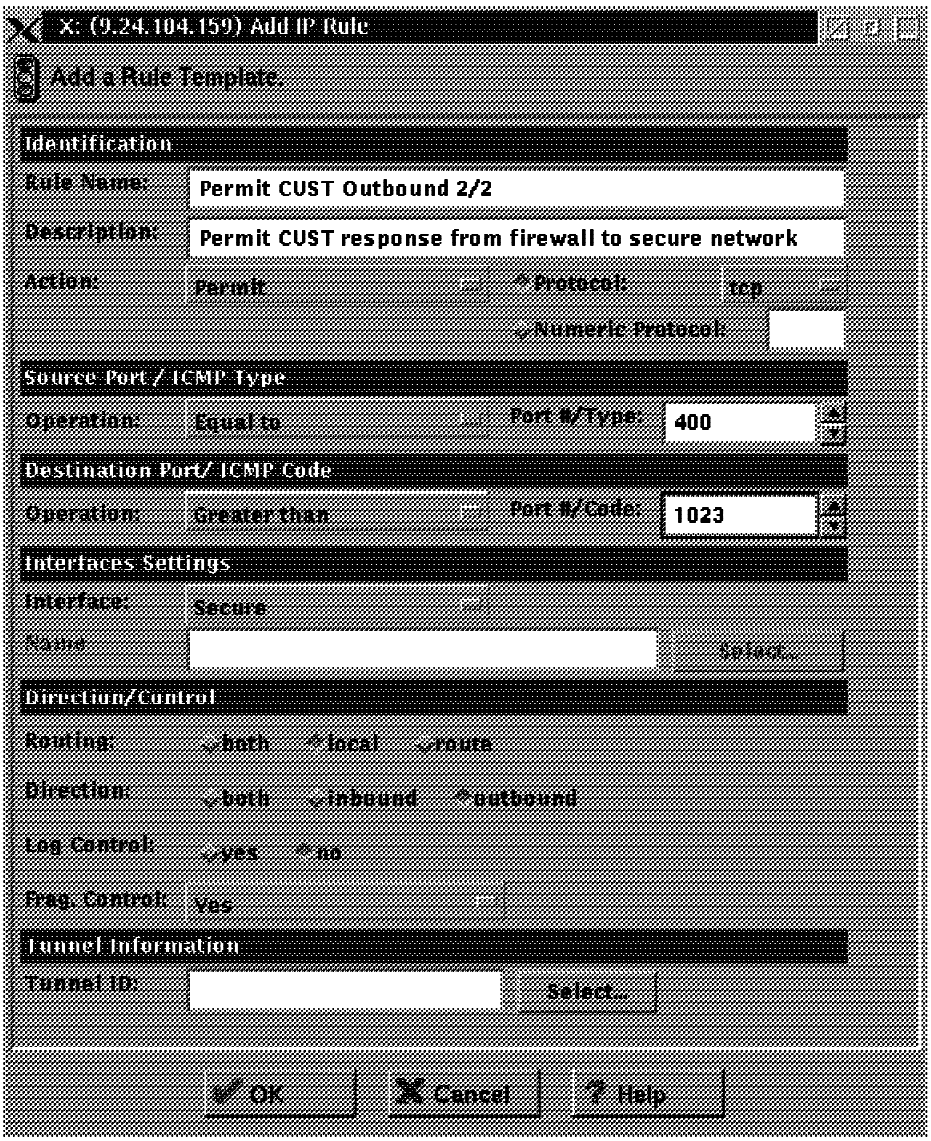


Figure 41. Parameters for "Permit CUST Outbound 2/2"

After creating this rule we can build the "Permit CUST" service that invokes the new rules. Select **Traffic Control** then **Connection Templates**, and **Services**. from the initial GUI navigator pane (see Figure 24 on page 45). You will see the list of existing services. In the list double click on **<NEW>** to see the new service dialog screen shown in Figure 42 on page 69.

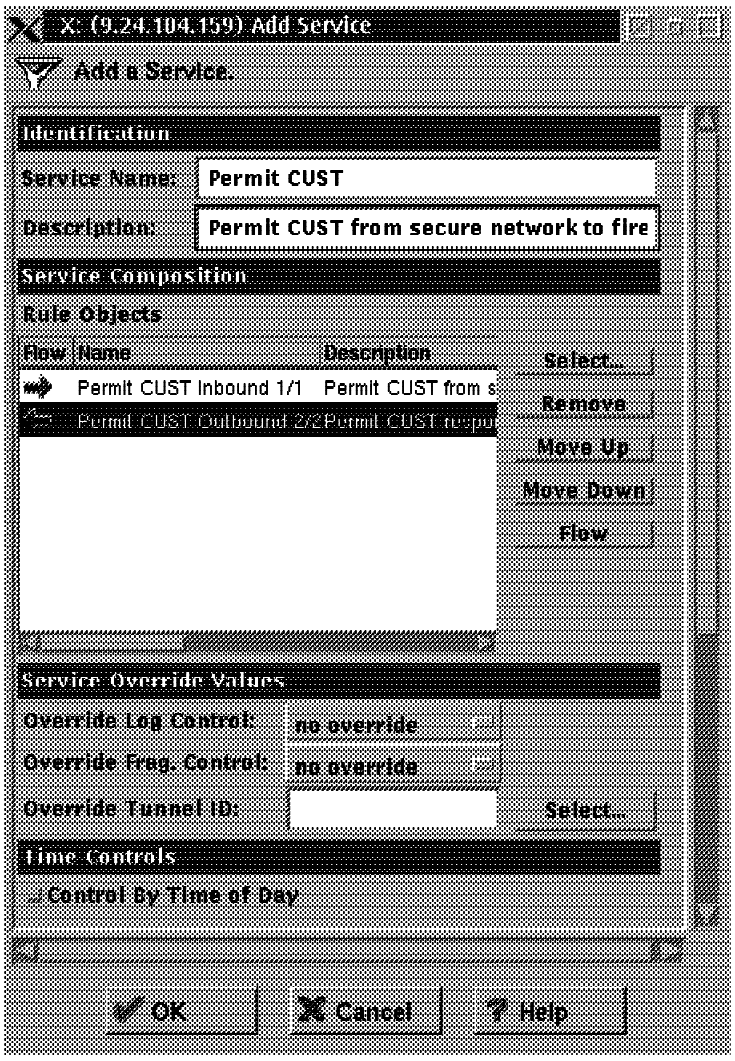


Figure 42. Create Service "Permit CUST"

Notice that the flow of the second rule has to be changed, because it applies to packets in the reverse direction (destination to source).

Finally, we can configure a connection, in the same way as for the previous example (see 5.2.1, “Standard Connections” on page 61) with the following content:

- Source object *Secure Network* (created in the previous example).
- Destination object *Secure Firewall* (also created in the previous example).
- Service *Permit CUST*.

5.2.3 Rule Base Activation

After you have defined the rule base you have to activate it. This will also create the filters file that IBM Firewall uses to control its packet filtering function (/etc/security/fwfilters.cfg).

Figure 43 on page 70 shows the options in the rule base activation window:

- Regenerate Connection Rules and Activate. If you choose this option the filters file will be created, using your definitions of connections, objects,

rules, services and socks. Packet filtering will be done according to the filters file. You cannot edit the filters file manually, because every time that you activate the rule base, the file fwfilters.cfg is overwritten.

- Deactivate Connection Rules. If you choose this option your rule base will be deactivated and packet filtering will be disabled.

  Note: The firewall will not route packets if the connection rules are disabled.

- List Current Connection Rules. If you choose this option you will see the content of the filters file /etc/security/fwfilters.cfg.
- Validate Rule Generation. If you choose this option the syntax of the rules in the rule base will be validated.
- Enable Connection Rules Logging. Choose this option to enable logging.
- Disable Connection Rules Logging. Choose this option if you don't want logging. This is not recommended!

Figure 43 shows the activation screen.

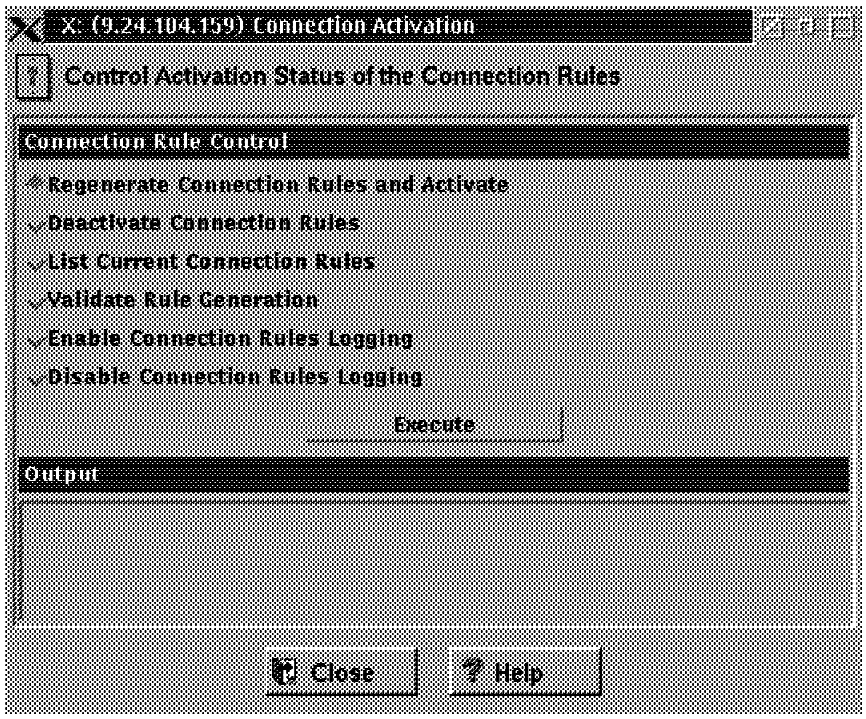


Figure 43. Connection Activation

Chapter 6. Examples of Rules for Specific Services

The objective of an Internet connection is to provide access to *services*:

- To allow your users to access services in the Internet
- To allow Internet users access to services that you provide

In most cases a firewall is configured to allow access to a combination of services of different types. In this section we will analyze some of the different services that you will want to provide, such as terminal emulation, file transfer or World Wide Web. We will consider the most important alternative ways to deliver them and give examples of rules to control them.

For a more extensive discussion about all the possible services, we recommend referring to *Building Internet Firewalls* (Chapman and Zwicky).

6.1 What Services Should You Provide?

One of the most important points is to decide which services you will provide. You should provide only those services that your users *need*, not the ones that they merely *want*. This point cannot be stressed enough. A service should be provided only if there is a business requirement, not if it's a "nice to have" feature.

It is good practice to use deny rules at the end of each section of rules that permit a given service, instead of relying on the implicit default deny everything rule (see 5.1.1, “Connections” on page 54). This will prevent problems that may arise if the rules file contains a later misconfigured permit rule.

In most of the examples below, we show two diagrams. The first is a schematic diagram of the network connection we are trying to achieve. The second represents the combination of rules, services and objects that make up the connection definition in the firewall rule base.

The schematic diagrams contain IP address information that is specific to a given installation. In order to make the samples in this chapter as generic as possible, we have substituted symbolic notations for IP addresses, masks and functions. The following list summarizes these notations:

s.s.s.s	Secure network IP address
sm.sm.sm.sm	Network mask of secure network
s.s.s.1	IP address of firewall secure interface
n.n.n.1	IP address of nonsecure firewall interface
M.M.M.M	Internal mail server IP address
SN	Abbreviation for secure network
NSN	Abbrevation for nonsecure network
N.V.6.K	Network manager (using SNMP)
S.Y.S.L	syslog host (destination of logging)
D.N.S.I	Internal DNS
N.S.E.R	Internal news server
N.F.E.E	External news feeder
p.p.p.p	Proxy
PPP1, PPP2, etc	Generic port numbers

The connection diagrams use a convention to illustrate the relationship between the different rule base components, as shown in Figure 44 on page 72.

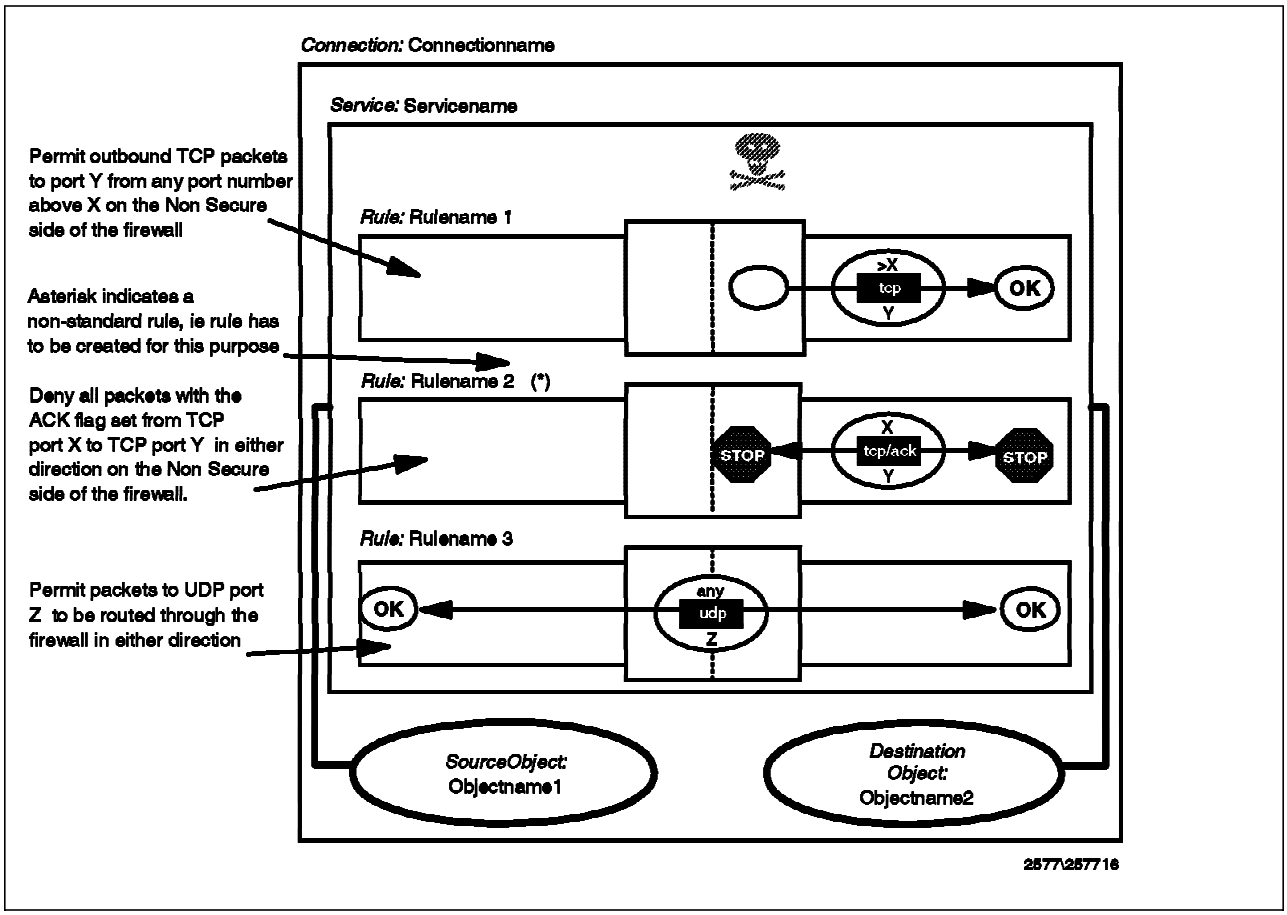


Figure 44. Key to the Connection Diagrams

6.2 Connection Rules that Should Always Be Present

No matter what services you are offering, you should always start your rule base with a set of rules that provide general protection, by explicitly denying several types of attack:

- Rules to block attempts at IP address spoofing
- Rules to control ICMP message flow
- A rule to isolate private networks from the Internet
- A rule to protect the SOCKS service on the nonsecure interface
- A rule to protect the Syslog server on the nonsecure interface
- A rule to protect from misuse of the loopback network
- A rule to block access to the system resource controller

You may also wish to place other rules near the top of the rule base, for example, to control broadcasts or prevent routed traffic. We will consider each of these in turn.

6.2.1 Rules to Block Attempts at IP Address Spoofing

In an IP address spoofing attack, the attacker impersonates the IP address of another machine. This is typically used to attack services that rely on IP addresses for user authentication. This attack is different from the source-routing attack, although many people confuse them. In both cases the attacker sends packets that appear to come from a trusted machine. The difference is that in attacks which subvert IP routing, the attacker expects also to receive packets that are intended for the trusted machine, whereas in the IP spoofing attack, the attacker sends packets that are harmful enough to do damage without the need for a direct reply. In this case, the attacker doesn't need to receive any IP packet; it is enough to be able to send packets.

A classic attack of this type is the TCP sequence number prediction attack. In this attack, the attacker does not need to receive any packet from the attacked host. It needs to be able to send packets and it also has to predict the TCP sequence number that will be used by the attacked machine in its replies to the impersonated machine (so that it can acknowledge them).

Figure 45 on page 74 illustrates this. In this example, we have a truster machine that trusts a machine with IP address 9.24.104.241. The attacker wants to send packets with the trusted address as source, instead of its own. To do this the attacker has to do the following three things:

1. It must somehow cause the real trusted machine to not respond to the truster. Any denial of service attack could be used, including the Ping 'O Death and SYN flooding attacks. Alternatively the attacker could just wait until the machine was down for some legitimate reason.
2. Next it sends TCP/IP packets to the truster, constructed with a source address of the trusted in the header. The truster will respond in the normal way, believing that it is receiving requests from the real trusted.
3. The responses have nowhere to go, and normally the timeout processing on truster would detect that trusted was not respondindg. In order to maintain the illusion, therefore, the attacker must be able to predict the sequence numbers of the packets that the truster sends. It can then fabricate requests and responses which appear to acknowledge the truster's messages.

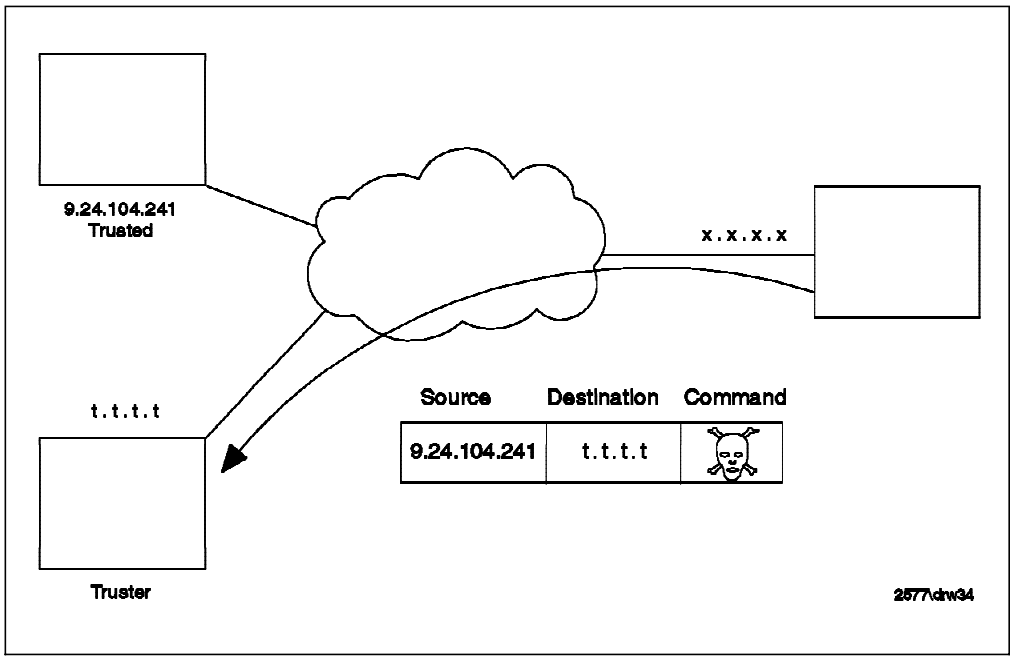


Figure 45. IP Spoofing Attack

To protect your network from address spoofing and routing corruption attacks, you should add a couple of connections to reject packets that have a source address within the secure network, but which appear on the nonsecure interface of the firewall (see *Actually Useful Internet Security Techniques* by Larry Hughes for a more complete discussion).

The following connections implement this policy. These should be the first two connections in your rule base.

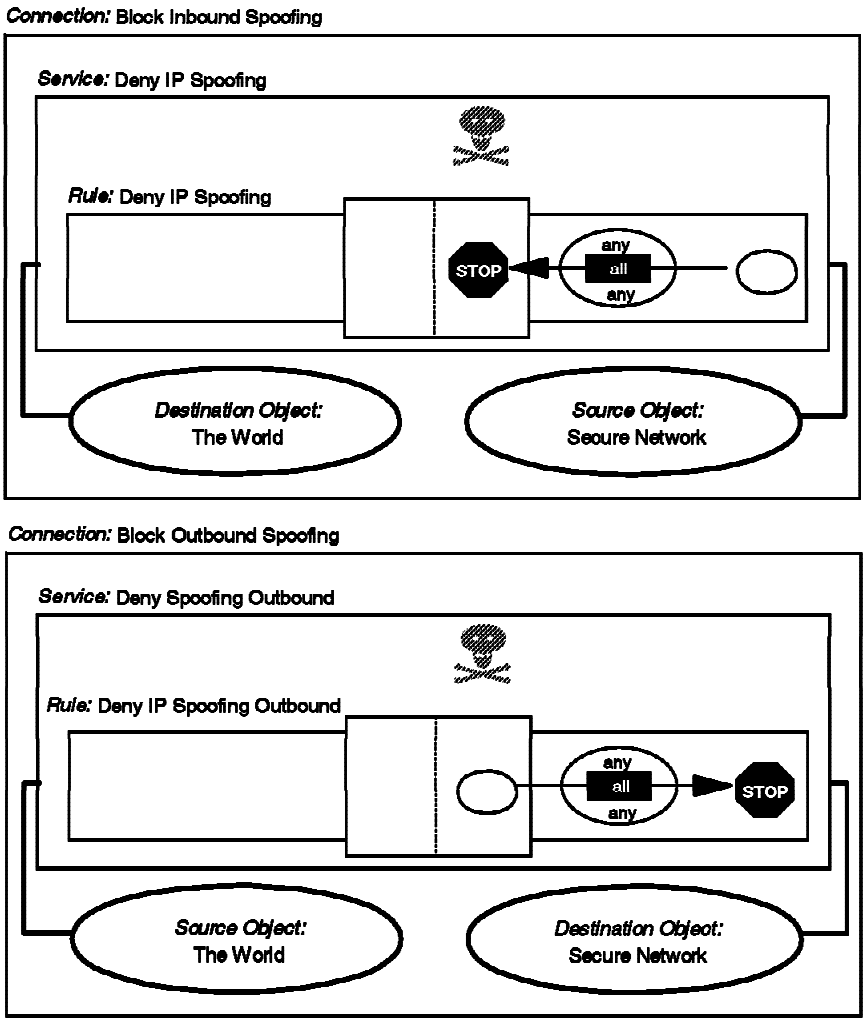


Figure 46. Connection Definitions to Prevent IP Spoofing

The two rules used in these connections are not part of the standard rule base, so you have to create them and the two services that use them.

The filter definitions resulting from these connections are:

```
deny s.s.s.s sm.sm.sm.sm 0 0 all any 0 any 0 nonsecure both inbound
deny 0 0 s.s.s.s sm.sm.sm.sm all any 0 any 0 nonsecure both outbound
```

6.2.2 Rules to Control ICMP Message Flow

The simplest thing to do is to block all ICMP messages from crossing the firewall. A simple connection to do this would be as follows:

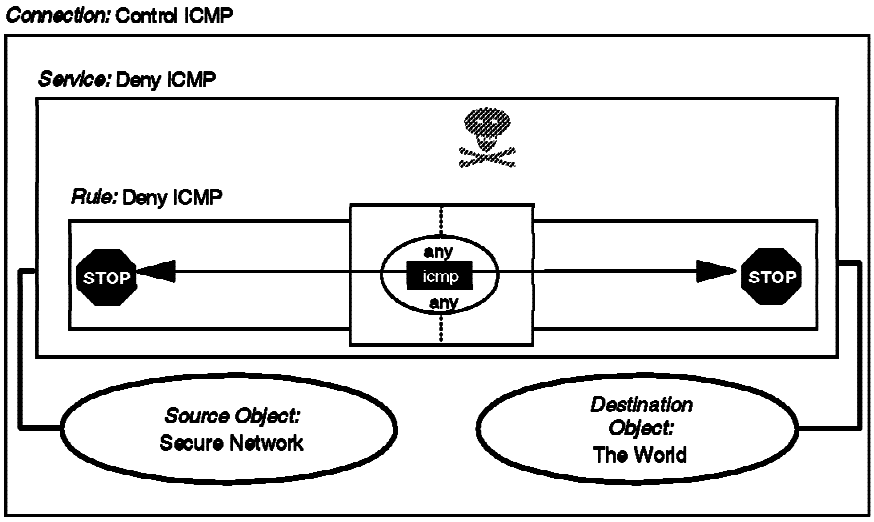


Figure 47. Connection Definition to Block All ICMP Messages

However, if you want to be more selective about which ICMP messages you want to allow to flow, you may want to add the connections described in 6.20, "Filtering Specific ICMP Messages" on page 130.

The filter definition resulting from this connection is:

```
deny 0 0 0 0 icmp any 0 any 0 both both both
```

6.2.3 Rule to Isolate Private Networks from the Internet

RFC 1597 defines ranges of IP addresses that are reserved for private, isolated networks. What this means is that the address ranges can be used within an organization, but they can never be one end of a session that crosses the Internet. The Internet backbone routers are configured not to route them.

You should implement the following connections to prevent these addresses from leaking into or out of your network:

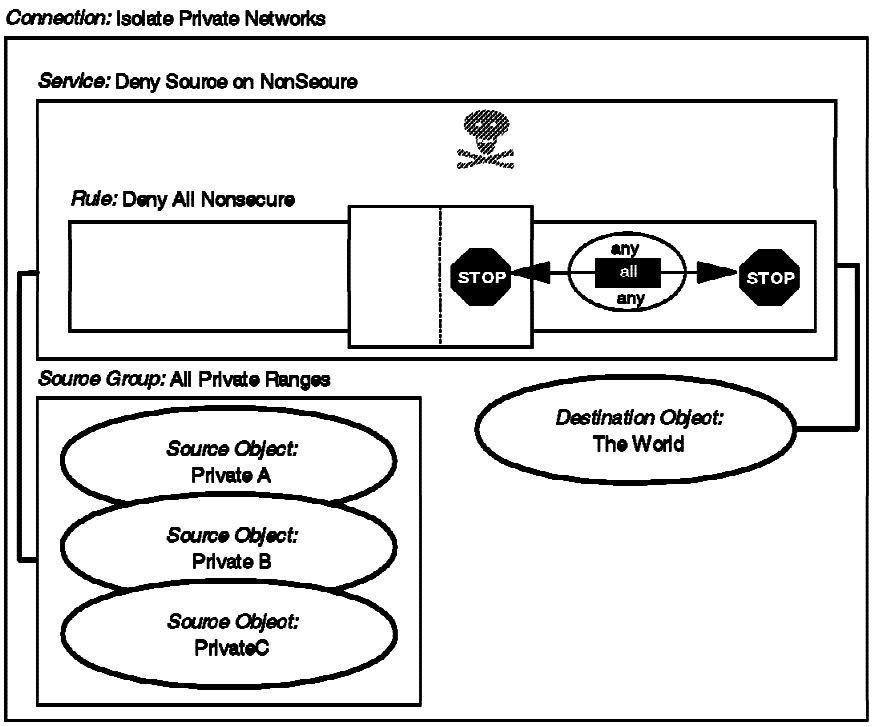


Figure 48. Connection Definition to Control RFC1597 Addresses

The three network objects in this connection, which are collected together in the *All Private Ranges* group are all the nodes in the ranges defined in RFC1597. That is to say, the following:

```
Private A:  Address = 10.0.0.0    Mask = 255.0.0.0
Private B:  Address = 172.16.0.0  Mask = 255.240.0.0
Private C:  Address = 192.168.0.0 Mask = 255.255.0.0
```

You may think that you do not have these addresses in your network, but in practice they are often used for testing purposes, so they may appear without warning.

The filter definitions resulting from this connection are:

```
deny 10.0.0.0    0xff000000 0 0 all any 0 any 0 nonsecure both both
deny 172.16.0.0  0xfff00000 0 0 all any 0 any 0 nonsecure both both
deny 192.168.0.0 0xffff0000 0 0 all any 0 any 0 nonsecure both both
```

6.2.4 Rule to Protect the SOCKS Service on the Nonsecure Interface

The SOCKS server normally only operates inside-out. That is, it provides a facility to allow secure network clients to access nonsecure servers, but not vice-versa. You should therefore place an explicit connection that prevents a nonsecure node from connecting to it:

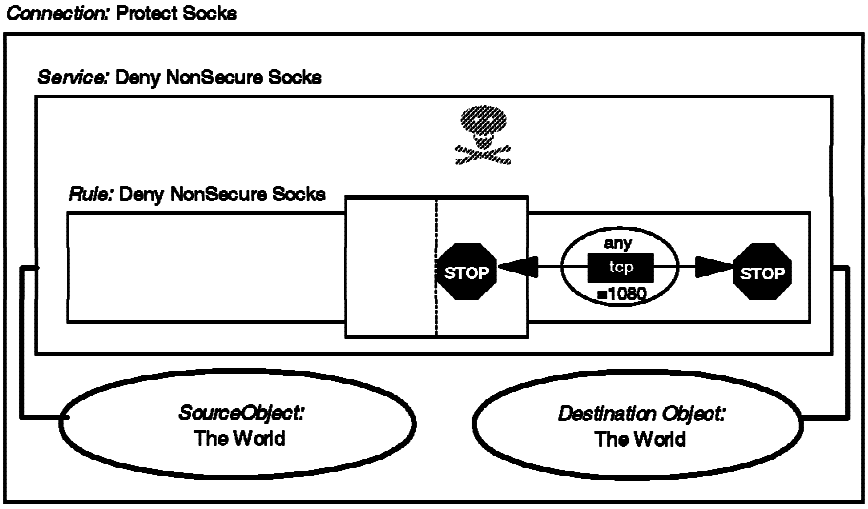


Figure 49. Connection Definitions to Protect SOCKS

The filter definition resulting from this connection is:

```
deny 0 0 0 0 tcp any 0 eq 1080 nonsecure both both
```

This rule can also be set by using the Security Policy panel (see 6.23, “Using Security Policy” on page 134).

6.2.5 Rule to Protect the Syslog Server on the Nonsecure Interface

Syslog has a facility for logging to a remote machine, using UDP port 514. The following connection prevents a nonsecure machine from attempting to connect to the server:

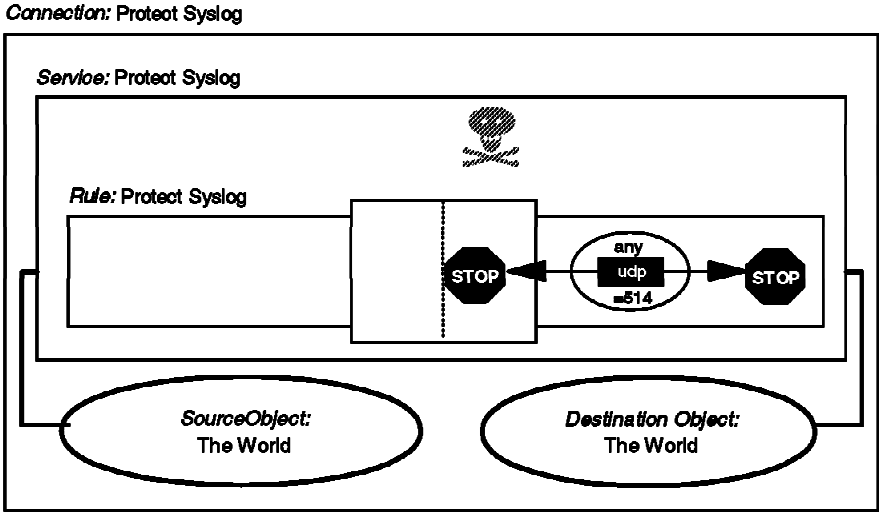


Figure 50. Connection Definition to Protect Syslog

The filter definition resulting from this connection is:

```
deny 0 0 0 0 udp any 0 eq 514 nonsecure both both
```

6.2.6 Rules to Protect From Loopback Network

Loopback is a logical IP interface that IP uses for internal communications. The loopback addresses should never appear on any real network interface. The following connection makes sure that they don't:

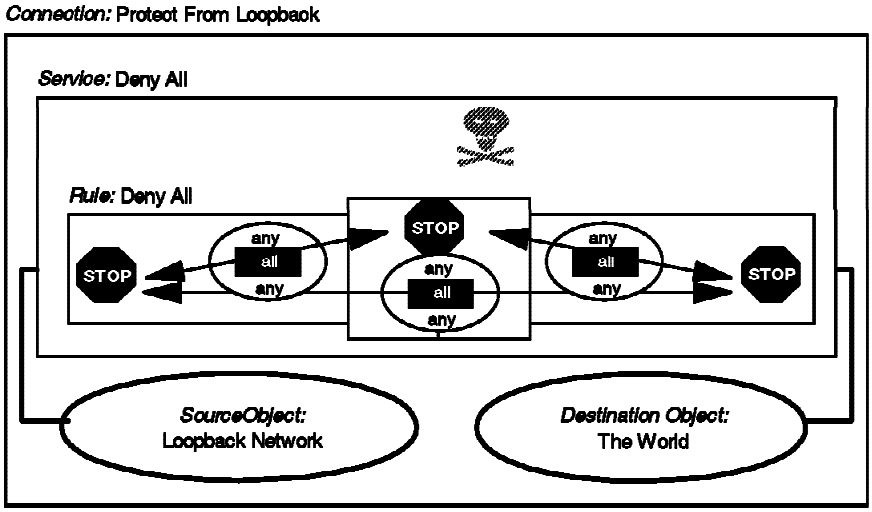


Figure 51. Connection Definition to Protect Loopback

The filter definition resulting from this connection is:

```
deny 127.0.0.0 0xff000000 0 0 all any 0 any 0 both both both
```

6.2.7 System Resource Controller

The system resource controller is used by AIX to control the resources of the system. It uses UDP port 200, so it should be blocked from outside access as it is a point that can be used to compromise your firewall (for example, by starting or stopping a service).

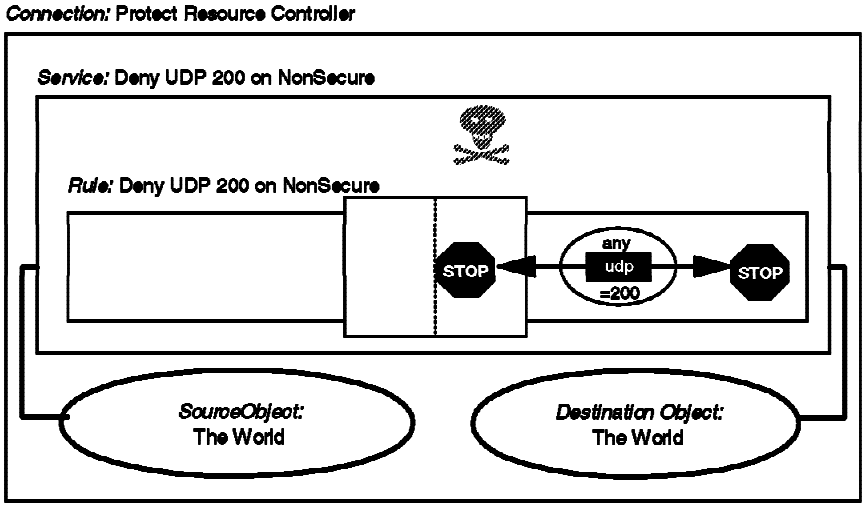


Figure 52. Connection Definition to Protect SRC

The filter definition resulting from this connection is:

```
deny 0 0 0 0 udp any 0 eq 200 nonsecure both inbound
```

6.2.8 Broadcast

Broadcasts should be blocked in order to reduce network traffic on the firewall. You also should be aware of multicast traffic, for example, the *all host multicast* address (224.0.0.1). See RFC 1112 *Requirements for Internet Hosts -- Communications Layers* for information about IGMP (Internet Group Management Protocol) and multicasting. Denying broadcast and multicast packets can cause excessive logging, so it is recommended to switch logging off for these "denial services".

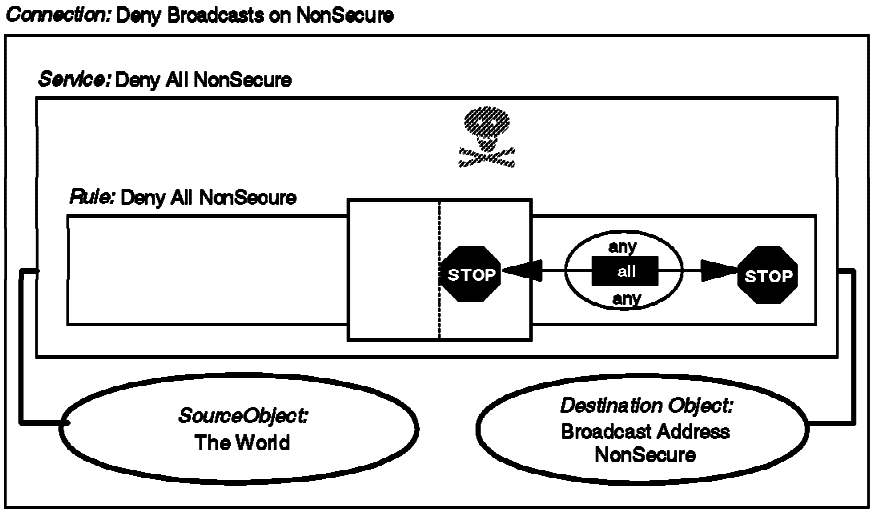


Figure 53. Connection Definition to Block Broadcasts

A similar connection can be implemented to reduce traffic on the secure interface.

The filter definitions resulting from this connection are:

```
deny 0 0 224.0.0.1 0xffffffff all any 0 any 0 nonsecure both both
deny 0 0 255.255.255.255 0xffffffff all any 0 any 0 nonsecure both both
```

The second rule only stops broadcast to 255.255.255.255; you may want to modify it for your specific case. You should use the inverse of the subnet mask used for the nonsecure interface as the broadcast address to stop.

Similar rules can also be set by using the Security Policy panel (see 6.23, “Using Security Policy” on page 134). The rule created when you check on "Deny braodcast to non-secure interface on the Security Policy panel is:

```
deny 0 0 0.0.0.255 0.0.0.255 udp any 0 any non-secure both both
```

This rule works for broadcasts to class C addresses only.

6.2.9 Routed Traffic

If you are running a dual homed firewall (in this context a non-routing firewall), you should already be preventing the firewall from routing traffic with the `no -o ipforwarding=0` option in /etc/rc.net. However, it is a good idea to have a rule to block routed traffic as well, in case the IP forwarding option is activated by mistake. The following connection will also prevent routing.

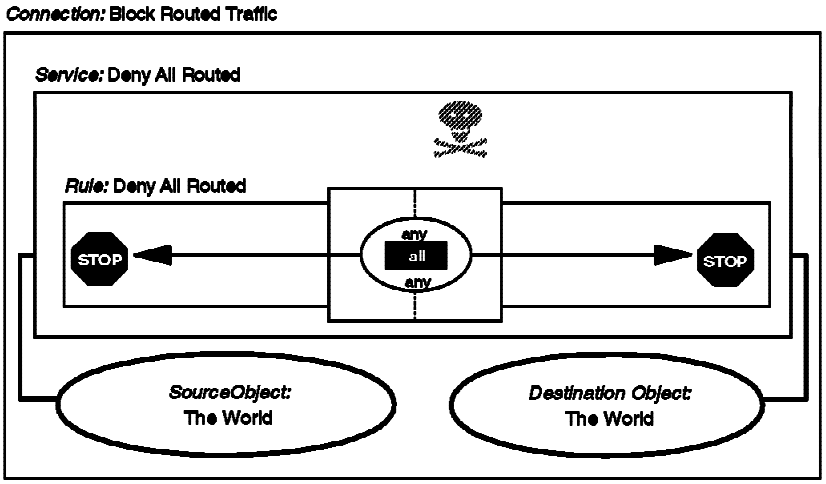


Figure 54. Prevent Routed Traffic

The filter definition resulting from this connection is:

```
deny 0 0 0 0 all any 0 any 0 both routed both
```

6.3 Telnet

Telnet is a protocol used to emulate terminal sessions. Telnet servers normally use TCP port 23, while the client uses one of the nonprivileged ports (starting from port 1024). Telnet uses passwords in order to authenticate the user. These passwords cross the network unencrypted.

The Telnet client may also be used to access other TCP-based services, for example, electronic mail (SMTP).

6.3.1.1 Possible Scenarios

We recommend you do not allow Telnet from the nonsecure network to the secure network. As Telnet sends the password unencrypted, an outsider can use a sniffer to grab passwords from the network and use them later. In the past there have been attacks to the main Internet nodes, in which attackers have installed sniffers in order to capture passwords (see CERT Advisory CA-95:18).

Even if you use one-time passwords, there are some serious security concerns. There have been attacks in which intruders anticipate a user connection being made. Once the user is authenticated the intruder hijacks the connection and starts to send its own packets (see CERT Advisory CA-95:01).

The only case in which you can safely allow incoming sessions is if you are using some encryption technique, such as the Secure IP Tunnel provided by IBM Firewall or the ones described in 6.22, “Secure Terminal Emulation” on page 133.

from Firewall to Secure Network:

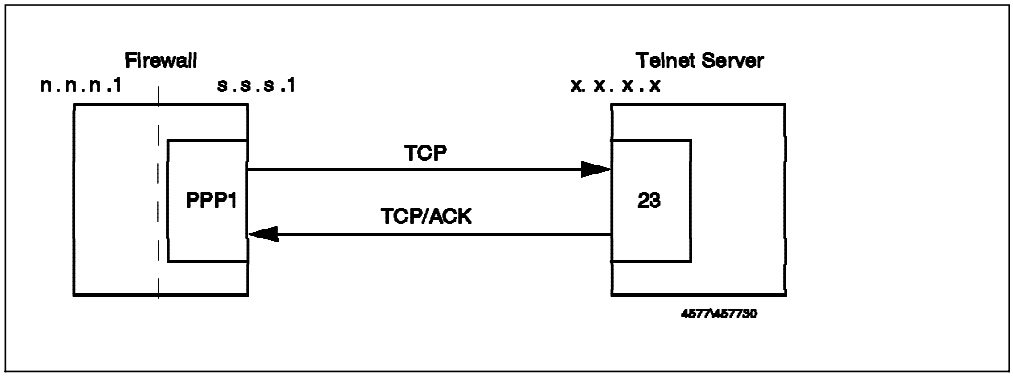


Figure 55. Telnet from Firewall to Secure Network

The following connection permits you to log in to a host in the secure network from the firewall, using Telnet. The first rule in the service allows traffic to be initiated by the firewall to any Telnet server in the secure network (TCP port 23). The second rule allows the Telnet Server (TCP port 23) to reply. The connection uses standard, predefined rules and service.

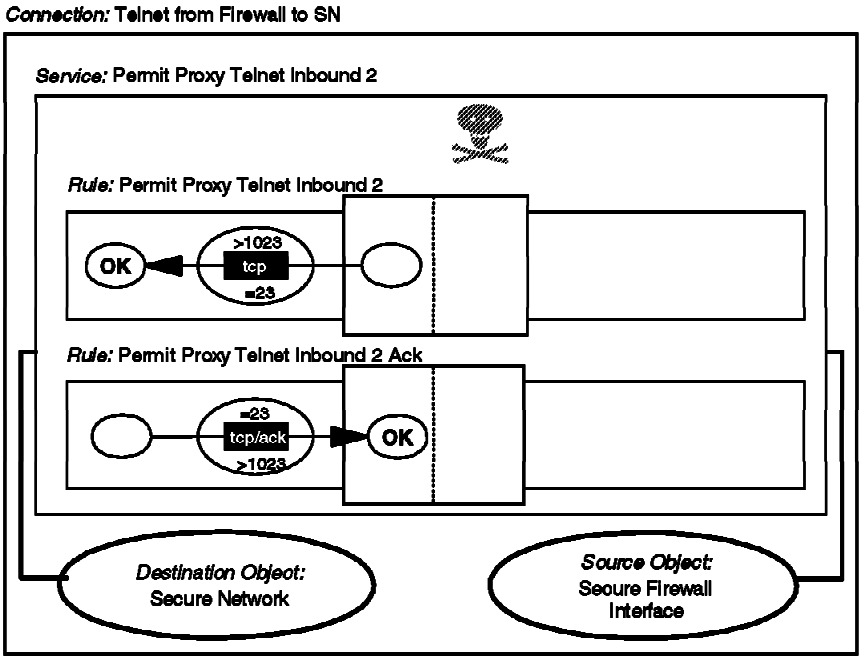


Figure 56. Connection Definitions for Telnet to Secure Network

This connection generates the following filter rules:

```
permit s.s.s.1 0xffffffff s.s.s.s sm.sm.sm.sm tcp     gt 1023 eq 23 secure local outbound
permit s.s.s.1 sm.sm.sm.sm s.s.s.1 0xffffffff tcp/ack eq 23 gt 1023 secure local inbound
```

Notice that we are using the tcp/ack format to prevent misuse of port 23 to establish a session (for example, a connection from port 23 to the SOCKS server in port 1080).

Telnet Using Proxy: Telnet with a proxy is a two-step connection. First the user logs into the firewall with a normal Telnet session. Once on the firewall, they use Telnet again to reach the final destination (which might be some other TCP

service, not necessarily Telnet on port tcp/23). The second log in may be entered explicitly or automatically. We will describe how to configure the Telnet and FTP proxy server in Chapter 8, “Configuring Proxy Services and SOCKS” on page 171.

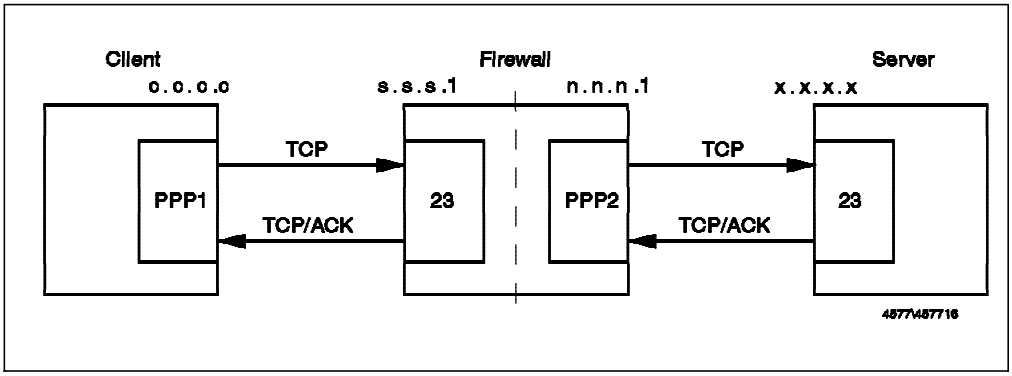


Figure 57. Telnet from Secure Network to Nonsecure Network Using Proxy

A connection definition can only reference one pair of network objects (source and destination). Because there are two parts to the session, one on either side of the firewall, we need to define two connections to describe it, as follows:

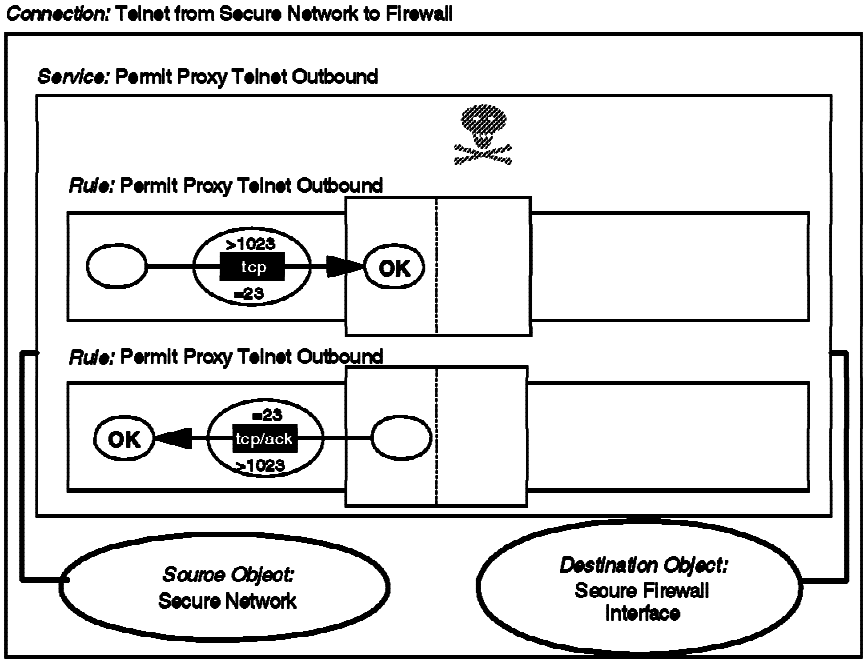


Figure 58. Connection from Secure Network to Proxy Telnet Server

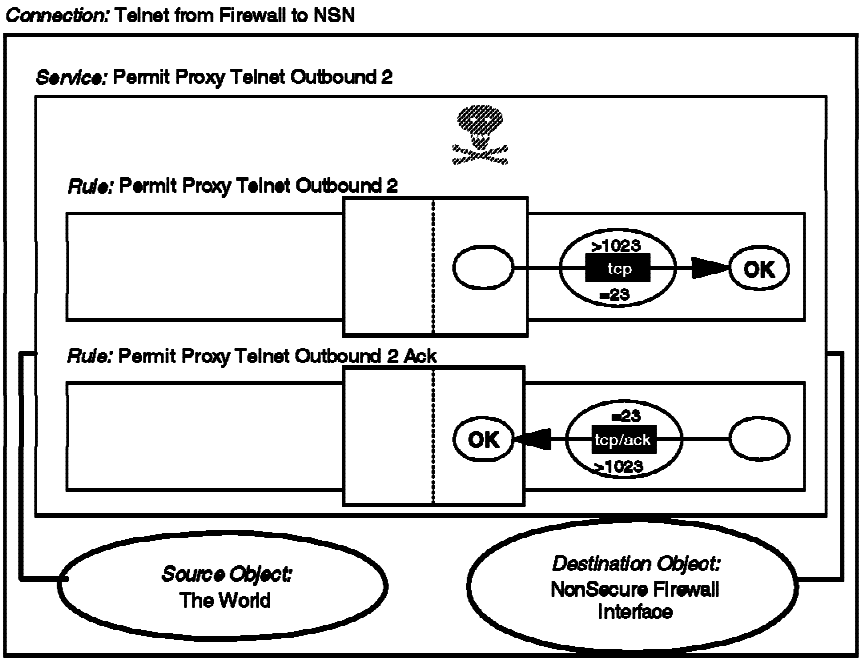


Figure 59. Connection from Proxy to Remote Telnet Server

These connections generate the following filter rules:

```
# Telnet from secure network to the Firewall
permit s.s.s.s sm.sm.sm.sm s.s.s.1 0xffffffff tcp     gt 1023 eq 23 secure local inbound
permit s.s.s.1 0xffffffff s.s.s.s sm.sm.sm.sm tcp/ack eq 23 gt 1023 secure local outbound

# Telnet from Firewall to the Nonsecure Network
permit n.n.n.1 0xffffffff 0 0 tcp     gt 1023 eq 23 nonsecure local outbound
permit 0 0 n.n.n.1 0xffffffff tcp/ack eq 23 gt 1023 nonsecure local inbound
```

Telnet Using SOCKS: Telnet using SOCKS is a three-step connection:

1. The user client connects to the SOCKS server on the firewall (using TCP port 1080).
2. The firewall connects to the identd server in the user's machine (this is an optional step, rarely used in practice).
3. The firewall establishes a connection to the final destination.

You will find a more detailed description of this process in 8.8, “Using the SOCKS Server” on page 190. As the ident authentication is optional, it is described in its own subsection (see 6.13, “ident” on page 116).

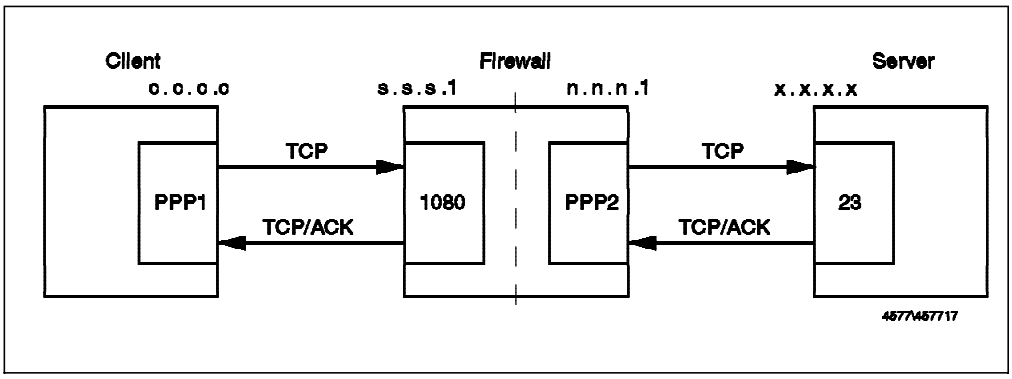


Figure 60. Telnet from Secure Network to Nonsecure Network Using SOCKS

The first part of the session is very similar to the proxy case, except using the SOCKS port instead of tcp/23, as follows:

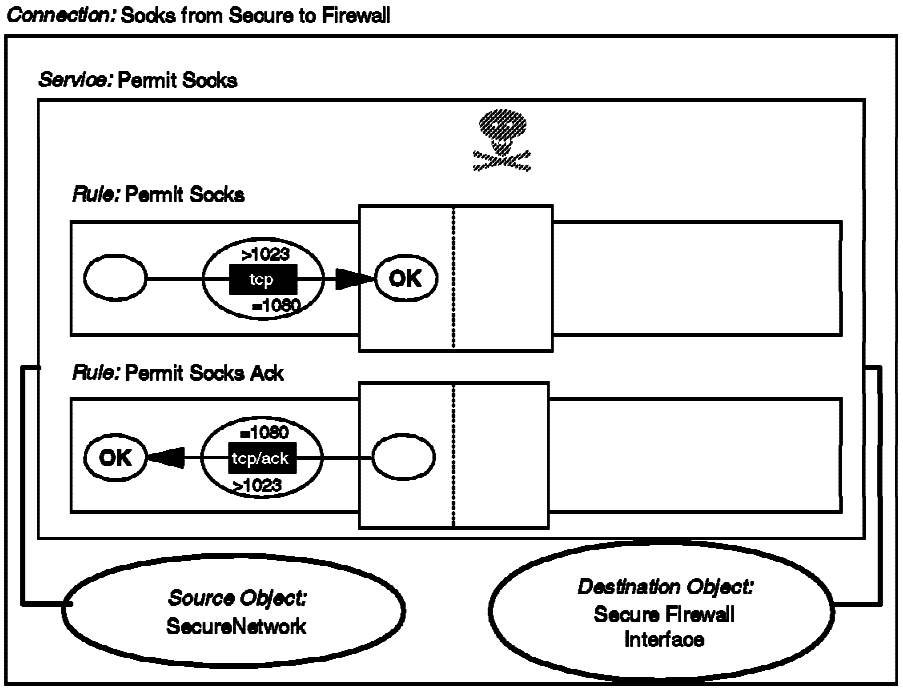


Figure 61. Connection from Secure Network to SOCKS Server

The second connection in the session is the same as Figure 59 on page 84.

These connections generate the following filter rules:

```
# Connection from the client to the Socks Server
permit s.s.s.s sm.sm.sm.sm s.s.s.1 0xffffffff tcp     gt 1023 eq 1080 secure local inbound
permit s.s.s.1 0xffffffff s.s.s.s sm.sm.sm.sm tcp/ack eq 1080 gt 1023 secure local outbound

# Telnet from Firewall to the Nonsecure Network
permit n.n.n.1 0xffffffff 0 0 tcp     gt 1023 eq 23 nonsecure local outbound
permit 0 0 n.n.n.1 0xffffffff tcp/ack eq 23 gt 1023 nonsecure local inbound
```

Telnet with IP Forwarding Enabled: We discourage you from allowing Telnet through your firewall without proxy or SOCKS to break the session and provide authentication. In general it is good practice to avoid allowing any routing through the firewall. That is, the best approach is to run it not as an IP router at all, but in a dual-homed mode.

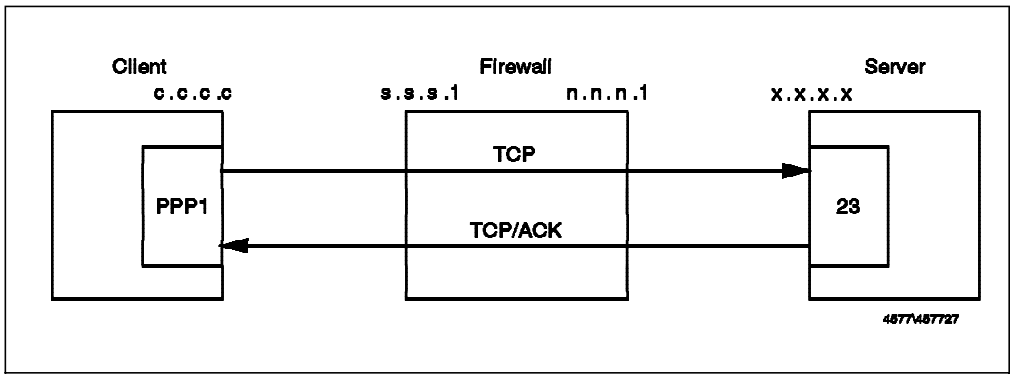


Figure 62. Telnet from Secure Network to Nonsecure Network, IP Forwarding

6.4 FTP. File Transfer Protocol

FTP also uses stream-oriented (TCP) sessions. However, it is more complicated than Telnet since it actually uses two different ports, one for the commands and the other for the data. It also has two different possibilities for establishing the connection, called normal-mode FTP (also called active-mode) and passive-mode FTP.

6.4.1.1 Normal Mode

The server is listening on tcp/21. The client, using a nonprivileged port, connects to the server establishing the control session. When the user enters a command like dir, get or put, the server, using port 20 (ftp-data), establishes a connection to a nonprivileged port of the client.

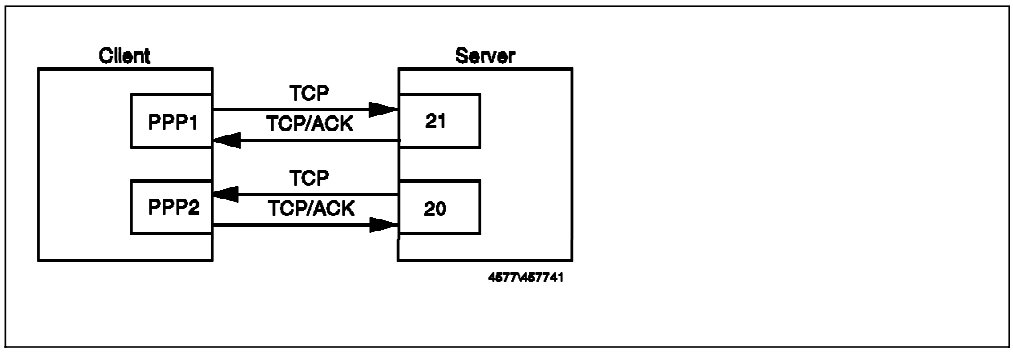


Figure 63. FTP Normal Mode

In this case there is an incoming connection that must be allowed from port 20 to an unknown port. This allows outsiders to misuse port 20 (see 3.1.4.1, “Source Porting” on page 34) so it is recommended to allow incoming connections only to the proxy server where you can limit the number of services that you provide.

6.4.1.2 Passive Mode

The server is listening on tcp/21. The client, using a nonprivileged port, connects to the server establishing the control session. When the user enters a command like dir, get or put, the client establishes a second connection from a nonprivileged port to a nonprivileged port of the server.

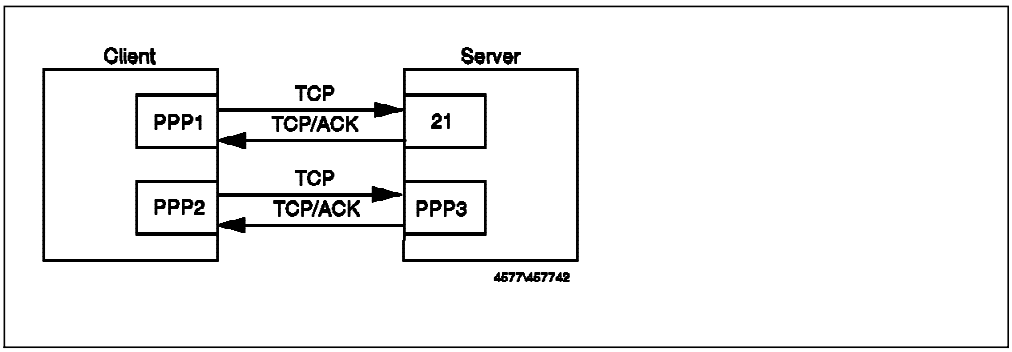


Figure 64. FTP Passive Mode

This avoids the problem of the incoming connection, but requires you to access nonfixed ports for the data channel connection (see RFC 1579, *Firewall Friendly FTP* for a complete discussion about FTP in a firewall context).

Normal Mode Routing, from Secure Network to Nonsecure Network: This cannot be allowed, as it requires incoming connections from TCP port 20, which would expose you to an attack by someone misusing the ftp-data port as a source port (see 3.1.4.1, “Source Porting” on page 34).

Normal Mode FTP from Firewall to Secure Network: The following connection allows the outbound control session to port 21 and the inbound data session from port 20. All of the rules within the connection are from the predefined set.

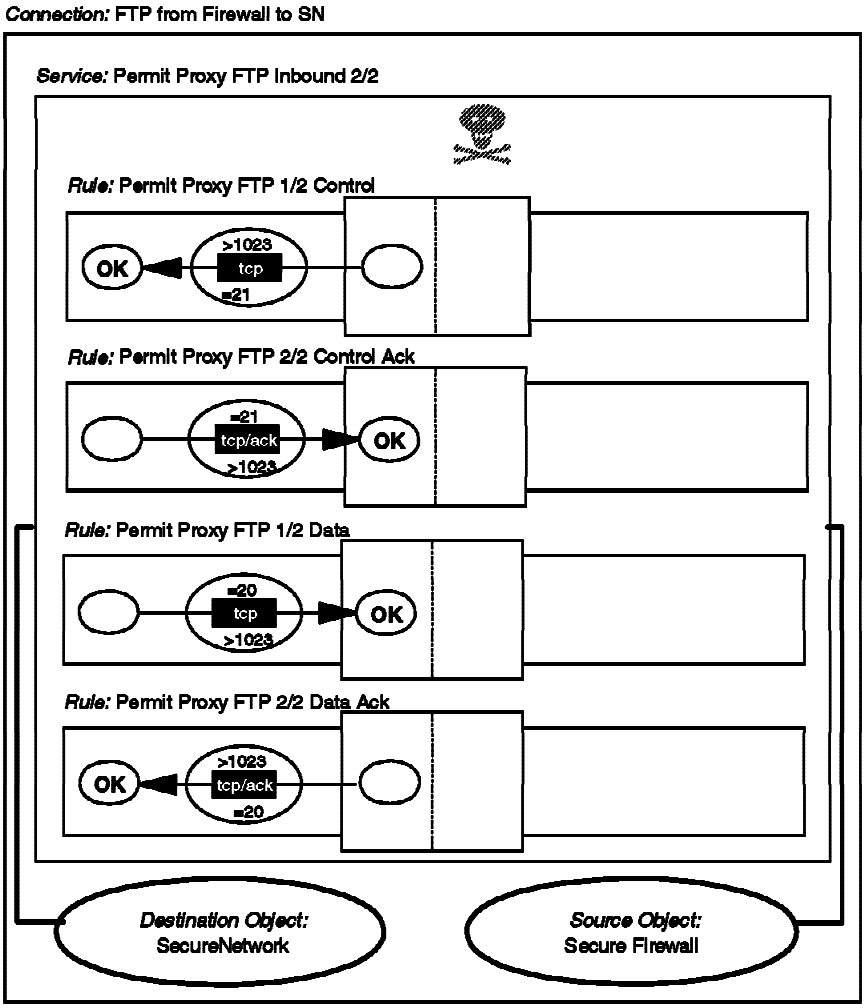


Figure 65. FTP from Firewall to Secure Network

```
# FTP From Firewall to SN. FTP Control Session
permit s.s.s.1 0xffffffff s.s.s.s sm.sm.sm.sm tcp     gt 1023 eq   21 secure local outbound
permit s.s.s.s sm.sm.sm.sm s.s.s.1 0xffffffff tcp/ack eq   21 gt 1023 secure local inbound

# FTP From Firewall to SN. FTP Data Session
permit s.s.s.s sm.sm.sm.sm s.s.s.1 0xffffffff tcp     eq   20 gt 1023 secure local inbound
permit s.s.s.1 0xffffffff s.s.s.s sm.sm.sm.sm tcp/ack gt 1023 eq   20 secure local outbound
```

FTP Proxy from Secure Network to Nonsecure Network: This can be allowed, but remember to protect any tcp services that the Firewall is providing on nonprivileged ports (normally this is only SOCKS, unless you have added some other service).

In this case the FTP client connects to an FTP server on the firewall. Once there, it is authenticated by the server in the normal way. Once the user uses the quote site command, the proxy connects to the server. To this point, we only have control sessions from the client to port 21 on the firewall and from the nonsecure side of the firewall to port 21 on the target server. When the user gets or puts a file, the FTP client will specify a mode for the transfer (either normal or passive). The FTP proxy will use the same type of transfer to connect to the final server.

The following is an example of this with normal-mode FTP.

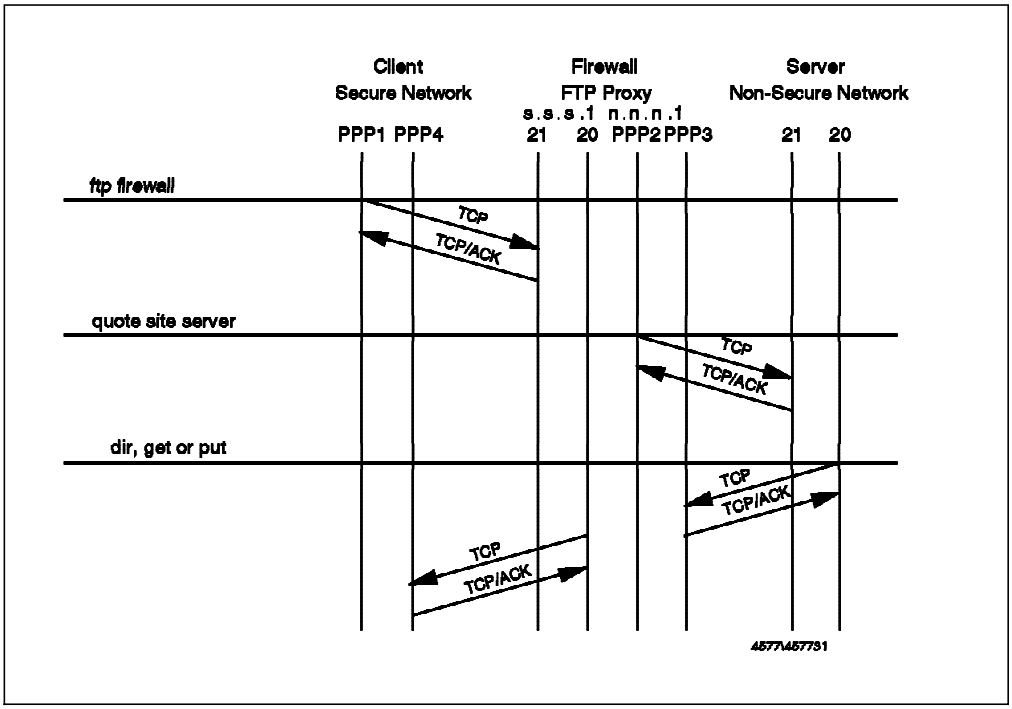


Figure 66. Normal Mode FTP from Secure Network to Nonsecure Network Using Proxy

The following connections provide FTP access from secure network to nonsecure network using the proxy server:

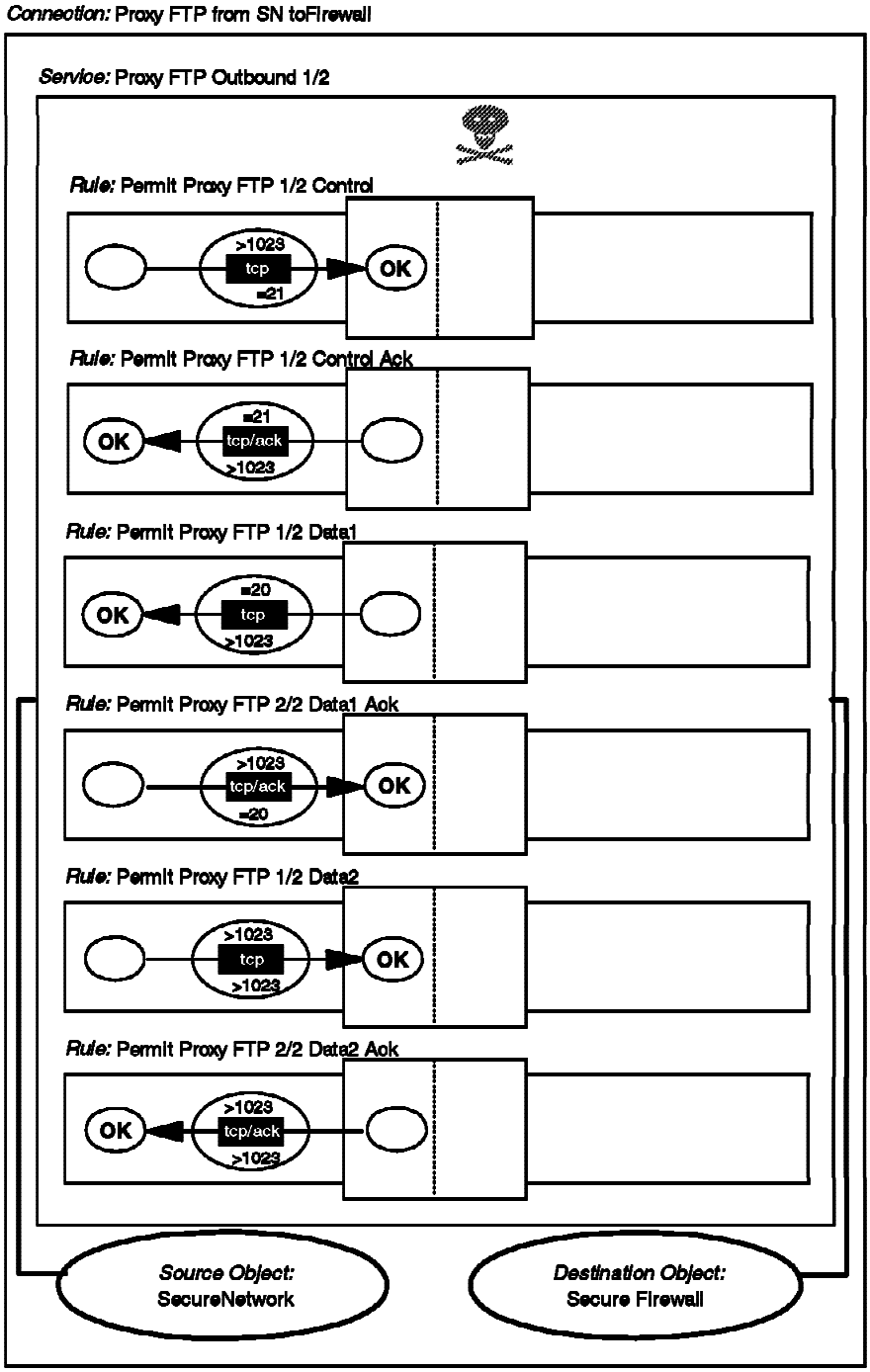


Figure 67. First Connection, Secure Network to FTP Proxy

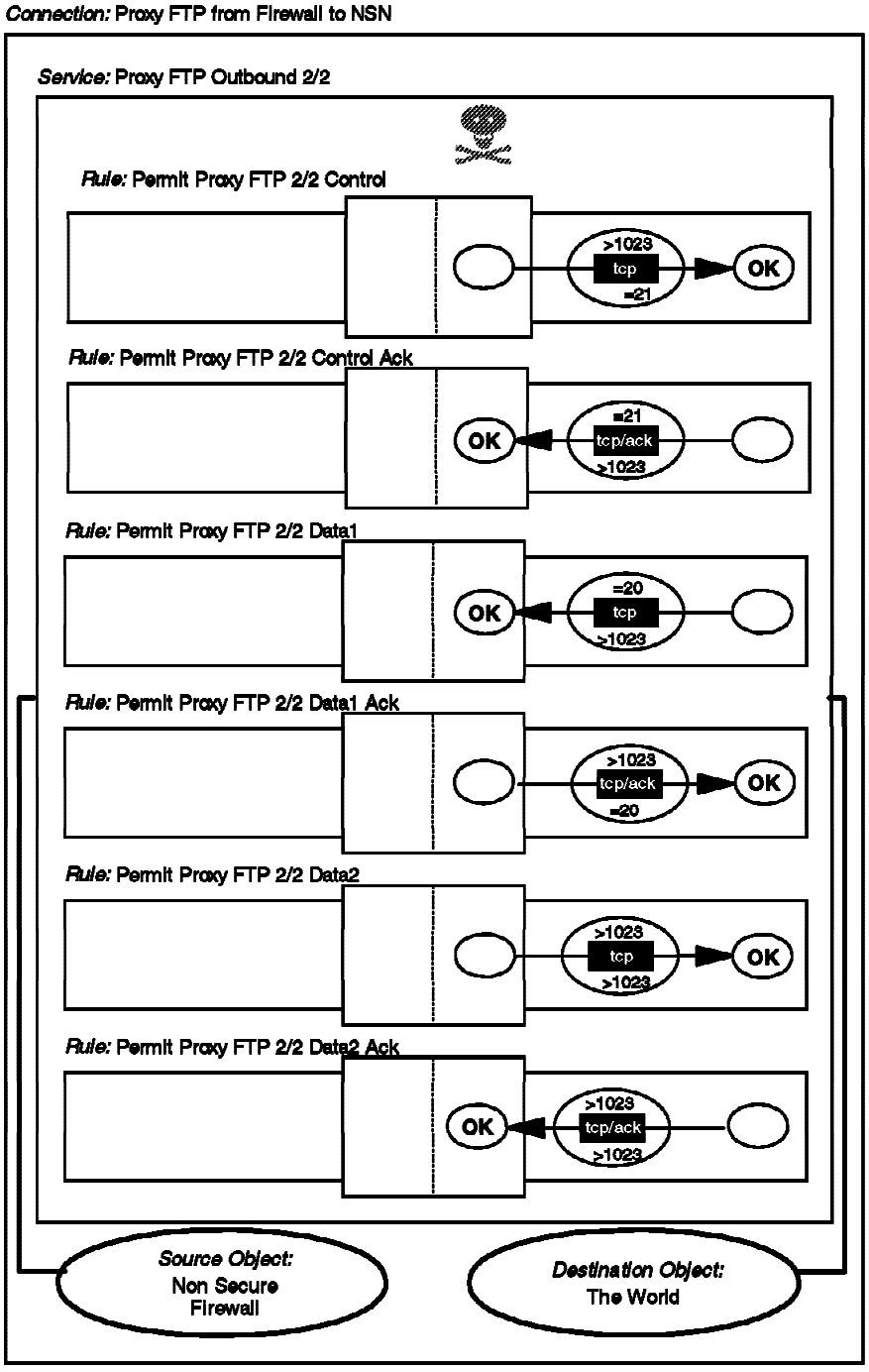


Figure 68. Second Connection, FTP Proxy to Nonsecure Network

The resulting filter rules are as follows:

```
# FTP From SN to NSN. FTP Control Session from Client to Firewall (1)
permit s.s.s.s sm.sm.sm.sm s.s.s.1 0xffffffff tcp     gt 1023 eq   21 secure local inbound
permit s.s.s.1 0xffffffff s.s.s.s sm.sm.sm.sm tcp/ack eq   21 gt 1023 secure local outbound

# FTP From SN to NSN. FTP Control Session from Firewall to Server (2)
permit n.n.n.1 0xffffffff 0 0 tcp     gt 1023 eq   21 nonsecure local outbound
permit 0 0 n.n.n.1 0xffffffff tcp/ack eq   21 gt 1023 nonsecure local inbound

# Normal Mode
```

```
# FTP From SN to NSN. FTP Data Session from Server to Firewall (3)
permit 0 0 n.n.n.1 0xffffffff tcp     eq   20 gt 1023 nonsecure local inbound
permit n.n.n.1 0xffffffff 0 0 tcp/ack gt 1023 eq   20 nonsecure local outbound

# FTP From SN to NSN. FTP Data Session from Firewall to Client (4)
permit s.s.s.1 0xffffffff s.s.s.s sm.sm.sm.sm tcp     eq   20 gt 1023 secure local outbound
permit s.s.s.s sm.sm.sm.sm s.s.s.1 0xffffffff tcp/ack gt 1023 eq   20 secure local inbound

# Passive Mode

# FTP From SN to NSN. FTP Data Session from Server to Firewall (3)
permit n.n.n.1 0xffffffff 0 0 tcp     gt 1023 gt 1023 nonsecure local outbound
permit 0 0 n.n.n.1 0xffffffff tcp/ack gt 1023 gt 1023 nonsecure local inbound

# FTP From SN to NSN. FTP Data Session from Firewall to Client (4)
permit s.s.s.1 0xffffffff s.s.s.s sm.sm.sm.sm tcp     gt 1023 gt 1023 secure local inbound
permit s.s.s.s sm.sm.sm.sm s.s.s.1 0xffffffff tcp/ack gt 1023 gt 1023 secure local outbound
```

FTP from Secure Network to Nonsecure Network Using SOCKS: The rules for FTP using SOCKS are very similar to the ones for the proxy. The difference is that in this case, the client only uses passive mode. You have to change the destination of the client's connection from 21 and gt 1023 to the SOCKS server on port 1080 and define only the passive mode rules.

You can create this connection by using the standard SOCKS connection definition for your secure network and a connection which uses a service that can be created easily by copying the service "Permit Proxy FTP Outbound 2/2" and removing 2 rules, see Figure 69 on page 93.

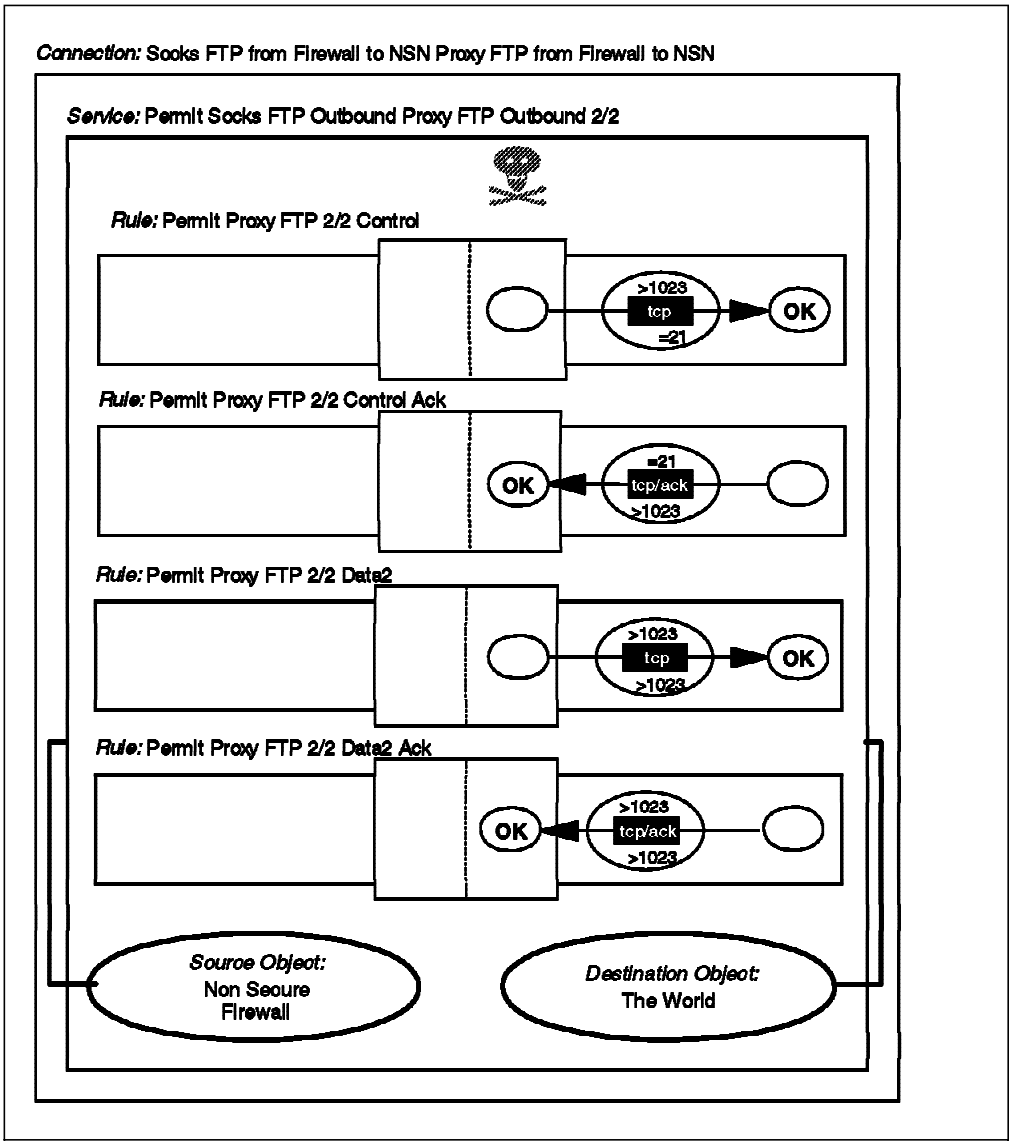


Figure 69. FTP from Secure Network To Nonsecure Network Using Socks

These connections generate the following filter rules:

```
# FTP From SN to NSN. FTP Control & Data Session from Client to SOCKS Server.
permit s.s.s.s sm.sm.sm.sm s.s.s.1 0xffffffff tcp     gt 1023 eq 1080 secure local inbound
permit s.s.s.1 0xffffffff s.s.s.s sm.sm.sm.sm tcp/ack eq 1080 gt 1023 secure local outbound

# FTP From SN to NSN. FTP Control Session from Firewall to FTP Server
permit n.n.n.1 0xffffffff 0 0 tcp     gt 1023 eq   21 nonsecure local outbound
permit 0 0 n.n.n.1 0xffffffff tcp/ack eq   21 gt 1023 nonsecure local inbound

# FTP From SN to NSN. FTP Data Session from Firewall to FTP Server (Passive Mode)
permit n.n.n.1 0xffffffff 0 0 tcp     gt 1023 gt 1023 nonsecure local outbound
permit 0 0 n.n.n.1 0xffffffff tcp/ack gt 1023 gt 1023 nonsecure local inbound
```

6.5 SMTP: Simple Mail Transfer Protocol

Mail formatted using the Simple Mail Transfer Protocol (SMTP) is one of the largest contributors to Internet traffic. In TCP/IP terms, SMTP is a straightforward, stream-oriented application. The SMTP server (the receiver of incoming mail) listens for connections on TCP port 25. The client uses any TCP port, usually one of the nonreserved ports (above 1023). In fact, with sendmail

version 8, there is a security option flag (F=R) that causes sendmail to connect from a privileged port, so the filter rules need to cater for this feature. For more information about this option, we recommend you to refer to *DNS and BIND in a Nutshell*, by Cricket Liu and Paul Albits (published by O'Reilly and Associates, ISBN 1-5659-20104).

As the SMTP protocol doesn't provide any control, a very important point in this service is to educate your users. Any user on the Internet could send E-mail messages with a forged origin, so you should tell your users that the apparent source of an E-mail message cannot always be trusted, unless the message is signed by a strong authentication mechanism such as Pretty Good Privacy (PGP). Fortunately, IBM Firewall will log the IP host name of the message sender, not the one that is specified in the SMTP connection handshake (in the HELO command)

There are many possible configurations for handling mail using IBM Firewall. We will discuss a few of them in Chapter 12, “Mail Handling” on page 231. For the purpose of this filter rule example we assume a desired configuration of a mail server in the secure network (M.M.M.M) and the use of the safemail gateway provided by IBM Firewall to deliver mail between the secure network and the nonsecure network.

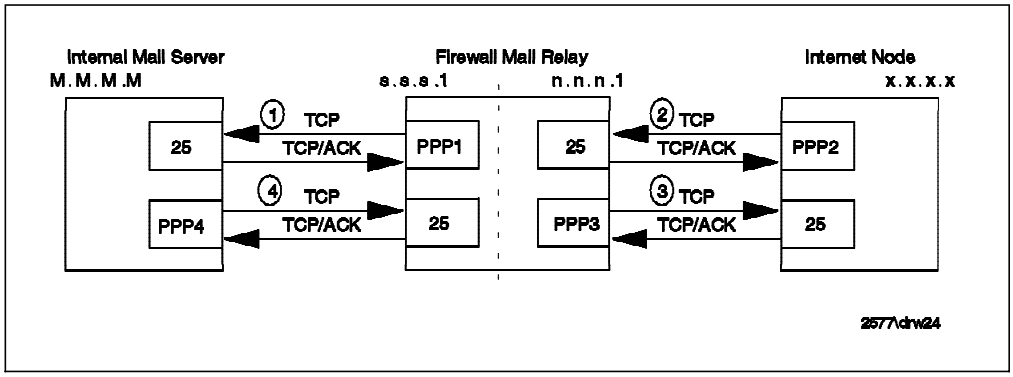


Figure 70. SMTP Mail Configuration

Since the sessions between the mail daemons may be set up in either direction, depending on where the mail arrives from, the connections have been designed to cater for either case.

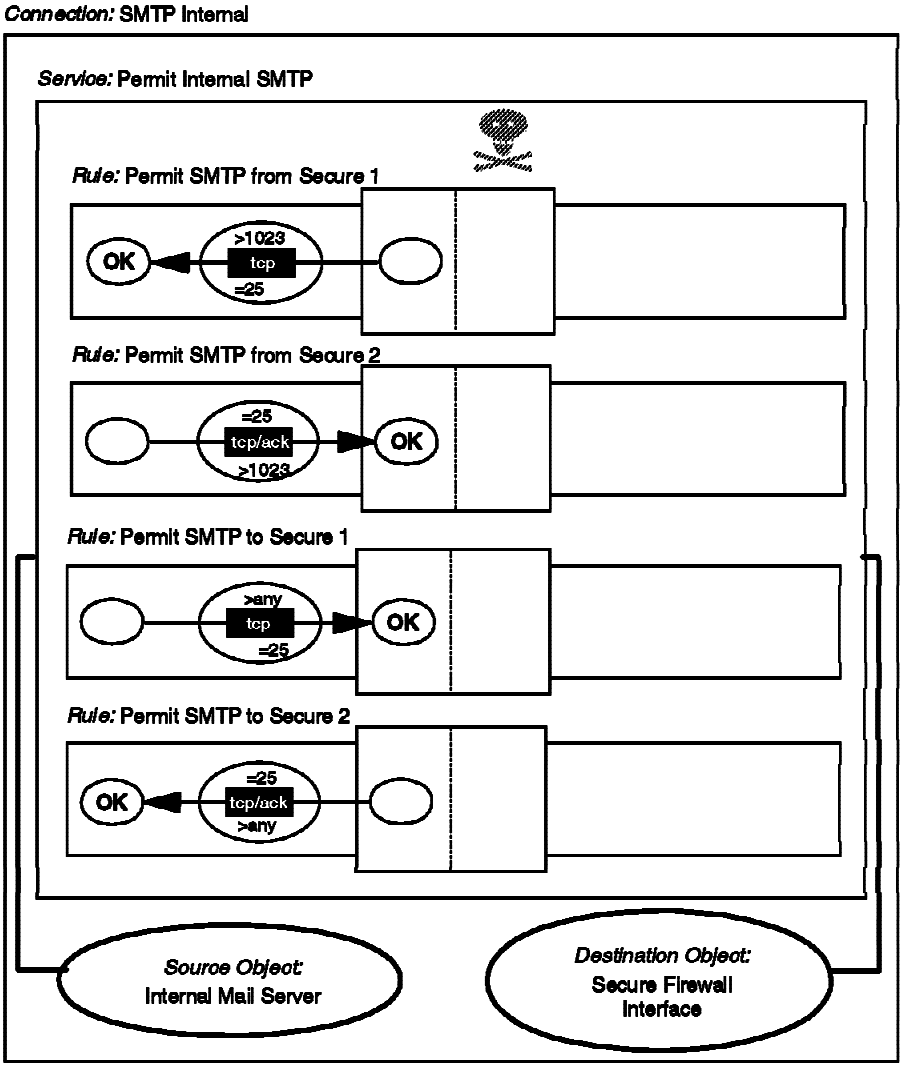


Figure 71. SMTP Mail from Secure Mail Server to Firewall Safemail Gateway

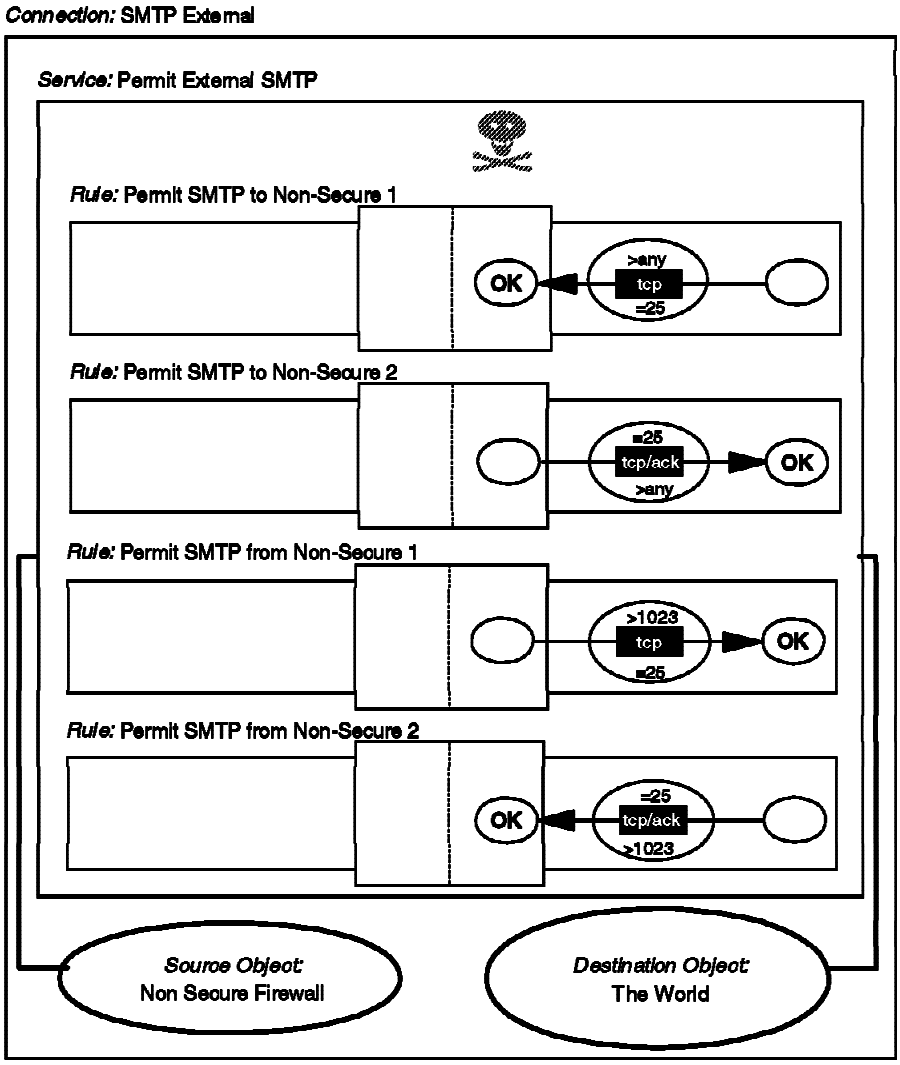


Figure 72. SMTP Mail from Firewall Safemail Gateway to Nonsecure Network

The resulting filter rules are as follows:

```
# Mail from the Firewall to the Internal Mail Server (1)
permit s.s.s.1 0xffffffff M.M.M.M 0xffffffff tcp     gt 1023 eq 25 secure local outbound
permit M.M.M.M 0xffffffff s.s.s.1 0xffffffff tcp/ack eq 25 gt 1023 secure local inbound

# Mail from the NSN to the Firewall (2)
permit 0 0 n.n.n.1 0xffffffff tcp     any   0 eq   25 nonsecure local inbound
permit n.n.n.1 0xffffffff 0 0 tcp/ack eq   25 any   0 nonsecure local outbound

# Mail from the Firewall to the NSN (3)
permit n.n.n.1 0xffffffff 0 0 tcp     gt 1023 eq   25 nonsecure local outbound
permit 0 0 n.n.n.1 0xffffffff tcp/ack eq   25 gt 1023 nonsecure local inbound

# Mail from the Internal Mail Server to the Firewall (4)
permit M.M.M.M 0xffffffff s.s.s.1 0xffffffff tcp     any   0 eq 25 secure local inbound
permit s.s.s.1 0xffffffff M.M.M.M 0xffffffff tcp/ack eq 25 any   0 secure local outbound
```

6.6 DNS: Domain Name Server

For reasons we have already discussed (see 2.1.4, “Domain Name Service” on page 17), it is not a good idea to allow nonsecure nodes to have unrestricted access to the name-to-address mapping of the nodes in the secure network. However, you will want some of your machines to be visible, and you will want your own name server to be able to access external name servers. We will further discuss the various DNS configurations in Chapter 11, “Domain Name Service” on page 215.

In our configuration, the internal DNS is responsible for resolving addresses for clients in the secure network (this includes the firewall). The firewall DNS is responsible for hiding internal information from outsiders and also for resolving outside addresses requested by the internal DNS.

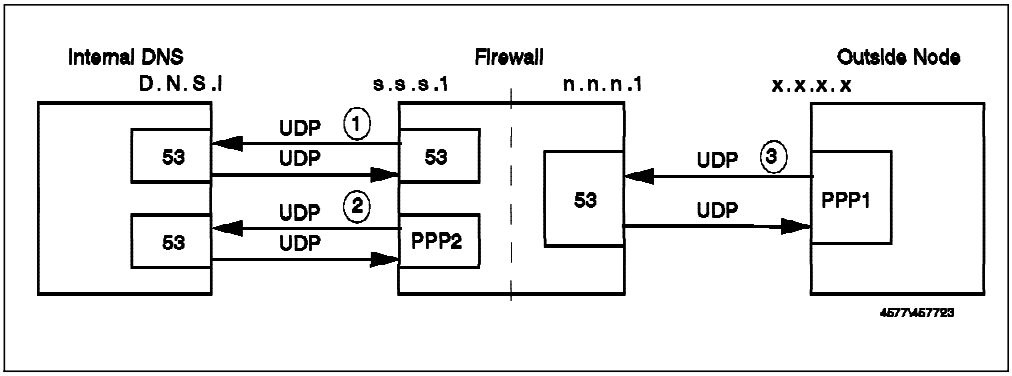


Figure 73. DNS

The following connections allow this DNS traffic:

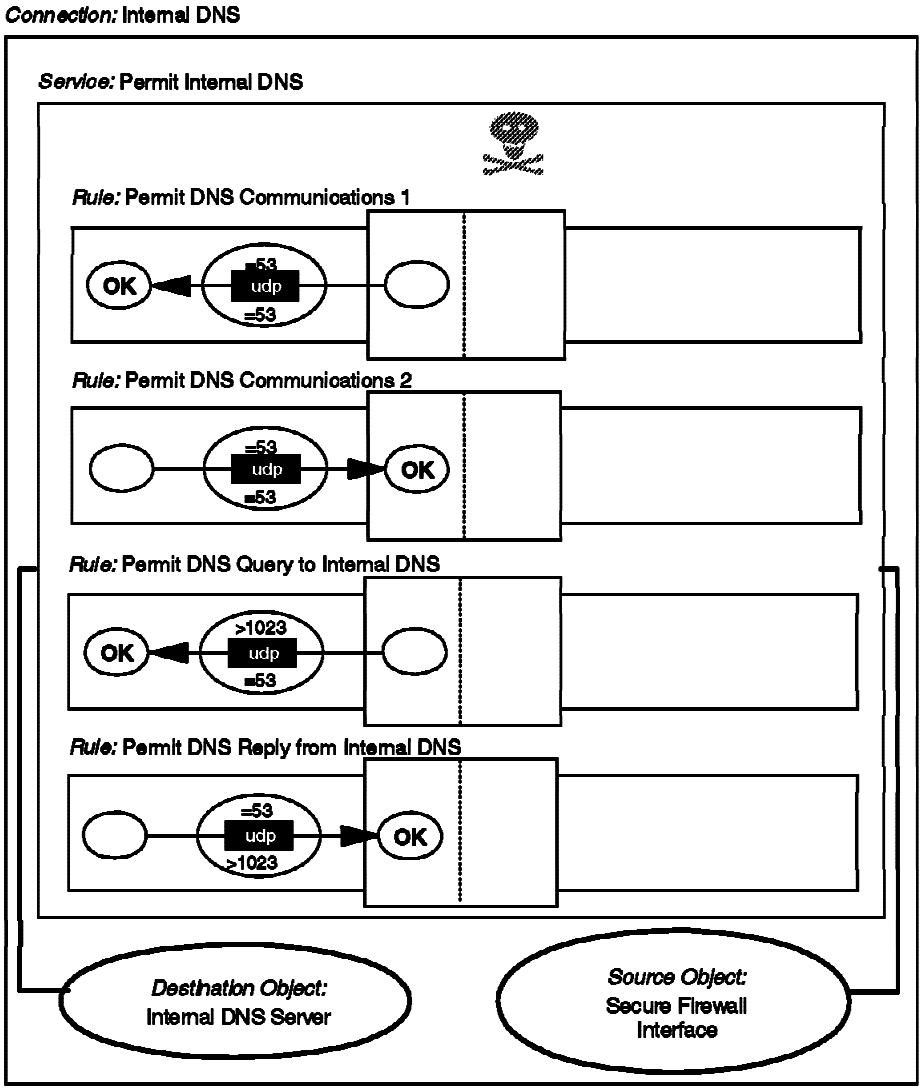


Figure 74. DNS between Secure Nameserver and Firewall

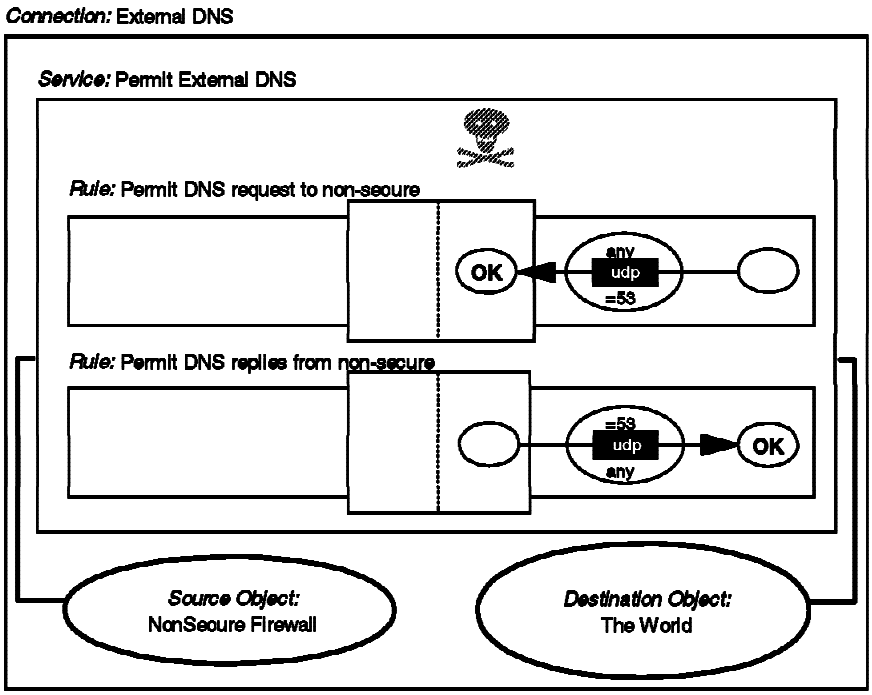


Figure 75. DNS Between Firewall and Nonsecure Nameservers

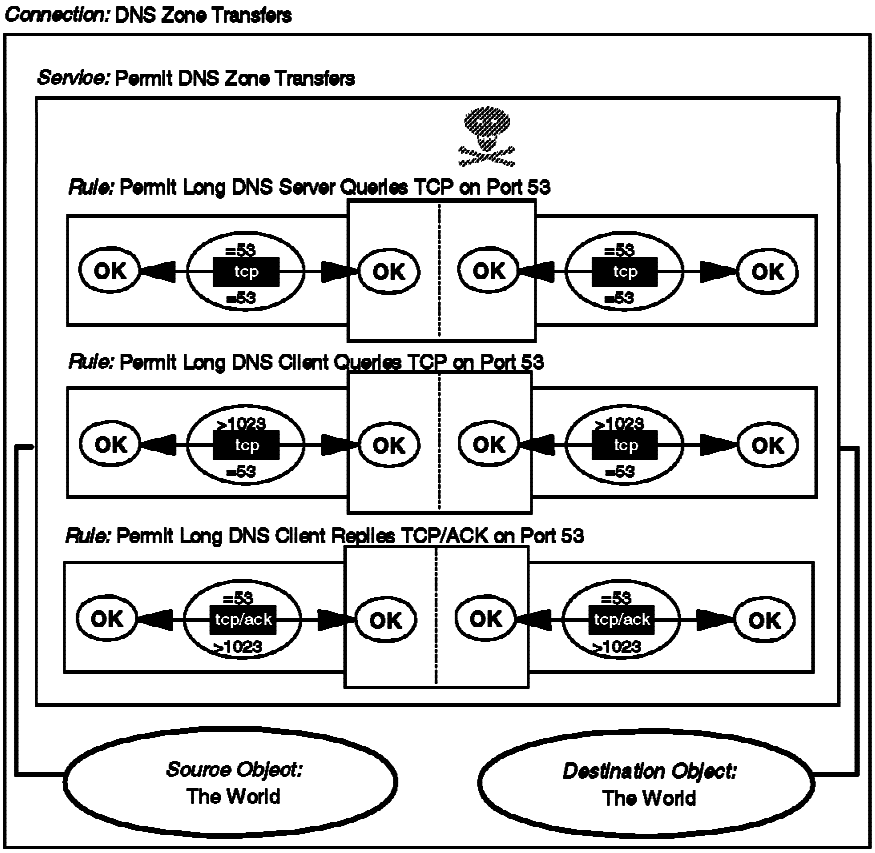


Figure 76. DNS Zone Transfers

The resulting filter rules are as follows:

```
# Communications Between the DNS Firewall and the Internal DNS.
# Requests & Replys from Firewall to Internal DNS Server (Clients of IBM Firewall)
permit s.s.s.1 0xffffffff D.N.S.I 0xffffffff udp eq 53   eq 53   secure local outbound
permit D.N.S.I 0xffffffff s.s.s.1 0xffffffff udp eq 53   eq 53   secure local inbound
permit s.s.s.1 0xffffffff D.N.S.I 0xffffffff udp gt 1024 eq 53   secure local outbound
permit D.N.S.I 0xffffffff s.s.s.1 0xffffffff udp eq 53   gt 1024 secure local inbound

# Requests from the Nonsecure Network to the Firewall
# Includes Communications Between the Firewall's DNS and the External DNS
permit 0 0 n.n.n.1 0xffffffff udp any 0 eq 53 nonsecure local inbound
permit n.n.n.1 0xffffffff 0 0 udp eq 53 any 0 nonsecure local outbound

# Allow use of TCP for DNS zone transfers
permit 0 0 0 0 tcp     eq   53 eq   53 both local both
permit 0 0 0 0 tcp     gt 1023 eq   53 both local both
permit 0 0 0 0 tcp/ack eq   53 gt 1023 both local both
```

The first connection permit requests and replies between DNS on the firewall and the internal DNS. It also allows the firewall itself (using resolv.conf) to resolve addresses from the secure network (directly) and from the nonsecure network, in which case the request will be forwarded to DNS on the firewall and then outside.

The second connection allows requests and replies between DNS on the firewall and the nonsecure network (to allow the firewall to resolve an external address on behalf of the internal name server).

The final connection allows the use of TCP for DNS zone transfers, so a server can retrieve, with just one connection, a whole sub-tree of the DNS name space.

6.7 NNTP: Network News Transfer Protocol

NNTP is a protocol used to transfer news. It is a TCP service in which the server listens on port 119. If your users employ an NNTP-capable reader with SOCKS support, you can use the configuration shown in Figure 77.

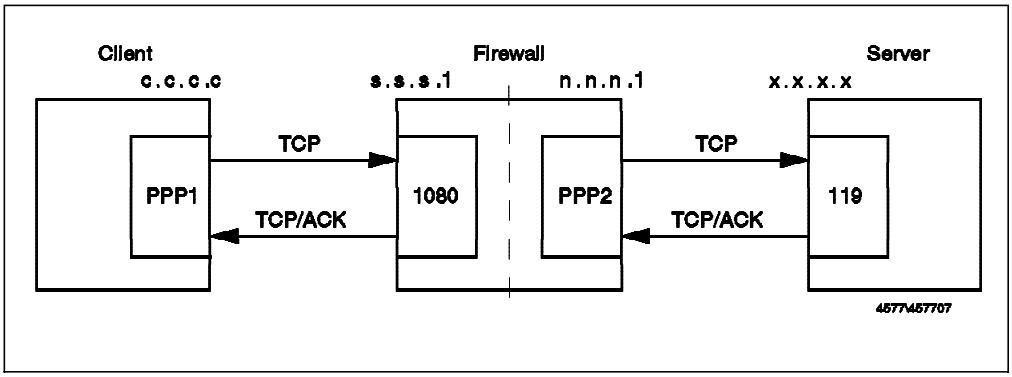


Figure 77. NNTP Client Using Socks Server

This requires two connection definitions. The first is a standard Socks client connection, as shown in Figure 61 on page 85. The second is shown below. Note that the rules used in this connection are not part of the predefined set, so they have to be manually added.

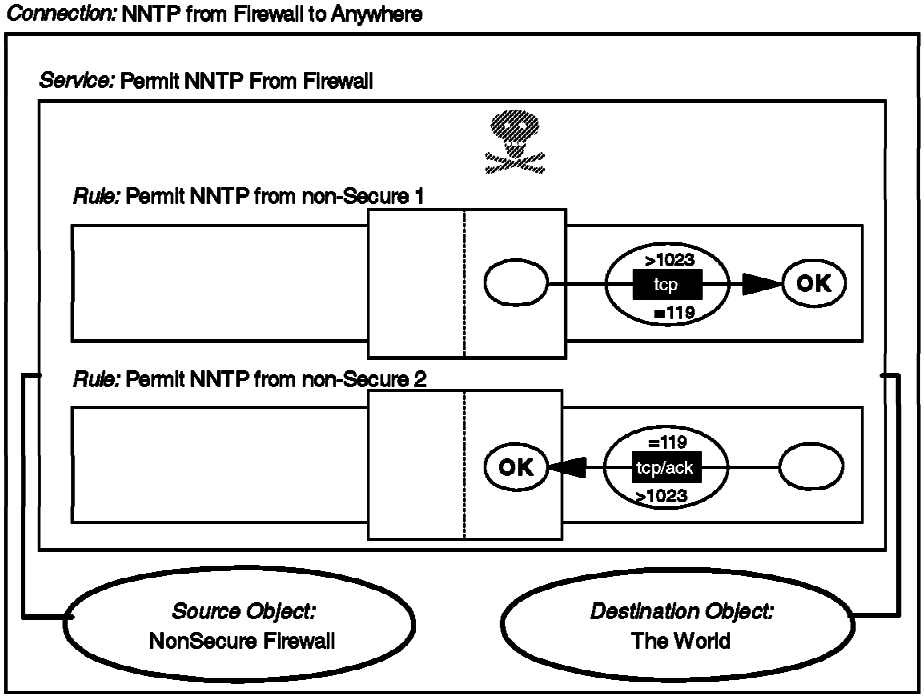


Figure 78. Connection to a Nonsecure NNTP Server

The filter rules generated by this configuration are as follows:

```
# Connection from the client to the Socks Server
permit s.s.s.s sm.sm.sm.sm s.s.s.1 0xffffffff tcp     gt 1023 eq 1080 secure local inbound
permit s.s.s.1 0xffffffff s.s.s.s sm.sm.sm.sm tcp/ack eq 1080 gt 1023 secure local outbound

# Connection from Firewall to the News Server in the Nonsecure Network
permit n.n.n.1 0xffffffff 0 0 tcp     gt 1023 eq 119  nonsecure local outbound
permit 0 0 n.n.n.1 0xffffffff tcp/ack eq 119  gt 1023 nonsecure local inbound
```

In the above case, the news server is outside the firewall, either in a remote network or in the DMZ. Often it is more convenient to have a news server in the secure network, in order to store news locally. This avoids duplicated traffic and allows you to restrict the news groups to only those that you would like to provide. You may also want to provide some internal news groups, which are not broadcast outside the firewall. If you do put a news server inside the secure network, you will receive news from an external news feeder to the news server. There is no standard proxy for NNTP in IBM Firewall, so the simplest approach is to allow IP forwarding of these packets.

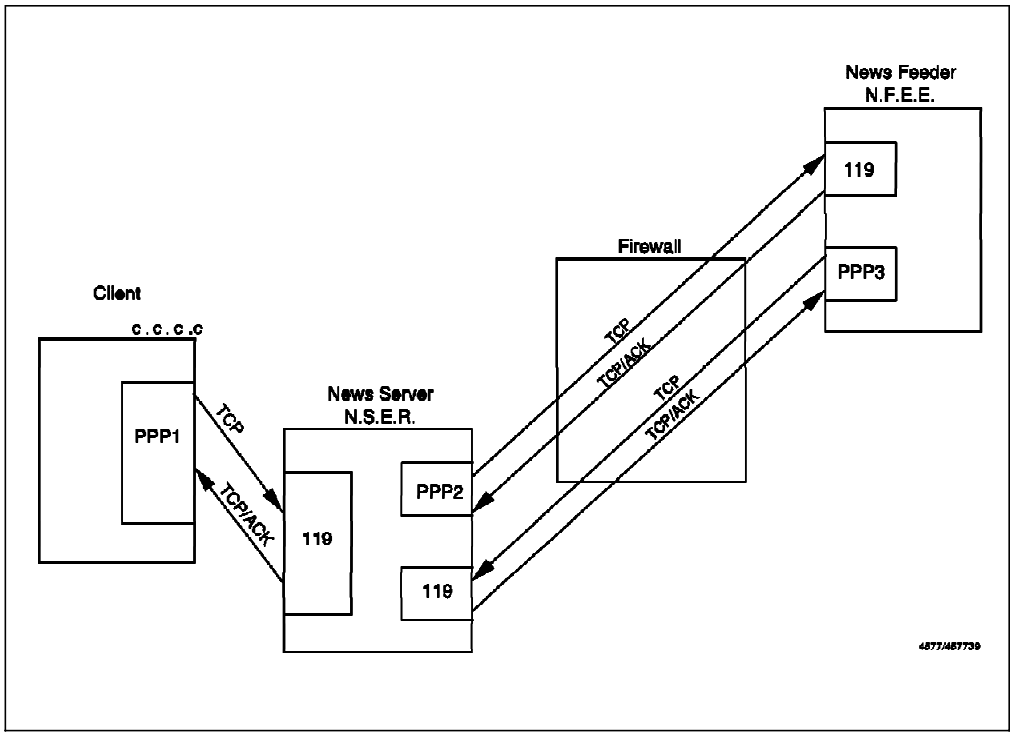


Figure 79. NNTP: News Server with IP Forwarding Enabled

The following connection definition can be used. As with the previous NNTP case, the rule definitions must be manually added, but in this case we can exploit the facility to reverse the direction of a rule, so in fact the first and third rules are identical, as are the second and fourth.

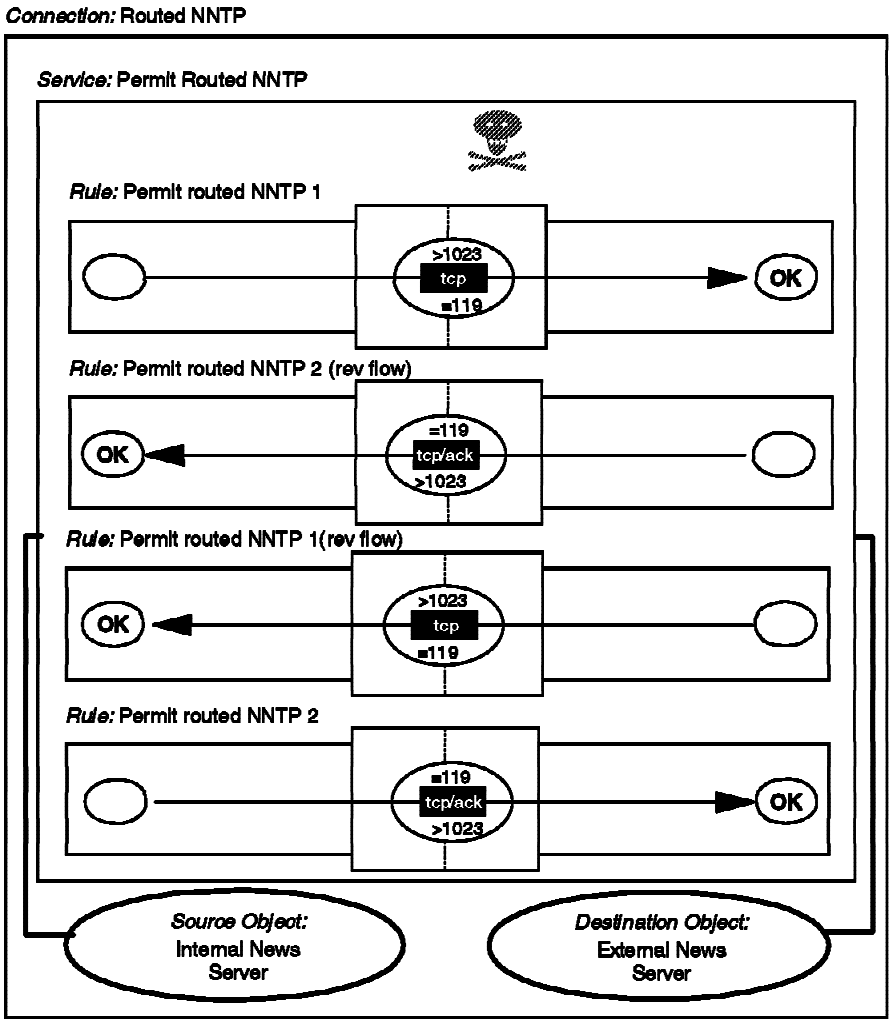


Figure 80. NNTP Routed through the Firewall

The following filter rules are implemented by this connection:

```
# Outgoing Postings. From Internal News Server to News Feeder.
permit N.S.E.R 0xffffffff N.F.E.E 0xffffffff tcp     gt 1023 eq 119 secure route inbound
permit N.F.E.E 0xffffffff N.S.E.R 0xffffffff tcp/ack eq 119 gt 1023 secure route outbound
permit N.S.E.R 0xffffffff N.F.E.E 0xffffffff tcp     gt 1023 eq 119 nonsecure route outbound
permit N.F.E.E 0xffffffff N.S.E.R 0xffffffff tcp/ack eq 119 gt 1023 nonsecure route inbound

# Incoming News. From News Feeder to Internal News Server.
permit N.F.E.E 0xffffffff N.S.E.R 0xffffffff tcp     gt 1023 eq 119 secure route inbound
permit N.S.E.R 0xffffffff N.F.E.E 0xffffffff tcp/ack eq 119 gt 1023 secure route outbound
permit N.F.E.E 0xffffffff N.S.E.R 0xffffffff tcp     gt 1023 eq 119 nonsecure route outbound
permit N.S.E.R 0xffffffff N.F.E.E 0xffffffff tcp/ack eq 119 gt 1023 nonsecure route inbound
```

We would prefer to run a dual-homed firewall with no IP forwarding being permitted at all. To do this, we need to either have a SOCKSified news server or a proxy in the firewall that communicates with both the news feeder and the news server and acts as a relay between the two. We wrote a simple program called tcp_relay that allows IBM Firewall to relay news connections without allowing IP forwarding. This program is based on code from *Actually Useful Internet Techniques* by Larry Hughes (published by New Riders, ISBN 1-56205-508-9).

The C source for the tcp_relay program is listed in Appendix D, “Sample NNTP Relay Program” on page 279 and you can retrieve it using anonymous FTP as described in Appendix A, “How to Get the Samples in This Book” on page 271.

If the tcp_relay program is used as NNTP proxy, the internal news server should be configured to receive news feed from the secure side of the firewall, not from the external news feeder. Similarly, the external news feeder should feed news to the nonsecure side of the firewall.

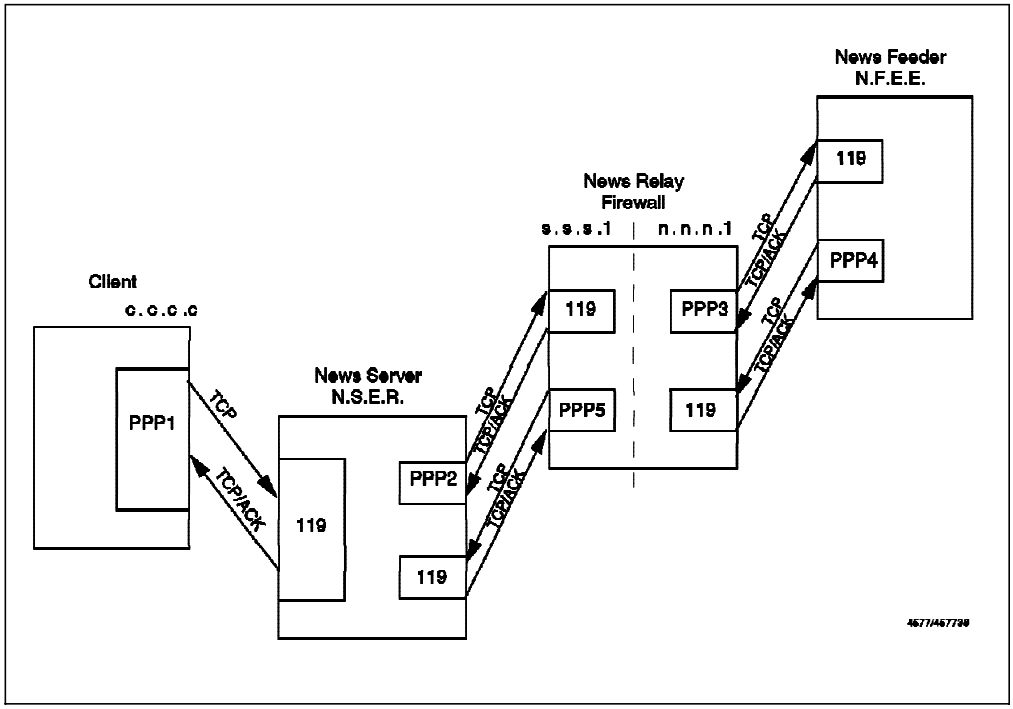


Figure 81. NNTP: News Server Using the tcp_relay Sample Program

These are the connections to allow news to be delivered using tcp_relay between the news feeder and the internal news server (all rules manually defined):

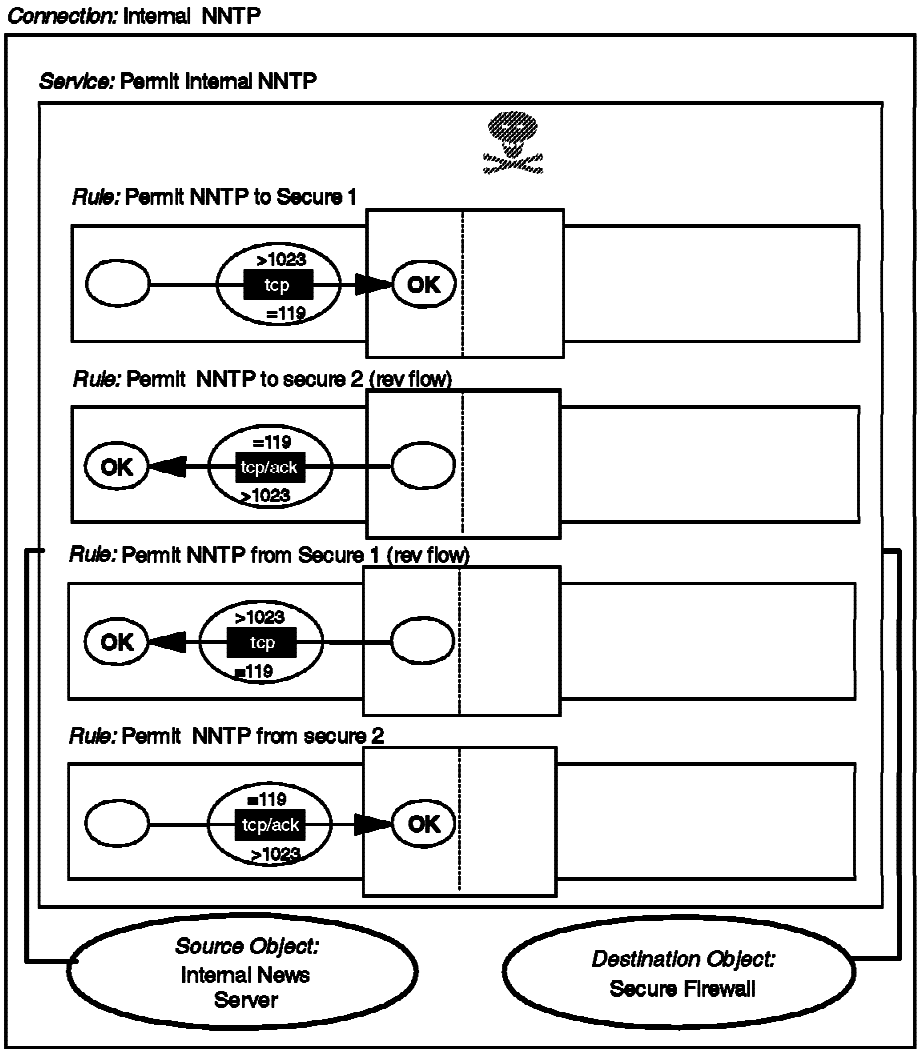


Figure 82. NNTP between Internal Server and tcp_relay

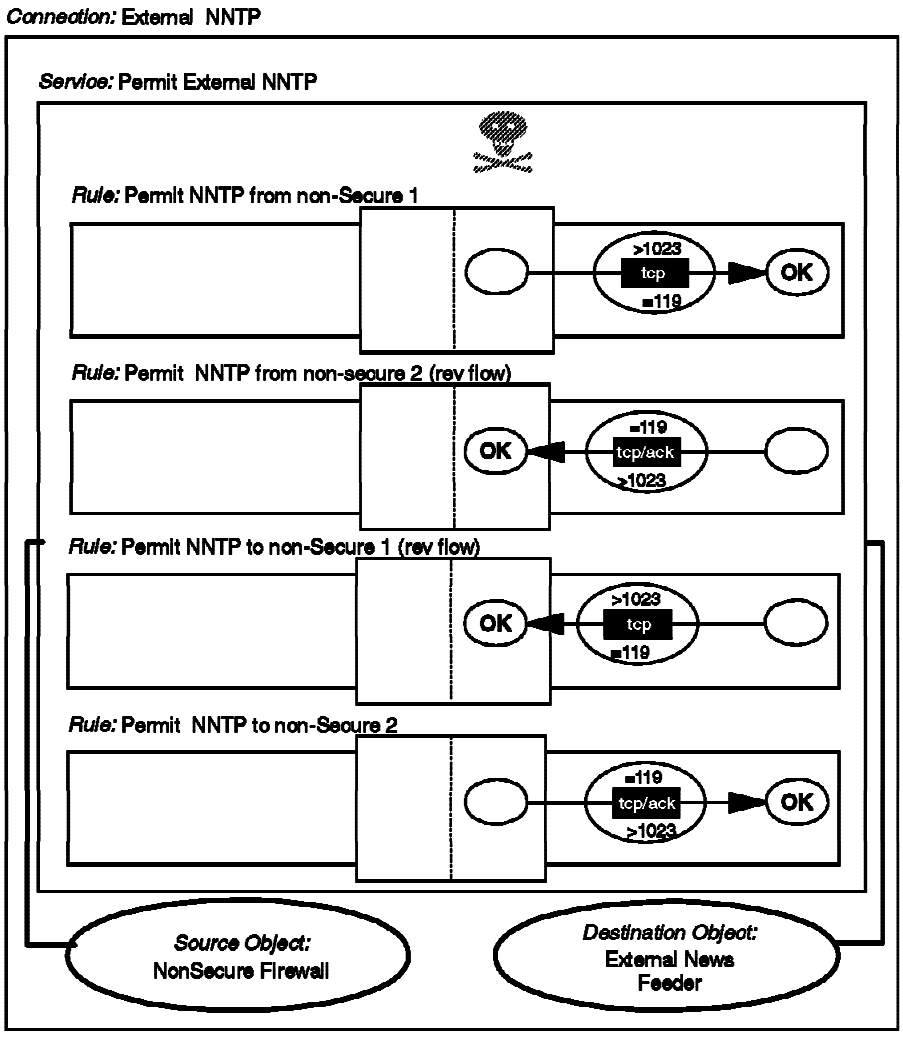


Figure 83. NNTP between External News Feeder and tcp_relay

As before, the rules are in pairs, with the flow of one pair being reversed.

The following filter definitions derive from these connections:

```
# Outgoing Postings. First Half (From News Server to Firewall)
permit N.S.E.R 0xffffffff s.s.s.1 0xffffffff tcp     gt 1023 eq  119 secure local inbound
permit s.s.s.1 0xffffffff N.S.E.R 0xffffffff tcp/ack eq  119 gt 1023 secure local outbound
# Outgoing Postings. Second Half (From Firewall to News Feeder)
permit n.n.n.1 0xffffffff N.F.E.E 0xffffffff tcp     gt 1023 eq  119 nonsecure local outbound
permit N.F.E.E 0xffffffff n.n.n.1 0xffffffff tcp/ack eq  119 gt 1023 nonsecure local inbound

# Incoming News. First half (From news feeder to Firewall)
permit N.F.E.E 0xffffffff n.n.n.1 0xffffffff tcp     gt 1023 eq  119 nonsecure local inbound
permit n.n.n.1 0xffffffff N.F.E.E 0xffffffff tcp/ack eq  119 gt 1023 nonsecure local outbound
# Incoming News. Second half (From Firewalls to Internal News Server)
permit s.s.s.1 0xffffffff N.S.E.R 0xffffffff tcp     gt 1023 eq  119 secure local outbound
permit N.S.E.R 0xffffffff s.s.s.1 0xffffffff tcp/ack eq  119 gt 1023 secure local inbound
```

6.8 HTTP - World Wide Web Sessions

The World Wide Web (WWW) is a collection of sites and services, loosely connected by a network of inter-document links. The normal way to access this network is using a graphical browser interface such as Netscape Navigator, Microsoft Internet Explorer or NCSA Mosaic.

From a technical point of view the main vehicle for WWW documents is the *HyperText Transfer Protocol* (HTTP). This is a lightweight protocol for requesting and receiving information using any encoding mechanism recognized by both the client and server. HTTP is a stateless protocol, that is, the server retains no continuing session information about a client. This has the benefit of allowing great simplicity and extensibility at the expense of some network bandwidth overhead.

The basic HTTP exchange is a simple request-response sequence:

1. A *request* is sent from an unprivileged port on the client to the server on TCP port 80.

   The request may contain either a simple GET action, or a more complex *Method* description (used, for example, when the client is sending a form that the user has filled in). The request also contains information about the client, such as the browser software in use and a list of the document formats that it can handle.

2. The *response* packets flow on the reverse path to the request (that is, from port 80 back to the requesting client port).

   The response is usually a document encoded using *Multi-Purpose Internet Mail Extensions* (MIME). The document will be written in one of the formats acceptable to the browser, most commonly the *HyperText Markup Language* (HTML), a document composition language which allows you to imbed links to other documents and to other graphical and multimedia objects.

WWW hyper-links do not always lead to other HTTP connections. Often they will take you to an anonymous FTP or gopher site. When you select the hyper-link to such a site, the WWW browser software invokes the appropriate service under the covers. So when you plan to provide this service, you should also consider the other related protocols.

As you can see from the previous description, most of the complexity of WWW lies in the higher-layer application code (which is a source for security concern, particularly as the content of Web documents becomes more interactive and as Web browsers become smarter). See *Safe Surfing: How to Build a Secure World Wide Web Connection*, SG24-4564, for more information.

6.8.1 Possible Scenarios

HTTP is a relatively new protocol. It was developed when firewalls were commonly in use on the Internet, so usually the Web browsers are proxy and SOCKS aware. As always, we aim to break all sessions at the firewall boundary, so a SOCKS or proxy arrangement is best. There is another important advantage in the use of proxy servers. Some proxy servers also cache Web documents and save the most frequently accessed ones. This is a considerable performance improvement for the users and also helps to reduce network traffic (and hence also costs, when your Internet connection charges are traffic-based). IBM Firewall includes a Web proxy server, but this is *not* a caching version. The

reason for this is that placing a complex application server of this kind on the firewall would break the KISS principle.

If you are going to provide a Web server for access by the rest of the world, it should be outside your secure network, ideally in a DMZ. The reason for this is that such servers are, by the nature of the application, likely to be complex and therefore vulnerable to attack. If you can isolate your server by putting it outside your secure network, you are limiting the damage that such break-ins can do. Many examples of attacks on Web servers have been demonstrated, so you should follow the CERT advisories to check if you are exposed.

Consider the following scenarios:

SOCKSified Client: In this case, clients in the secure network connect to the SOCKS server on the firewall, which relays their sessions into the Internet. One disadvantage of this configuration is that there is no caching of documents, so if multiple users load the same document, it will be retrieved multiple times.

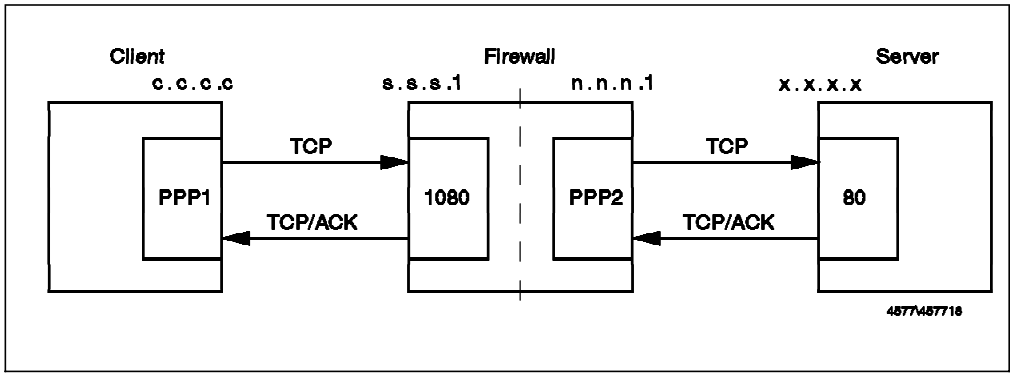


Figure 84. HTTP from Secure Network, Using a SOCKSified Client

This configuration requires two connections. The first is the familiar SOCKS client connection shown in Figure 61 on page 85. The second connection is as follows:

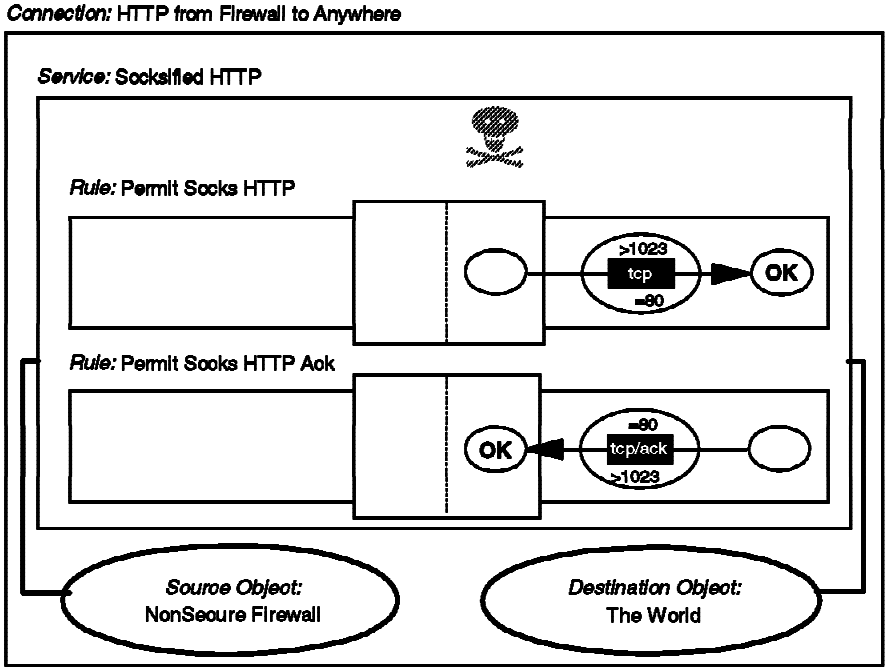


Figure 85. HTTP from SOCKS to Nonsecure Server

The equivalent filter definitions are as follows:

```
# Connection from the client to the Socks Server
permit s.s.s.s sm.sm.sm.sm s.s.s.1 0xffffffff tcp     gt 1023 eq 1080 secure local inbound
permit s.s.s.1 0xffffffff s.s.s.s sm.sm.sm.sm tcp/ack eq 1080 gt 1023 secure local outbound

# Connection from Firewall to the Server in the Nonsecure Network
permit n.n.n.1 0xffffffff 0 0 tcp     gt 1023 eq 80 nonsecure local outbound
permit 0 0 n.n.n.1 0xffffffff tcp/ack eq 80 gt 1023 nonsecure local inbound
```

Socksified Proxy: This extends the SOCKS configuration by placing a proxy server in the secure network. Clients connect to the proxy, which serves documents from cache if it is available or establishes a connection across the SOCKS relay, if not. The proxy server has to have SOCKS support built in for this configuration to work correctly. Many proxy servers provide this facility, including the IBM Internet Connection family.

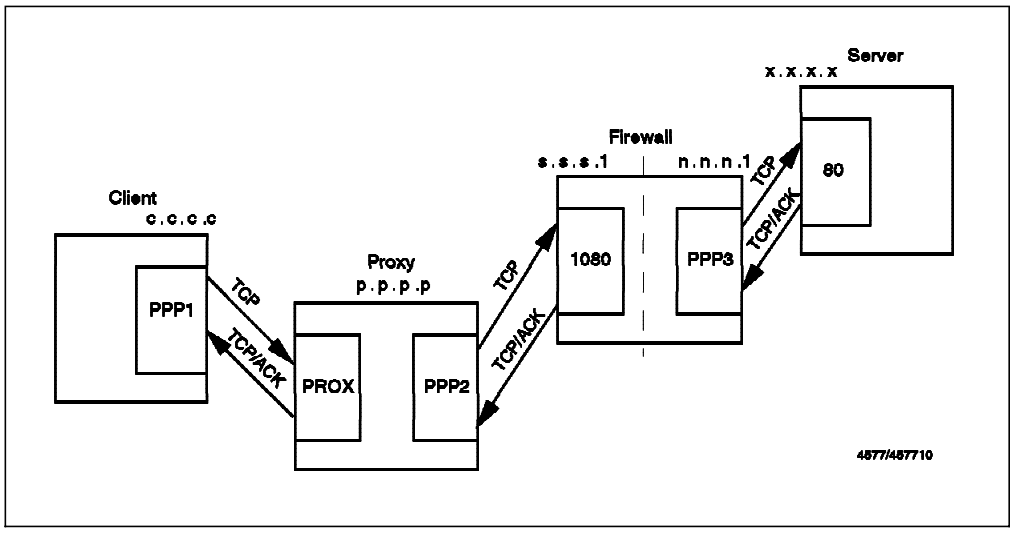


Figure 86. HTTP from Secure Network, Using a Socksified HTTP Proxy

The filtering rules for this case are the same as for the Socksified Client, except that instead of allowing connections from every client in the secure network, you can restrict them to the proxy server only. You could also restrict access to the SOCKS server so that *only* the proxy server can use it for HTTP. To do this, you would put the following lines in the sockd.conf file:

```
# Restrict socksified HTTP Connections (port 80) only to HTTP Proxy
permit p.p.p.p 255.255.255.255 eq 80
deny 0.0.0.0 0.0.0.0 eq 80
```

Using the IBM Firewall HTTP Proxy Server in Place of SOCKS: The two previous examples (“SOCKSified Client” on page 108 and “Socksified Proxy” on page 109) both use the SOCKS server to relay sessions through the firewall. In each case you could replace it with the HTTP proxy server that comes as part of IBM Firewall by simply using rules that permit access to the proxy port (usually tcp/8080), in place of the SOCKS port (tcp/1080).

Proxy in the Secure Network with IP Forwarding Enabled in the Firewall: This has the disadvantage that you must enable IP forwarding in the firewall, but it is relatively secure because the filter rules are very restrictive (just one host and outbound connection only).

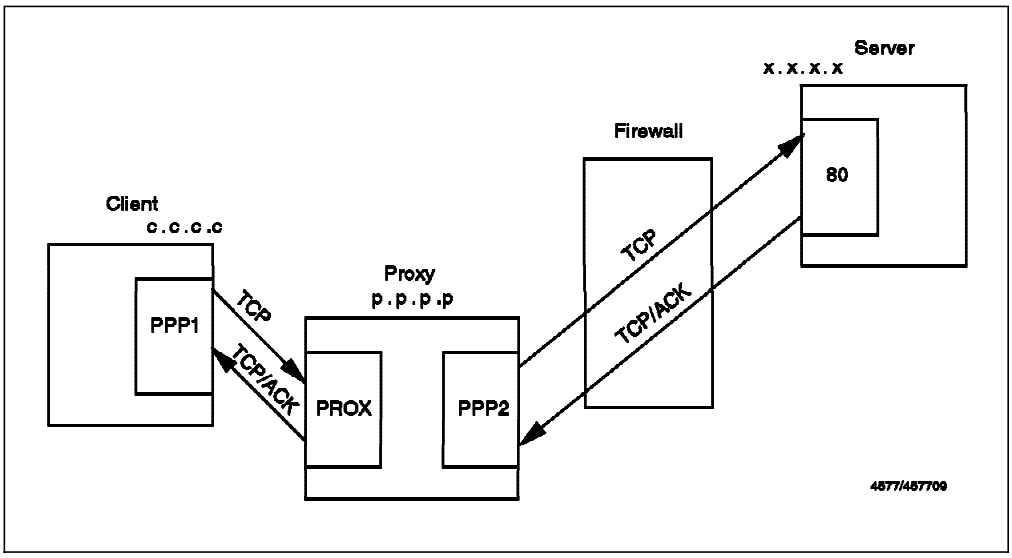


Figure 87. HTTP from Secure Network, HTTP Proxy with IP Forwarding Enabled

Proxy Outside the Secure Network: This case is also a good solution, especially if you are using a DMZ configuration. You can put the proxy server in the DMZ for caching. Ideally you should only allow your users to access the proxy using SOCKS.

The clients request a document using SOCKS from the proxy that is outside the secure network. The proxy does the caching, and you don't have to allow routing in the firewall, so you can run a dual-homed configuration.

In this case, you will need your Web browsers to support concurrent SOCKS and HTTP proxy. You can accomplish this if your client TCP/IP implementation has built-in SOCKS support (for example, in Warp V4) by specifying a proxy in the browser configuration menu, and then use the SOCKS runtime from the TCP/IP configuration options (that is, not the browser SOCKS option).

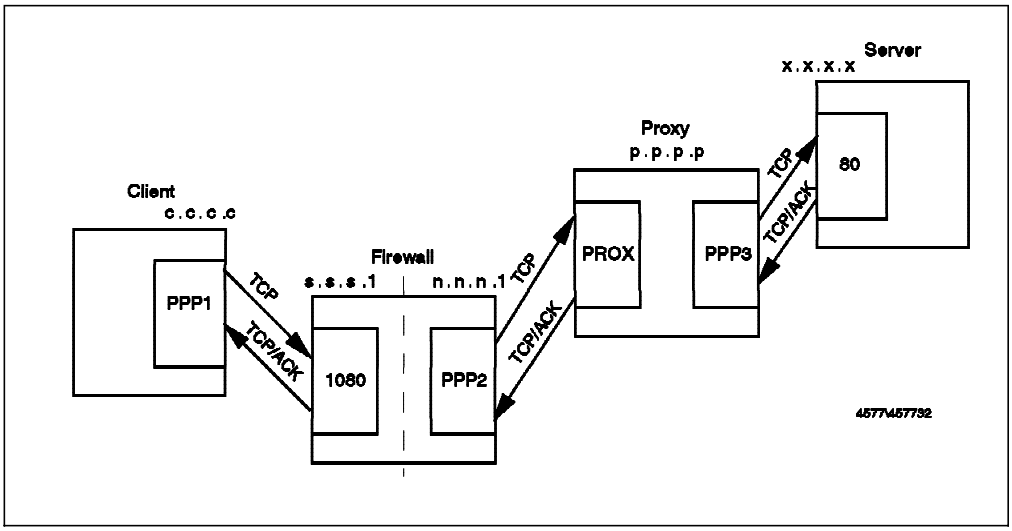


Figure 88. HTTP Proxy in the Nonsecure Network (Preferably Using DMZ)

In this kind of configuration it is very important to treat the proxy server in the DMZ as part of your firewall. It should be monitored closely for any sign of attack. If a cracker were to manage to take control of it he could use it to set up

a *Web Spoof*. This means that the attacker has control over all communications to anywhere in the Web. He can choose to simply act as a relay, recording passwords and information as it flows past, or he could intervene and insert his own messages or malicious executables into the pages that your users retrieve.

6.9 SSL: Secure Sockets Layer

SSL is a protocol developed by Netscape Communications Corporation along with RSA Data Security, Inc. It attempts to provide an alternative for the standard TCP/IP socket API to resolve problems of authentication and encryption. Although this means that, in theory, it could be used to protect any TCP/IP application, it is used almost exclusively for HTTP. It is used in Web-based applications to transport confidential data such as credit card numbers and is also able to provide authentication of the Web server, using public key certificates. Client authentication is defined in the SSL standard, but it is not commonly used yet.

From a firewall perspective SSL is a standard TCP service in which the client uses any nonprivileged port and the server uses port 443. Follow the recommendations for HTTP and substitute port 443 in place of 80.

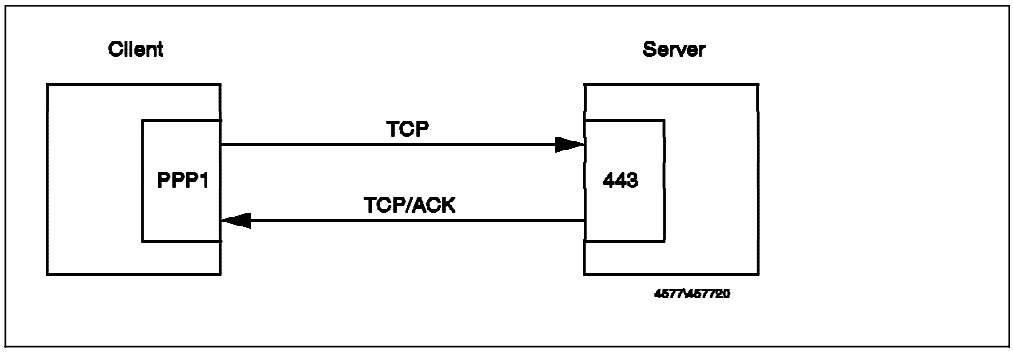


Figure 89. SSL Connection between Client and Server

6.10 S-HTTP

S-HTTP is a protocol designed in order to correct the lack of authentication and encryption of HTTP. S-HTTP is very rarely used in practice. SSL is a much easier protocol to administer and, although it is less flexible than S-HTTP, it has proved adequate for most secure Web applications.

From a filtering point of view, S-HTTP is identical to HTTP. The server listens for connections on tcp port 80, and the clients use any nonprivileged port. The server will realize if it is an HTTP connection or an S-HTTP connection from the URL (if it starts with http, it is HTTP, if it starts with shttp, it is S-HTTP). Web documents retrieved with S-HTTP should not be cached in a proxy server, because the proxy server cannot determine what S-HTTP options are appropriate.

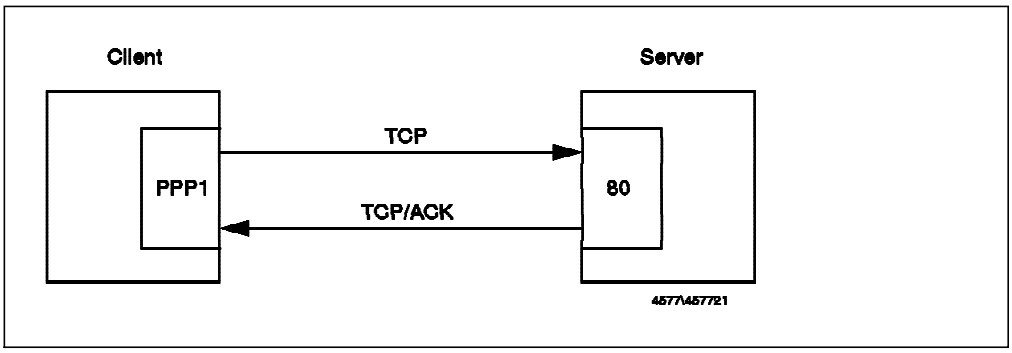


Figure 90. Secure HTTP Connection between Client and Server

6.11 Gopher

Gopher is a protocol used for information retrieval. It is not as popular as HTTP, but there are still some Gopher servers to which you may wish to provide access. Usually the client machine does not use a specific client for Gopher, because the most popular Web browsers already have support for this protocol.

The server uses tcp/70, and the client can use any nonprivileged port.

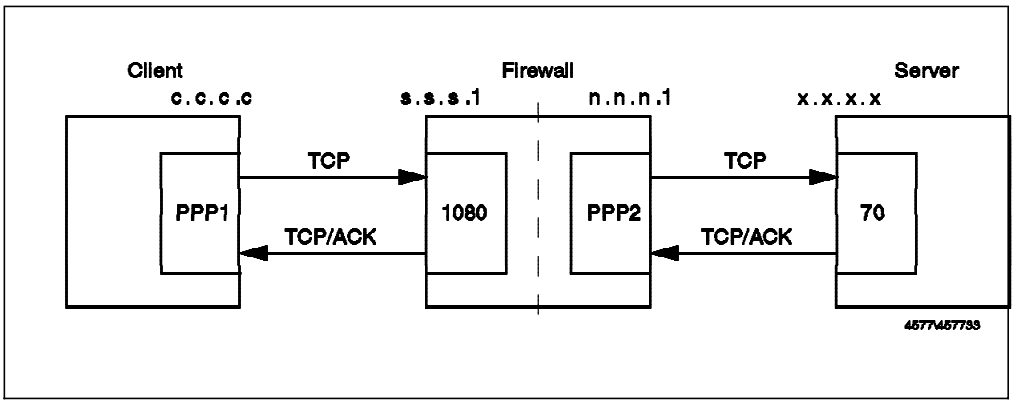


Figure 91. Gopher, Using a Socksified Client

Two connections are needed for this configuration. The first is the standard socks client connection (see Figure 61 on page 85). The second is as follows (the rules in this connection are not part of the standard set, so they must be manually created).

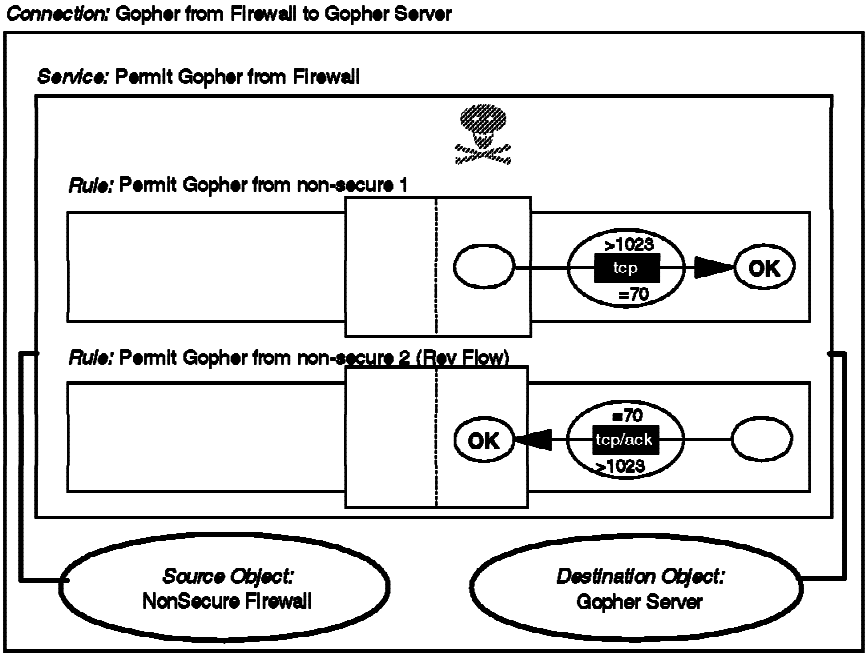


Figure 92. Gopher from SOCKS to Nonsecure Server

```
# Connection from the client to the Socks Server
permit s.s.s.s sm.sm.sm.sm s.s.s.1 0xffffffff tcp     gt 1023 eq 1080 secure local inbound
permit s.s.s.1 0xffffffff s.s.s.s sm.sm.sm.sm tcp/ack eq 1080 gt 1023 secure local outbound

# Connection from Firewall to the Server in the Nonsecure Network
permit n.n.n.1 0xffffffff 0 0 tcp     gt 1023 eq 70 nonsecure local outbound
permit 0 0 n.n.n.1 0xffffffff tcp/ack eq 70 gt 1023 nonsecure local inbound
```

6.12 Lotus Notes and Domino

A full description of Lotus Notes firewall configurations can be found in *The Domino Defense: Security in Lotus Notes and the Internet*, SG24-4848. We will describe the major aspects regarding the firewall rule base setup.

A notes server is normally inside a private enterprise network. Defining rules to access the server is not an issue in that case. If we are using Domino to provide access to Notes databases from the Internet it is recommended to position the Domino server in the DMZ, as visualized in Figure 93 on page 115. It is put in the DMZ, because a Domino server is, from the point of view of a client on the Internet, just another HTTP server, so we apply the same practices for positioning and protecting it as an HTTP server (6.8, “HTTP - World Wide Web Sessions” on page 107).

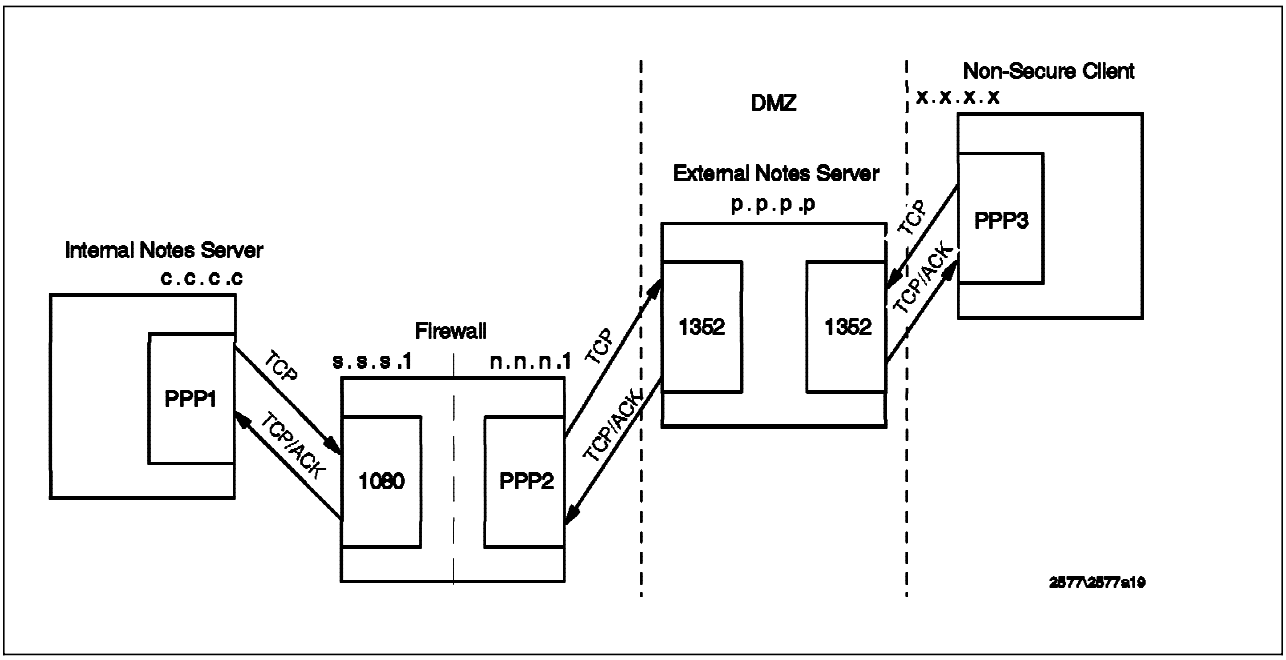


Figure 93. Domino in a DMZ Configuration

To keep the Domino server up to date we use Notes' replication capabilities. Because we only want to allow outgoing Notes connections we have to use the "Push-Pull" kind of replication, initiated from the secure network. This means that although database updates are replicated in both directions, the replication session is only initiated by the secure Notes server.

Replication and client access to a Notes server are both handled by the same standard TCP protoocol on port 1352. To split up the connection on the firewall we can use a similar setup as with HTTP via SOCKS. For this setup we need to implement the SOCKS client connection, (Figure 61 on page 85) plus the following connection to allow the SOCKS server to communicate with the external Notes server.

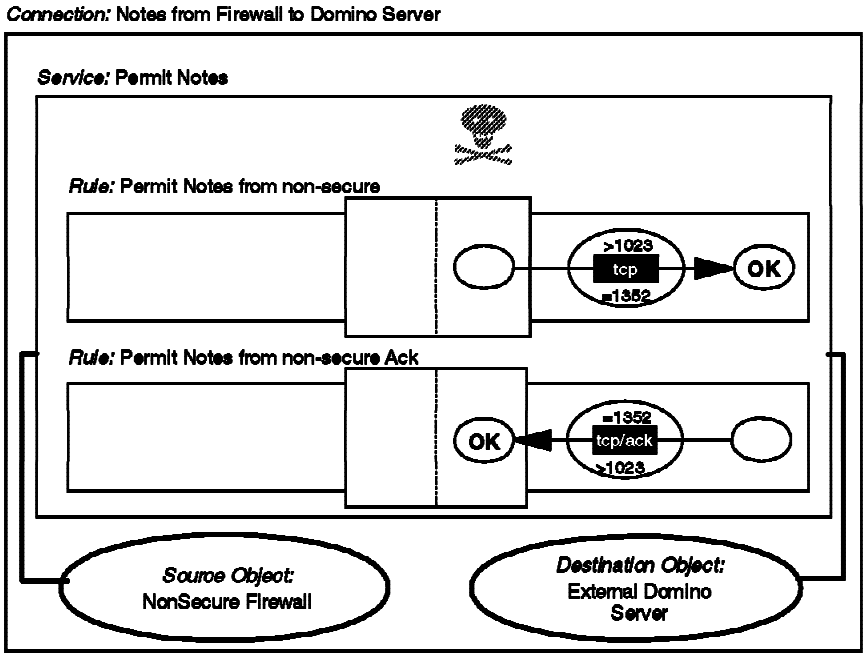


Figure 94. Lotus Notes from SOCKS to Nonsecure Server

The equivalent filter rules are as follows:

```
# Connection from the Internal Notes Server to the Socks Server
permit c.c.c.c 0xffffffff s.s.s.1 0xffffffff tcp     gt 1023 eq 1080 secure local inbound
permit s.s.s.1 0xffffffff c.c.c.c 0xffffffff tcp/ack eq 1080 gt 1023 secure local outbound

# Connection from the Firewall to the External Notes Server on the DMZ
permit n.n.n.1 0xffffffff p.p.p.p 0xffffffff tcp     gt 1023 eq 1352 nonsecure local outbound
permit p.p.p.p.0xffffffff n.n.n.1 0xffffffff tcp/ack eq 1352 gt 1023 nonsecure local inbound
```

To use this setup you also have to configure your internal Notes server by editing the SOCKS proxy field in the Notes Server document to reflect the address of the firewall. If your version of Notes does not support SOCKS, you can try the tcp_relay setup as described in Appendix D, “Sample NNTP Relay Program” on page 279. In this case, you need to configure the internal Notes server to replicate to the firewall instead of the external Notes server, and configure tcp_relay only in one direction by using the following line in the file /etc/tcp_relay.cfg:

```
client-ip    1352   domino-ip  1352
```

6.13 ident

The ident protocol is used to authenticate the user of a connection. The ident client sends a request containing a user ID and information about a TCP port that the user ID appears to be using. The ident server responds positively if the user really is using the port. In practice there are very few platforms that implement the ident server as standard and it has no value on systems that do not require users to log in (such as Windows 3.1, Windows 95 and OS/2).

The server (the machine who's connection is being authenticated) uses tcp/113 and the client (the machine that is trying to authenticate the connection) uses a nonprivileged port.

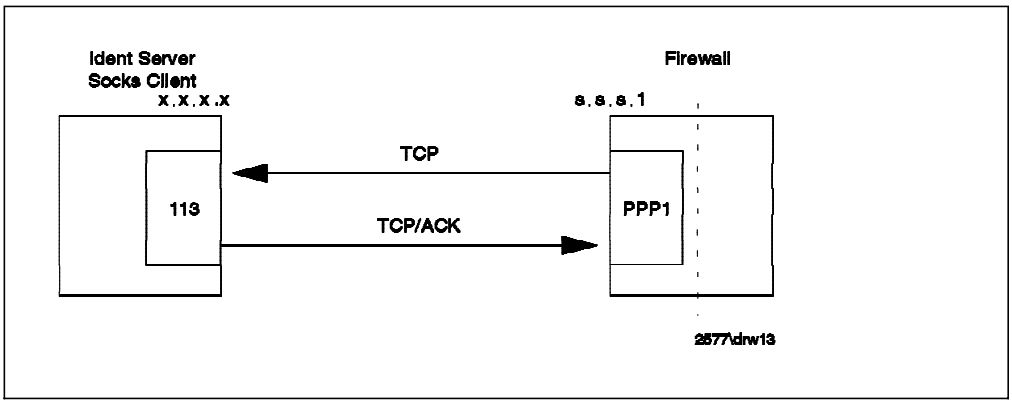


Figure 95. ident, for Authentication from the Firewall to the Secure Network

If you are planning to use SOCKS with user authentication, you should enable this service from the firewall to the secure network. Figure 96 shows you this connection.

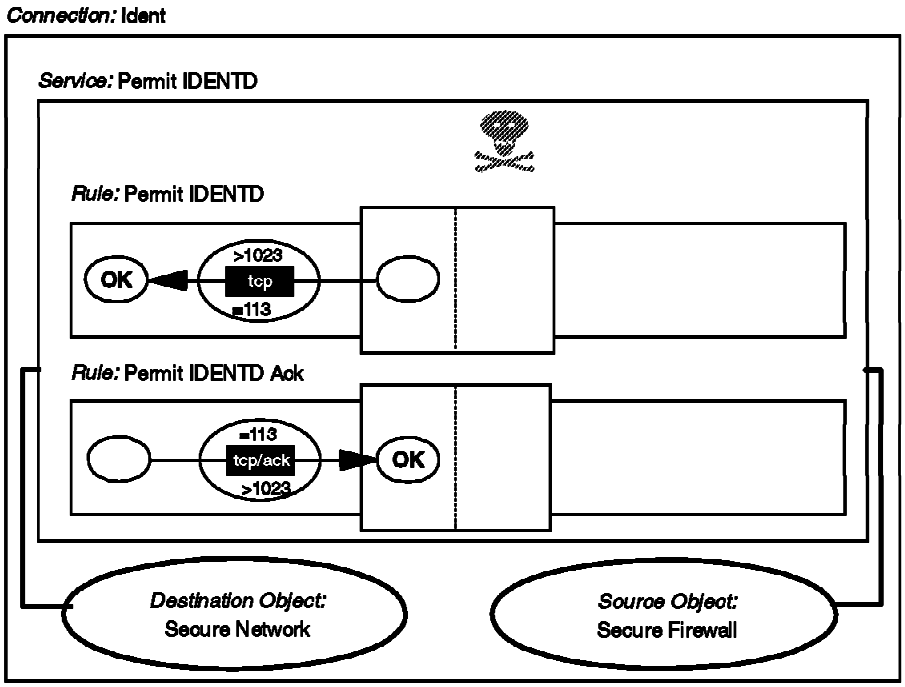


Figure 96. Ident from SOCKS to Secure Network

6.14 Syslog

Syslog is a service used in order to manage log messages from multiple machines in a centralized place. It is mostly used by UNIX machines, but now is gaining acceptance in other devices, such as hubs. Syslog should not be allowed outside the secure network, because it can be used for a denial of service attack (if you have a DMZ, it is safe to use syslog between servers in the DMZ and a logging host within the secure network).

It is also convenient to log firewall messages in another machine. We will describe this in Chapter 14, “Logging” on page 251.

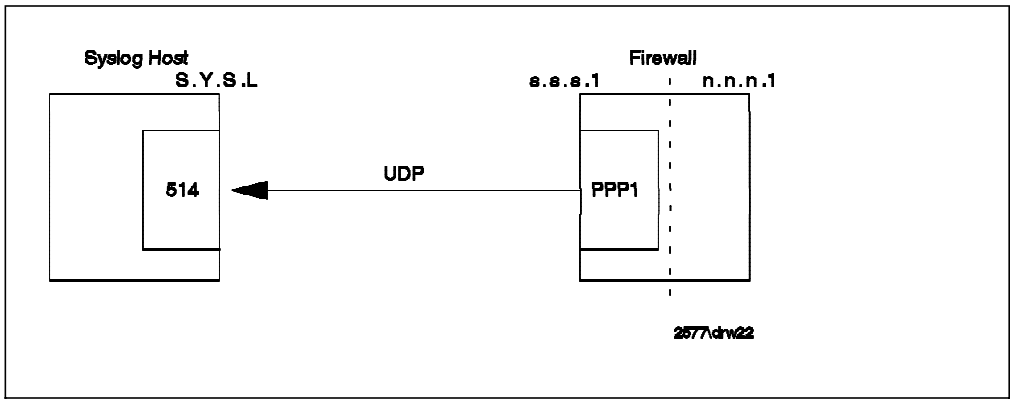


Figure 97. Syslog, from the Firewall to a Logging Host

Normally you send the firewall logs to a machine in the secure network. Syslog uses UDP protocol to send the logs. Figure 98 shows the needed connection to do that.

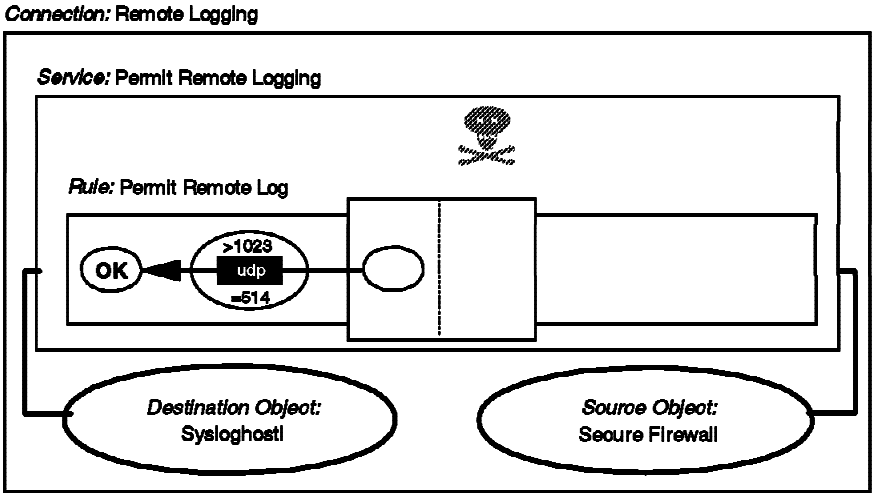


Figure 98. Syslog from Firewall to Secure Log Host

6.15 Traceroute

Traceroute is a service that is useful in allowing network administrators to track the path that an IP packet is following in order to reach its final destination. It works by sending UDP packets from one high port (port number > 1023) to another high port. It selects a free UDP Port (in our tests on AIX 4.2.1 we only saw packets from ports greater than 40000) and starts to send packets to different high ports (in AIX 4.2.1 it starts by default from port 33434, unless you specify this with -p).

In order to discover the path, it plays some tricky games with the TTL value of the packet (this field must be decremented by routers everytime they forward the packet). First it sends a UDP packet with TTL=1, so the first router gets the

packet, decrements the TTL field, and then discards the packet because the TTL reached 0. After discarding the packet, the router sends an ICMP TTL exceeded message to the sender, so the sender learns the address of the first hop.

Then it uses a TTL value of two, and it gets the second router address. It keeps getting router addresses with TTL exceeded messages until the packet reaches the destination host. Once the destination host receives the packet, it realizes that it doesn't have any service on the high port, and so it sends an ICMP port unreachable message to the sender.

From this description, you can see that using traceroute involves several UDP packets flowing from the sender to the destination, ICMP TTL exceeded messages flowing from the routers to the sender, and finally an ICMP port unreachable message from the destination to the original sender.

Traceroute from the Firewall:

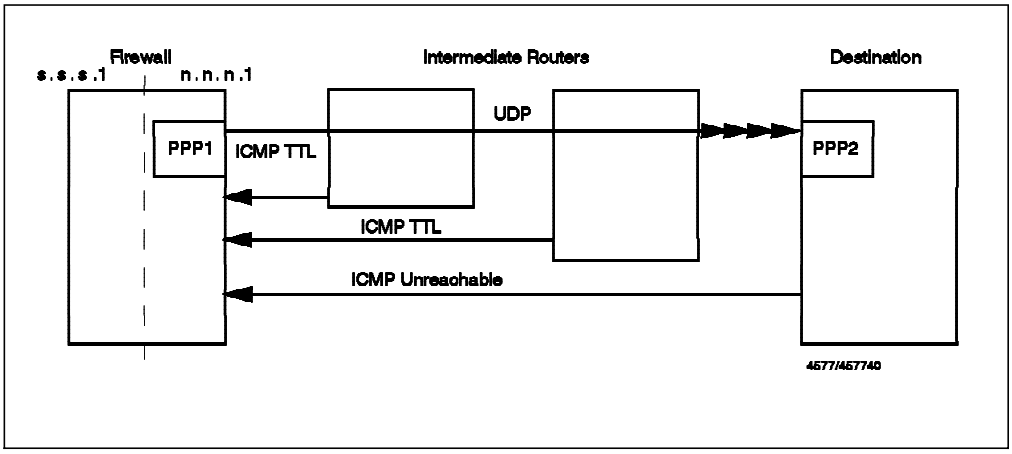


Figure 99. Traceroute Session from the Firewall

This configuration can be safely permitted. In order to do this you must send high UDP packets and accept ICMP TTL and port unreachable messages. Figure 100 on page 120 shows you the connections needed from the secure network to the firewall secure interface.

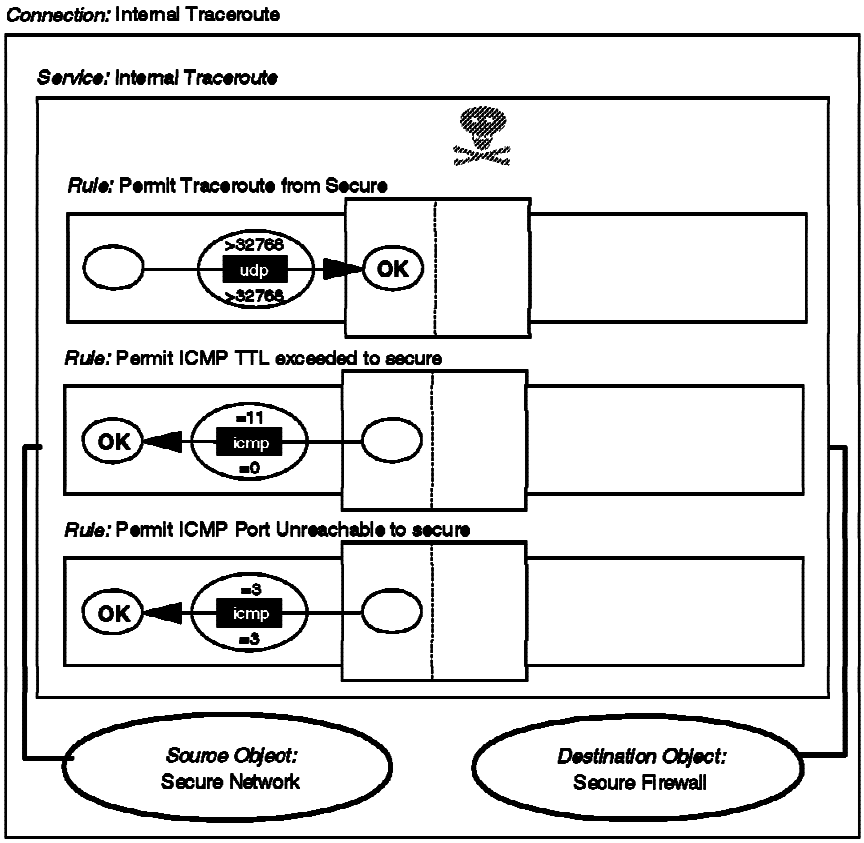


Figure 100. Traceroute from Secure Network to Firewall

Figure 101 on page 121 shows you the connections needed from the firewall nonsecure interface to the world.

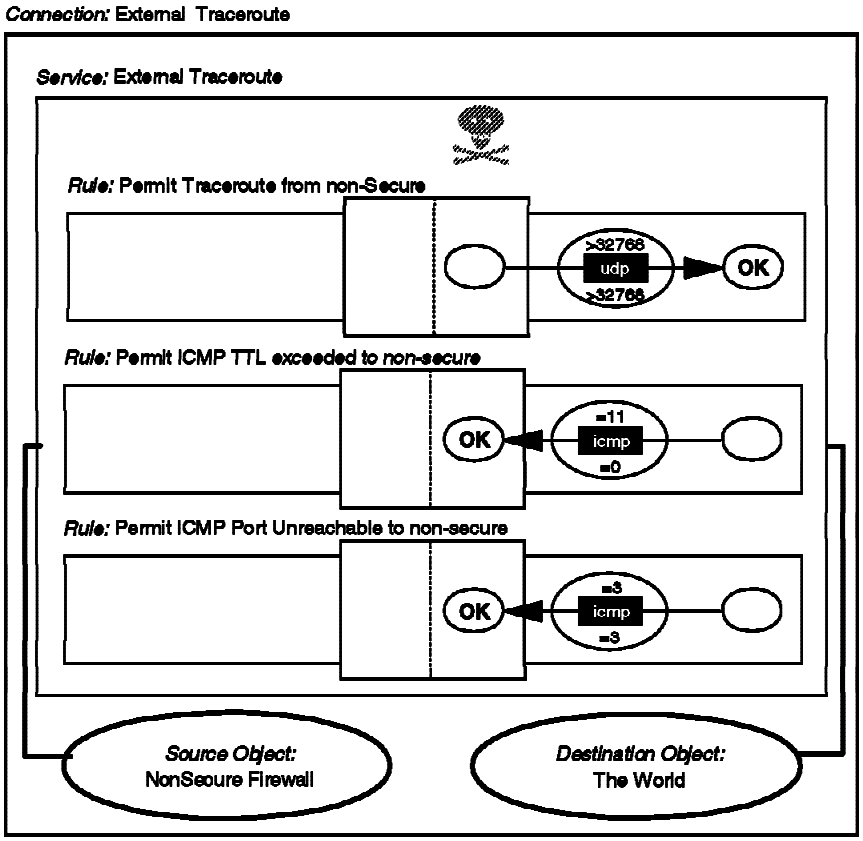


Figure 101. Traceroute from Firewall to Nonsecure Network

Note: The ICMP rules are not necessary when the consolidated ICMP connection (see "Consolidated ICMP Connection" on page 130) is implemented.

The connections implement the following filter rules:

```
# Traceroute to NSN. Outgoing UDP Packets
permit n.n.n.1 0xffffffff 0 0 udp gt 32768 gt 32768 nonsecure local outbound
# Traceroute to SN. Outgoing UDP Packets
permit s.s.s.1 0xffffffff 0 0 udp gt 32768 gt 32768 secure     local outbound
# Traceroute. Reply from the Routers
permit 0 0 n.n.n.1 0xffffffff icmp eq 11 eq 0 nonsecure local inbound
permit 0 0 s.s.s.1 0xffffffff icmp eq 11 eq 0 secure     local inbound
# Traceroute. Reply from the final node.
permit 0 0 n.n.n.1 0xffffffff icmp eq 3  eq 3 nonsecure local inbound
permit 0 0 s.s.s.1 0xffffffff icmp eq 3  eq 3 secure     local inbound
```

Traceroute from Internet to the Firewall: In order to enable this service, you must allow outgoing ICMP port unreachable messages. You may not want to do this because it would be useful to an attacker as a fast way to discover which services you are providing. You can safely allow traceroute from the secure network.

We found out that Windows NT Version 4.0 implements the traceroute command using echo request and echo reply messages. The command is tracert.

6.16 Network Management Sessions

You may want a network management application (such as TME 10 NetView) to be able to monitor across the secure/nonsecure network boundary. Most such applications use the Simple Network Management Protocol (SNMP) for network polling, plus ICMP echo (PING) requests to ascertain whether devices are available or not.

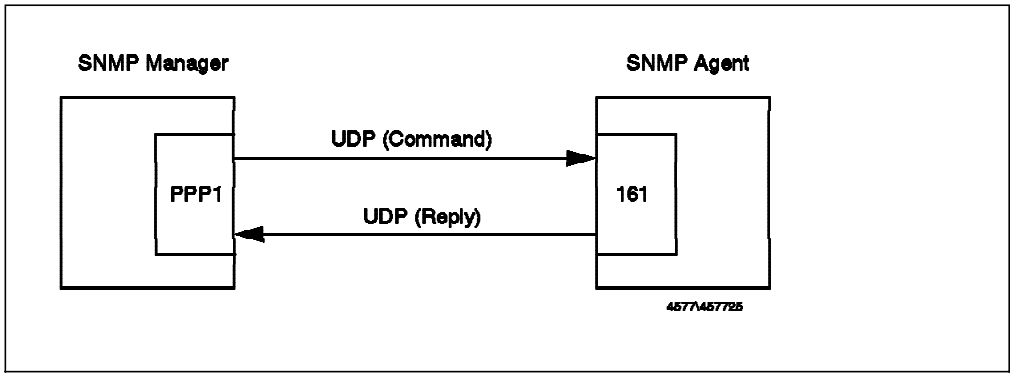


Figure 102. SNMP Manager Querying SNMP Agent

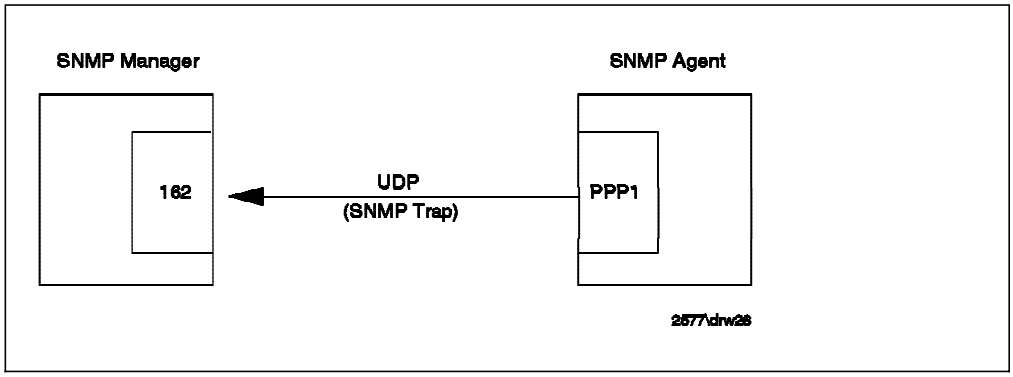


Figure 103. SNMP Agent Notifying SNMP Manager

The following connections will enable an SNMP-based network manager (NV6000) in the secure network to monitor the firewall. The rules used in these connections are not part of the standard set.

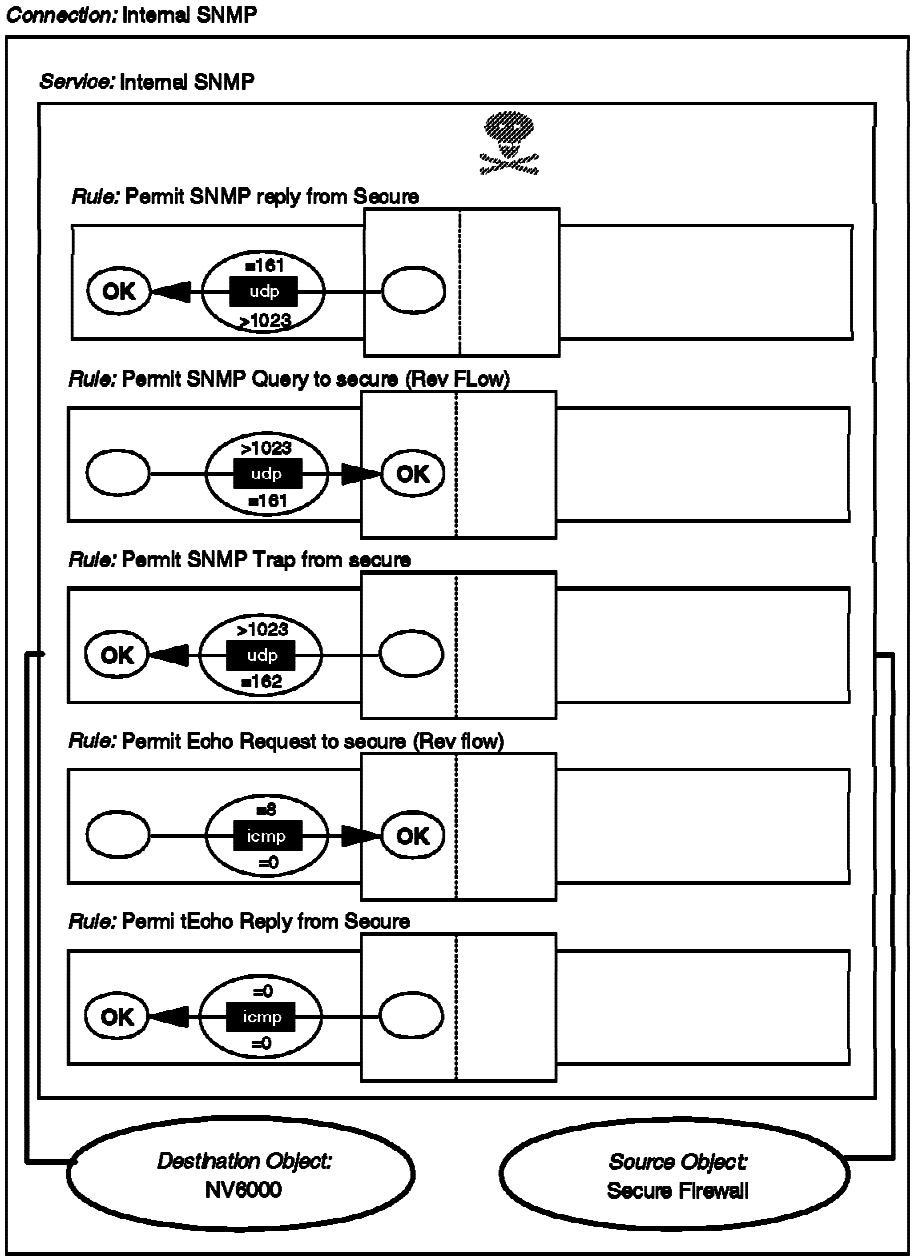


Figure 104. Network Management Sessions

Note: The ICMP rules are not necessary when the consolidated ICMP connection (see “Consolidated ICMP Connection” on page 130) is implemented.

The first two rules allow SNMP GET, GETNEXT and SET requests to be sent by the manager node to the SNMP agent on the firewall and responses from the firewall. The third rule allows the firewall to send an SNMP trap to the manager node (the manager receives traps by listening on UDP port 162). The final rules allow an ICMP echo request from the network manager to the firewall and an echo reply from the firewall to the network manager.

Note that care should be taken with SNMP if you have any devices that allow update via SNMP SET. Most workstations and routers allow SNMP GET requests only, so there is little damage a cracker can do. However, some devices,

notably LAN hubs, allow remote control and configuration functions via SNMP SET requests. SNMP security is imposed by the agent which limits manager access based on the manager IP address and a community name field. Both of these fields are carried in clear in the UDP packet, so it is not difficult for an attacker to fool the agent into giving him full control.

The following filter rules are implemented by the above connection:

```
# SNMP Get, GetNext and Set from Network Manager to Firewall
permit N.V.6.K 0xffffffff s.s.s.1 0xffffffff udp gt 1023 eq 161 secure local inbound
permit s.s.s.1 0xffffffff N.V.6.K 0xffffffff udp eq 161 gt 1023 secure local outbound

# SNMP Traps from Firewall to Network Manager
permit s.s.s.1 0xffffffff N.V.6.K 0xffffffff udp gt 1023 eq 162 secure local outbound

# Ping from N.V.6.K to Firewall
permit N.V.6.K 0xffffffff s.s.s.1 0xffffffff icmp eq 8 any 0 secure local inbound
permit s.s.s.1 0xffffffff N.V.6.K 0xffffffff icmp eq 0 any 0 secure local outbound
```

6.17 Archie

Archie is a service useful for searching programs in anonymous FTP servers. The archie servers maintain a database of program names, locations and descriptions.

Archie is a UDP protocol that uses port 1525 for the server and nonprivileged ports for the client. There are several ways in which users can use this protocol, circumventing UDP packets. They can use a WWW gateway, a Telnet client, or access archie through mail.

In order to use the WWW gateway, they just need HTTP access (see 6.8, “HTTP - World Wide Web Sessions” on page 107). In this case, they will just have to open a Gateway Form Document, like http://www-ns.rutgers.edu/htbin/archie (see Figure 105 on page 125).

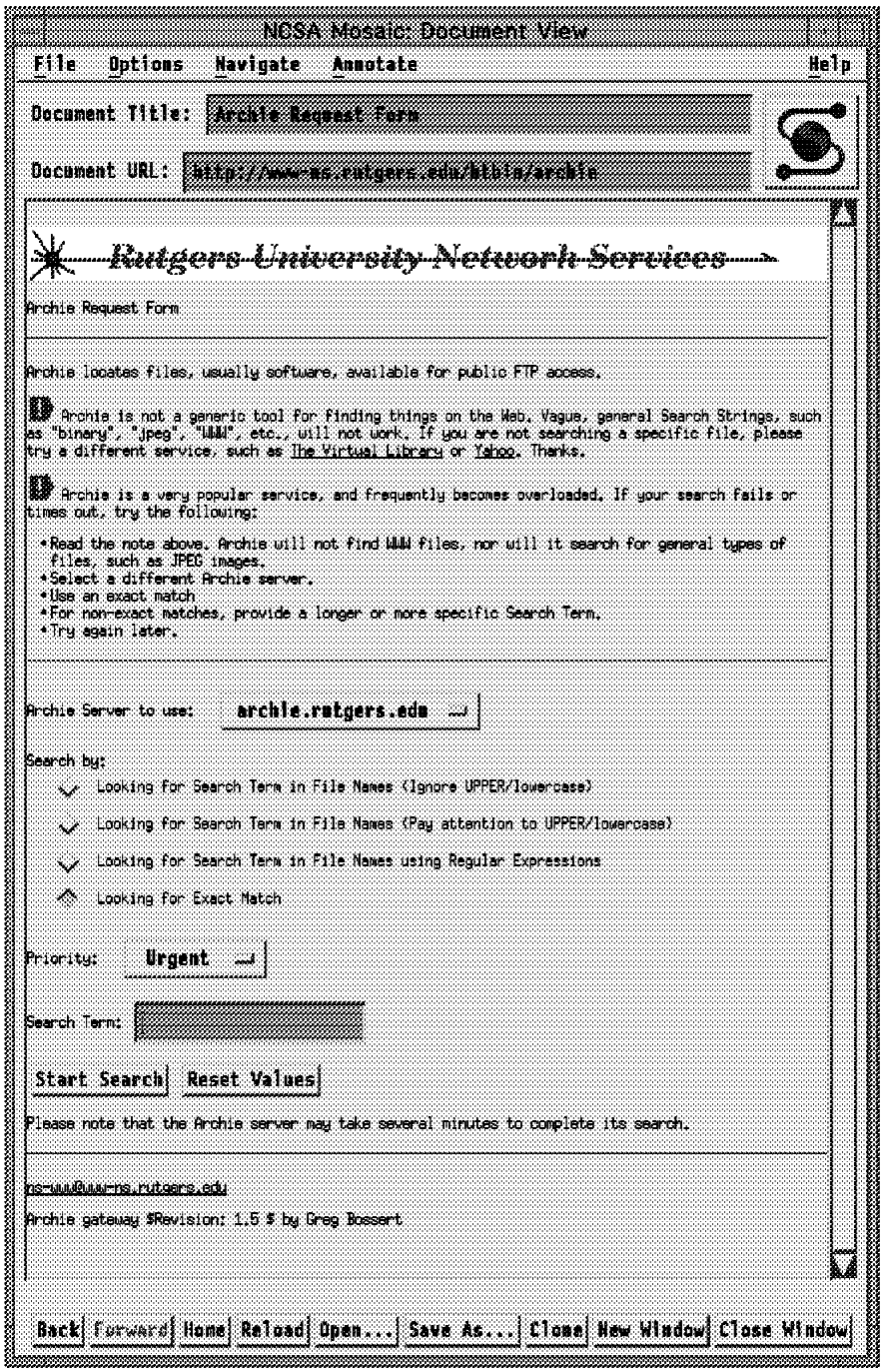


Figure 105. Archie Using an HTTP Gateway

In order to use the Telnet service, the user needs outgoing Telnet access. They telnet to an archie server, log in as user ID archie, and then they can do any query.

```
rs600020:/adrian/bin > rtelnet archie.rutgers.edu
Trying 128.6.21.13...
Connected to archie.rutgers.edu.
Escape character is 'Á]'

Solaris 2 (dogbert.rutgers.edu) (pts/7)

login: archie
# Message of the day from the localhost Prospero server:

          Welcome to the Rutgers University Archie Server!
 ____________________________________________________________________________

 7/31/95  -  The Rutgers Archie server has been moved to its new home;
             A Sun SPARCserver 20/71!  Please let us know if you
             encounter any problems.
 ____________________________________________________________________________

          Type "help" for information on how to use Archie.


# Bunyip Information Systems, Inc., 1993, 1994, 1995

# Terminal type ¦aixterm' is unknown to this system.
# ¦erase' character is ¦Á?'.
# ¦search' (type string) has the value ¦sub'.
archie> whatis "icmp echo"
ping                    PD version of the ping(1) command. Send ICMP ECHO
                        requests to a host on the network (TCP/IP) to see
                        whether it's reachable or not
archie> quit
# Bye.
Connection closed by foreign host.
```

Figure 106. Sample Archie Session Using Telnet

6.18 WAIS

WAIS (Wide Area Information Servers) is a service used to search through large text databases. It uses TCP as a transport layer; servers use port 210, and the clients use any nonprivileged port.

One easy way to provide WAIS service is through an HTTP gateway. The client just selects a URL that provides a WAIS service (for example, http://www.ai.mit.edu/the-net/wais.html) and submits the query. In this way, if you are already providing HTTP access you can provide, WAIS access for free (you just need to educate your users).

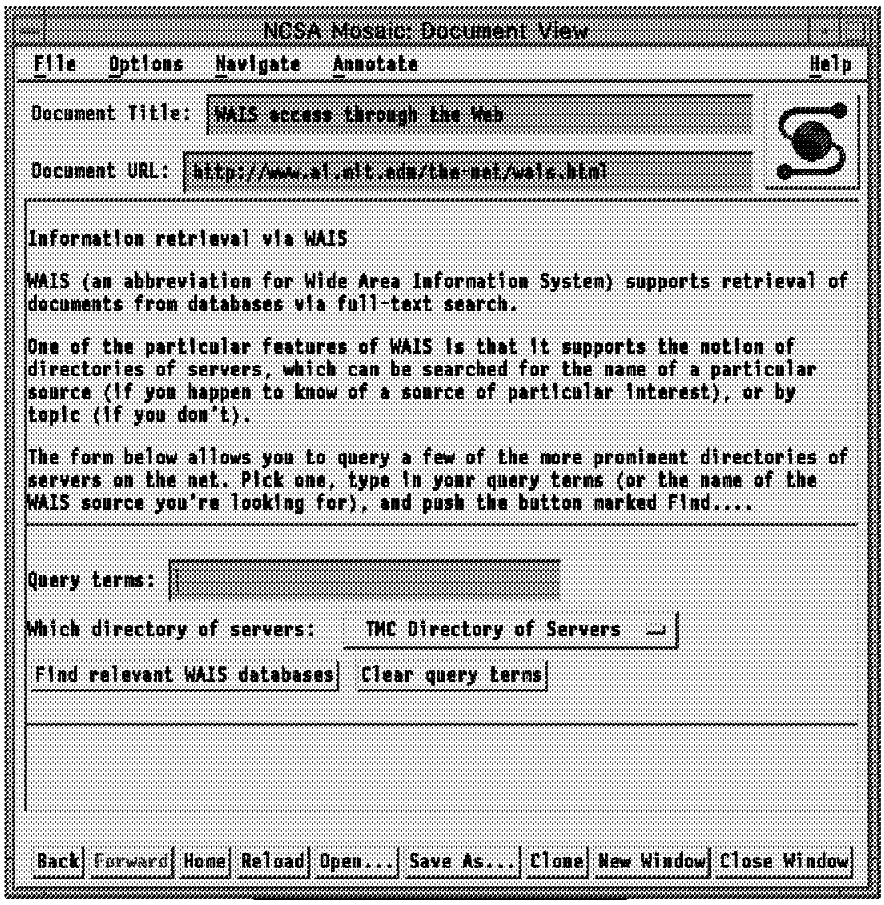


Figure 107. WAIS Using an HTTP Gateway

You could also allow SOCKSified dedicated clients. To permit this you will need the following connection that allows the firewall to contact the WAIS server on port 210 (note that the rules used in this service are not part of the standard set and so must be manually defined).

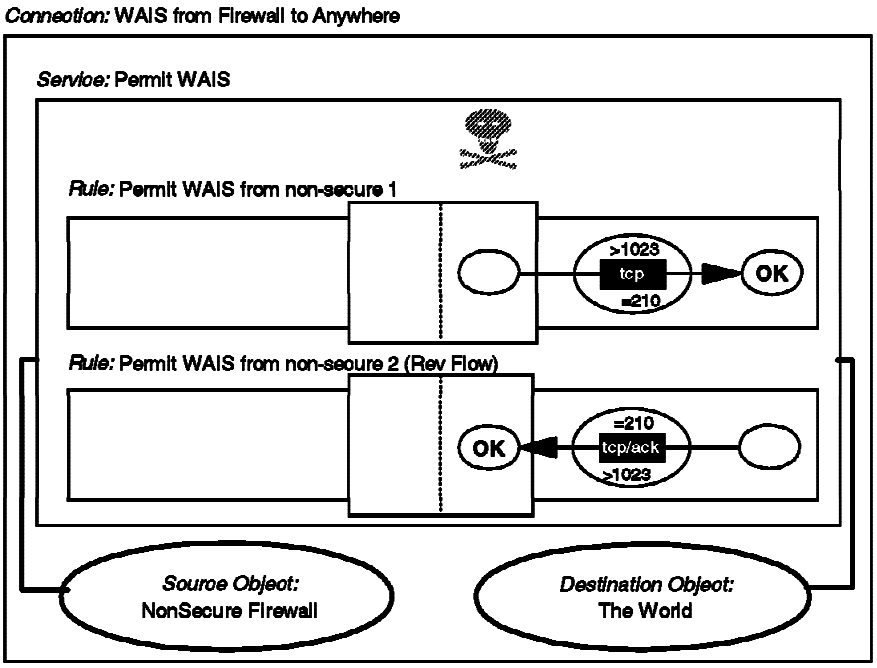


Figure 108. WAIS from SOCKS to the Nonsecure Network

```
# Connection from the client to the Socks Server
permit s.s.s.s sm.sm.sm.sm s.s.s.1 0xffffffff tcp     gt 1023 eq 1080 secure local inbound
permit s.s.s.1 0xffffffff s.s.s.s sm.sm.sm.sm tcp/ack eq 1080 gt 1023 secure local outbound

# Connection from Firewall to the Server in the Nonsecure Network
permit n.n.n.1 0xffffffff 0 0 tcp     gt 1023 eq 210  nonsecure local outbound
permit 0 0 n.n.n.1 0xffffffff tcp/ack eq 210  gt 1023 nonsecure local inbound
```

6.19 Finger

Finger is a protocol that is used to determine which users are logged in a system. The server uses tcp/79.

This useful service has two well-known problems, one on the client side and the other on the server side.

The server side may be misused by an attacker in order to gain information about the machine. You will not have this problem, because IBM Firewall 3.1 will, by default, have disabled this service for you. However, it is a good idea to provide some contact information (for example, a phone number that a remote administrator can call), so it is useful to replace the normal finger daemon with something that prints out a fixed message. You can do this simply by updating the finger entry in /etc/inetd.conf, for example:

```
finger stream  tcp     nowait  nobody  /usr/bin/cat cat /etc/myinfo.txt
```

Having set up this limited finger response, you will want to allow finger requests to be received by the firewall. The following connection permits this:

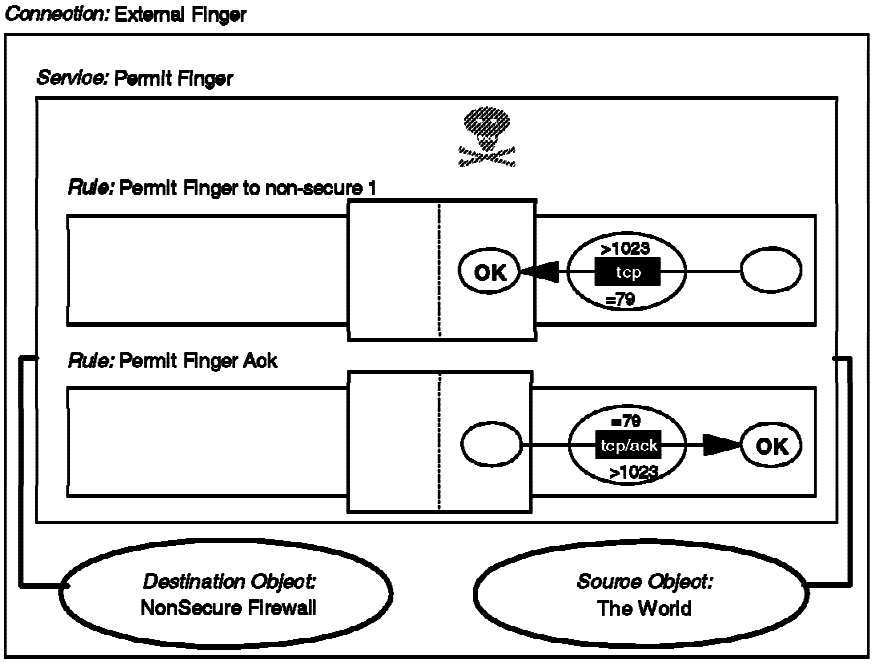


Figure 109. Getting a Finger from the World

```
# Connection to the firewall from external finger client
permit 0 0 n.n.n.1 0xffffffff tcp     gt 1023 eq 79 nonsecure local inbound
permit n.n.n.1 0xffffffff 0 0 tcp/ack eq 79 gt 1023 nonsecure local outbound
```

The problem on the client side is a result of paranoia. If you think that someone is attacking you, finger is a useful tool to try to find out who it is. Some people invoke finger automatically within a script file to gain information when they detect a possible attack. There have been a lot of cases in which the attackers have replaced the finger daemon by a program that will send you an endless stream of data. If you execute finger within a script, this can quickly fill your file system.

The AIX finger client does not control these types of attacks, so if you decide to use finger to gain information we suggest you pipe the output through the `head` command, to limit the amount of output. (In our tests, the AIX finger filled a 300 MB file system, so this is a real problem.)

The finger client must also be protected from nonprintable characters introduced by the server. We recommend you use some replacement for the standard client, such as safe_finger, which is included in the SATAN package.

Finally, you have to be careful if you provide a finger service and use the finger client for obtaining information. You could trigger a denial of service case (also called finger war), in which someone fingers at you and then you finger them back, endlessly (mutual paranoia).

6.20 Filtering Specific ICMP Messages

At the start of this chapter we suggested that a simple but robust approach to ICMP is just to block all ICMP messages. Many of the ICMP messages may be misused by an attacker, but there are also many which are benign, and which contribute to the smooth running of your applications.

In 3.1.2, "An Introduction to ICMP Packets" on page 24 we described the different types of ICMP packets at some length and included a recommended filtering strategy for each one. The following connection will implement those recommendations.

Consolidated ICMP Connection:

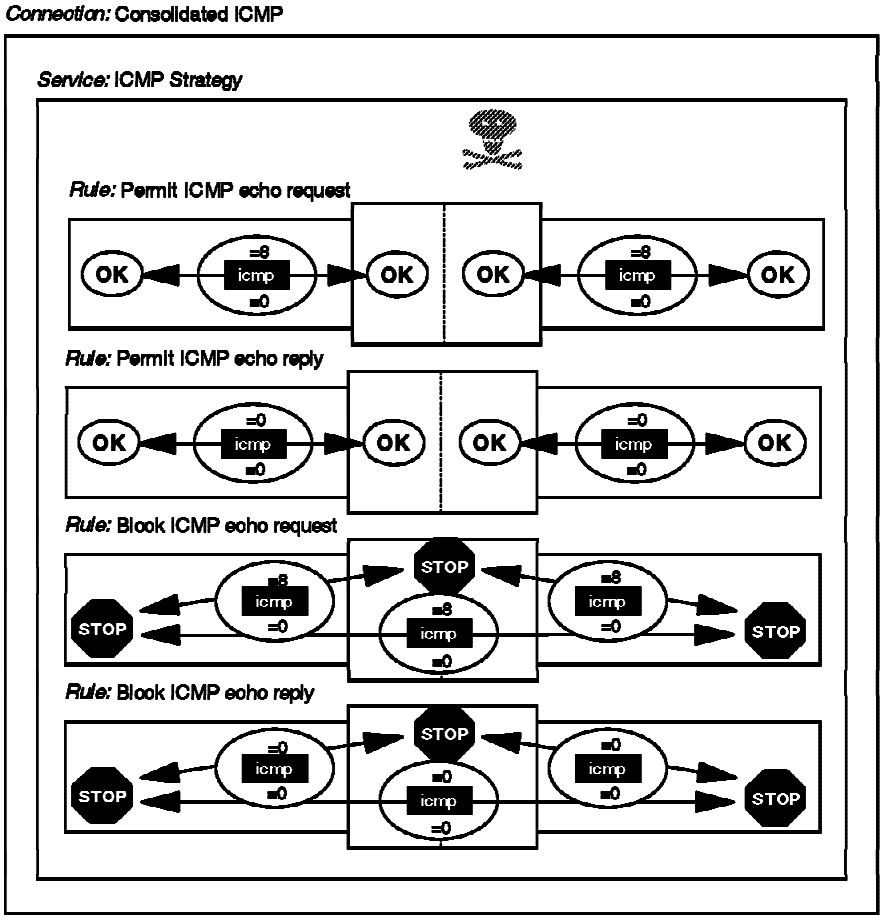


Figure 110. Consolidated ICMP Control Connection (Part 1)

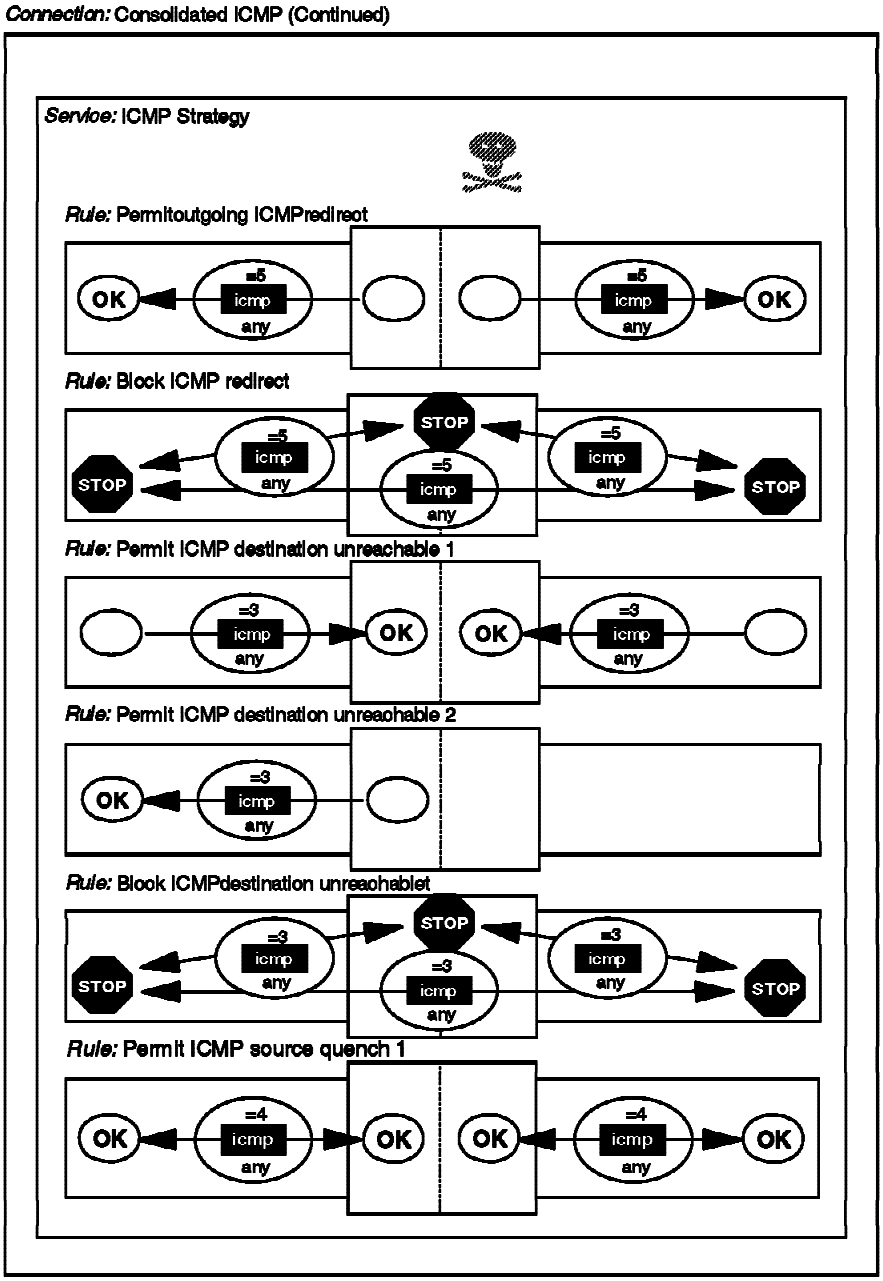


Figure 111. Consolidated ICMP Control Connection (Part 2)

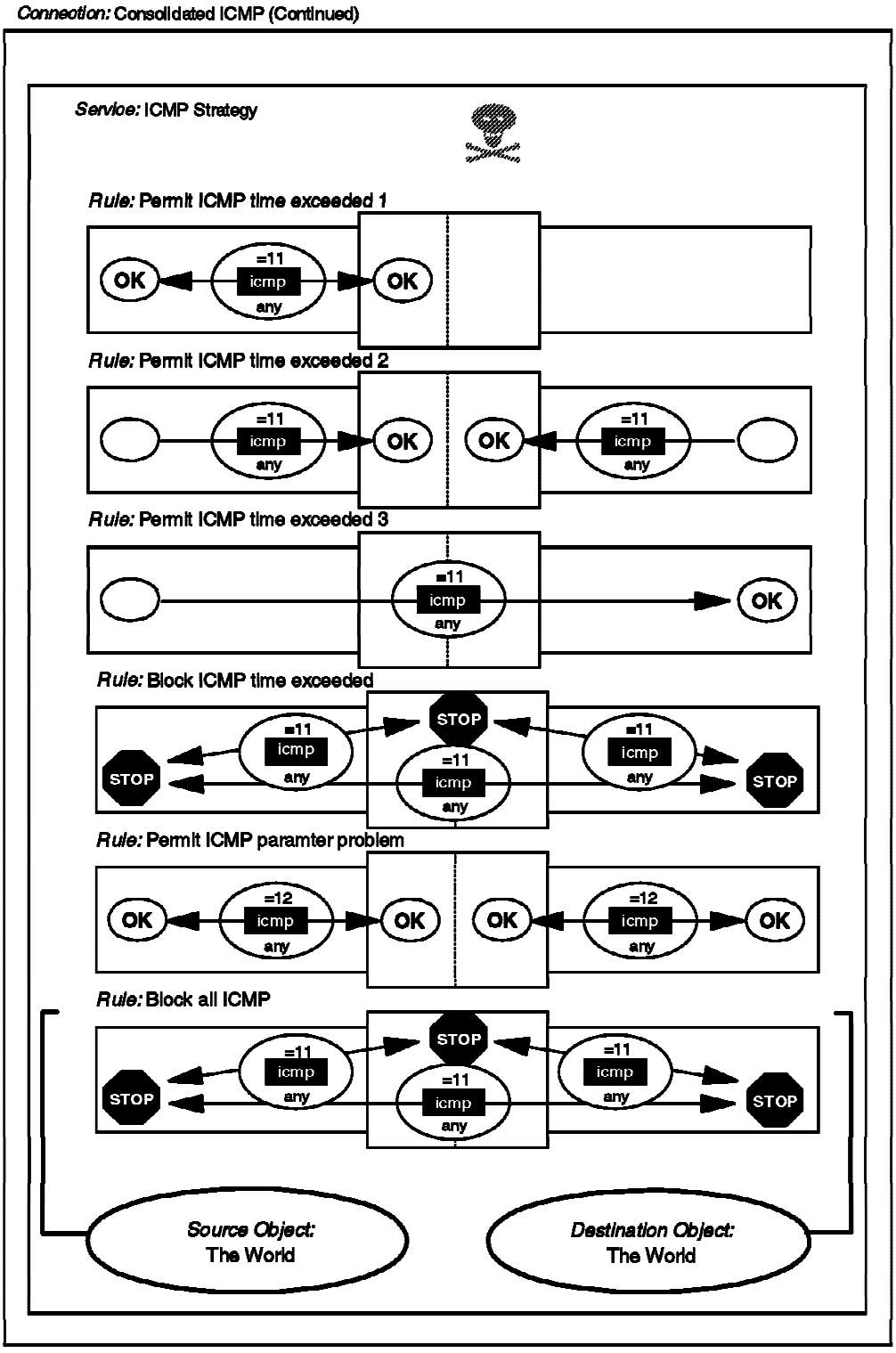


Figure 112. Consolidated ICMP Control Connection (Part 3)

6.21 Other Protocols

As the Internet is constantly evolving, there are always new protocols, so we want to stress the last but most important rule of the firewall. This default rule is always added by IBM Firewall:

```
# Everything that is not explicitly allowed is denied.
deny 0 0 0 0 all any 0 any 0 both both both
```

6.22 Secure Terminal Emulation

In some cases it is necessary to allow users to access firewall-protected machines from the outside. An ideal solution would be to use Secure IP Tunnel, but unfortunately it is not always available.

This forces the following scenario upon us:

- Users log in to the firewall and authenticate themselves reliably.
- Users invoke Telnet on the firewall and log in to the internal machine they need.

In general, it is extremely undesirable to let an external user log in to the firewall itself. One way to eliminate potential problems is by restricting this access to the barest minimum (which is a standard feature of the IBM Firewall proxy server) plus ensuring that login authentication is sufficiently strong.

To achieve it, we recommend the following:

1. Using a modified Telnet, which supports strong user authentication. To provide necessary strength, it is recommended to use a random challenge approach. An example of such would be the SecureNetKey card, the SecureID card or S/KEY (the IBM Firewall Telnet proxy supports all of them). So when you attempt to log in, you telnet to the firewall, type in your user name and receive back a challenge of 8 random digits. You type those into your card and it gives you another set of 8 digits. Now you type those as a response to the challenge. If the response is correct (that is, what was expected), you log in successfully; otherwise, the login is denied. See Appendix E, “IBM Firewall Related Products” on page 285 for details.
2. Giving users a restricted shell with a bare minimum number of commands allowed (two should be enough: telnet and exit).

From this point on, a user (after logging in successfully) can telnet to the internal system.

This approach has the following two flaws:

- User login to the firewall itself is secure, but the next step (logging in to the internal machine from the firewall) may expose confidential data (host name, login name and login password) to an eavesdropper.
- It is vulnerable to TCP session hijacking.

To combat these problems, you need to go one step further and provide encryption between the firewall and the remote user. The following tools are available:

- Encrypted Session Manager (ESM). Before telneting to the firewall you start ESM in local mode, and then you log in; after that you start a copy of ESM on the firewall in server mode. This establishes an encrypted session between you and the firewall, and subsequent logins to internal hosts are protected from eavesdropping. Clearly, you would have to add `esm` to the list of permitted commands and possibly put it in the user profile (which should not allow the user to be able to edit).

  ESM was developed in the US, so cryptographic export restrictions apply. You can find more information about ESM at http://www.epm.ornl.gov/~dunigan/cfsesm.txt.

- Secure Shell (SSH). You can substitute rlogin and rlogind with their counterparts from the SSH distribution. We suggest that they be modified to support the authentication card. These programs are a drop-in replacements for the r utilities (rlogin in this case) that will provide you with encrypted login to the firewall. It uses RSA for key exchange, and it can use IDEA, DES or another cipher for encryption. SSH can be used freely by anyone for any purpose. Permission is required to sell it commercially. For more information about SSH, see http://www.cs.hut.fi/ssh. It was written by Tatu Ylonen, from the Helsinki University of Technology, Finland.

- Secure Telnet (STEL). This provides link encryption before the login starts, guarantees data integrity, and is easily modifiable to support the authentication card. It is a drop-in replacement for telnet and telnetd. It can use DES, Triple DES or IDEA for encryption. In order to exchange the session keys, it uses a modified version of the Diffie Hellman algorithm.

  For more information about STEL, see http://www.epm.ornl.gov/~dunigan/stel.txt

The key points here are as follows:

- Ensure that user login to the firewall cannot be compromised by insisting on strong authentication.
- Ensure that user activity on the firewall is always logged. .
- Ensure that user activity on the firewall is restricted as much as possible.
- Ensure that the user can establish an encrypted channel between their end and the firewall before connecting to an internal host.
- Data integrity is achieved by running cryptographic checksum on every packet between the user and the firewall. In case of encrypted connection, this may not be necessary, but it never hurts.

For a complete list of cryptographic software, refer to http://www.cs.hut.fi/crypto/software.html, or http://www.epm.ornl.gov/~dunigan/security.html.

6.23 Using Security Policy

Some of the rules mentioned in this chapter can be set up using the Security Policy panel. If you select Security Policy from the main GUI panel (see Figure 24 on page 45), you get the screen shown in Figure 113 on page 135.

.

Figure 113. Security Policy Panel

From this panel you can set up the rules for DNS, broadcast and SOCKS similar to the ones mentioned in this chapter.

The advantage of using the Security Policy panel over the the the method described in Chapter 5, “IBM Firewall 3.1 Rule Base” on page 53, is that is standard and provides a common way of setting up basic filters in all the IBM Firewall installations. The disadvantages are that is not possible to control the rule placement and the rules could be too general or too specific.

The Security Policy panel can also be used to:

- close the nonsecure interface in case you suspect the firewall is under attack
- open the firewall: no restriction to the traffic (use only for testing as this opens the door of the secure network to everybody)
- enable/disable the Telnet and/or FTP transparent proxies.

Chapter 7. Secure IP Tunnel

In this chapter we will discuss the secure IP tunnel. It is a mechanism provided by IBM Firewall in order to allow secure communications between secure networks over a nonsecure intervening network like the Internet. It constructs a virtual private network (VPN) between two different sites providing authentication and encryption.

7.1 Secure IP Tunnel Standards

Secure IP tunnel products have existed in the market for years. However, due to the lack of an IP security standard, these products were proprietary in nature (that is, they can only establish secure tunnels with their own product). IBM and other organizations are actively involved in the development of interoperability standards that will allow firewalls from different manufacturers to establish tunnels between them. The basis for these standards is a group of RFCs that are being guided through the standards process by the Internet Engineering Task Force (IETF) IP Security Protocol (IPSec) working group. You can find the charter for the IPSec group, plus links to the IPSec RFCs at http://www.ietf.org/html.charters/ipsec-charter.html.

Basically, IPSec is a network layer security protocol which will provide authentication and encryption to IP datagrams. IPSec defines two new mechanisms to achieve these security objectives. They are the IP Authentication Header (AH) and the IP Encapsulating Security Payload (ESP). The details of AH and ESP can be found in RFC 1286 and RFC 1287 respectively, and the overall architecture of IPSec is described in RFC 1285.

IBM Firewall 3.1 provides three kinds of secure IP tunnels to cater for different situations:

1. An IBM tunnel is used between IBM Firewalls. It uses IPSP (IP Security Protocol) which is an IBM unique protocol. It features an automatic key update mechanism, using UDP port 4001. Under this scheme, a new encryption key is generated at regular intervals and communicated through the tunnel encrypted under the current key.
2. A manual tunnel uses the IPSec standard and can be established between an IBM Firewall and:
   - An IPSec compliant firewall
   - An AIX IPSec client (it is supplied with IBM Firewall 3.1 for AIX software)

   In fact, an AIX IPSec client can establish a manual tunnel with any IPSec compliant host. The operation of an AIX IPSec client is described in 7.4, “Using the AIX IPSec Client” on page 149.

   Manual tunnel currently does not support any key refresh mechanism. Hence, when configuring a manual tunnel, it will be necessary to inhibit key updates.
3. A dynamic tunnel also uses the IPSec standard, but it is used between an IBM Firewall and a Windows 95 secure remote client. It is configured but not activated until the remote client starts the tunnel. The configuration of a dynamic tunnel is described in 7.7, “Windows 95 IPSec Client” on page 160.

The IPSec RFCs are very broadly based. In order to bring the technology to fruition more rapidly, RSA Data Security, Inc. convened a group of leading firewalls and TCP/IP manufacturers in an initiative called S/WAN. The objective of S/WAN is to demonstrate interoperability using a current draft of the IPSec standards. At the time of writing, IBM Firewall had demonstrated interoperability with other firewall vendors. The result of the interoperability test between IBM Firewall and other vendors is listed in Table 7. You can find more details about the S/WAN initiative under the RSA home page located at http://www.rsa.com.

Table 7. RSA VPN Interoperability Test Results

Vendor/Firewall	**Tested Functions**
Checkpoint FireWall-1	TU, AH+ESP, FTP, PING
FTP OnNet	TR, AH+ESP, Telnet
IBM	TU
Morning Star SecureConnect	TU, AH+ESP
Raptor Eagle	TU, AH+ESP, HMAC+ESP, PING, FTP, Telnet
Secure Computing Sidewinder	TU, AH+ESP, PING, Telnet
SOS Brimstone	TU, AH+ESP, FTP, Telnet, PING
TimeStep PERMIT	TU, AH+ESP, PING, FTP
TIS Gauntlet	TU, AH+ESP
Gemini GTFW-GD	TU, AH+ESP, PING, Telnet, FTP, HTTP

Where:

AH Authentication Header (RFC1826)
ESP Encapsulating Security Payload (RFC 1827)
TU Tunnel Mode ESP
TR Transport Mode ESP
RC5 ESP w/ RC5 Cipher
HMAC Keyed MD5 for Message Authentication

It should be noted that there is no key management protocol specified in the IPSec RFCs (1825 - 1827). AH and ESP are designed to be independent of the key management protocol. However, they can be coupled to any key management protocols via the Security Parameters Index (SPI) in their header.

This independence makes the early deployment of AH and ESP mechanisms possible while the key management protocol standard is developing. The IPSec working group has recently submitted the Internet Key Management Protocol (IKMP) to IETF for consideration as a draft standard. IKMP is based on the Internet Security Association and Key Management Protocol (ISAKMP) and the OAKLEY Key Determination protocol (OAKLEY).

IBM is in the process of implementing IKMP, and it will be incorporated into the future release of IBM Firewall.

7.2 Operation of the Secure Tunnel

The secure IP tunnel relies on symmetric-key cryptography to enforce data security. This means that the firewalls at each end of the tunnel have a *shared secret* in the form of an encryption key known to both of them. Using this key, the secure IP tunnel provides two different types of security:

1. Authentication, in which the sending firewall appends a message authentication code (MAC) to the messages it sends through the tunnel. The MAC is constructed from the message contents and the encryption key using a one-way hash function. The receiving firewall performs the same operation and, if the MAC matches, it knows that the message is authentic.
2. Encryption, in which the data within the message is encrypted using the secure key, so that it cannot be viewed in transit.

Authentication and encryption can be used independently. In fact, you can enable or disable the two features for each different tunnel. A typical scenario will have multiple secure networks (for example, branches of a company that are in different cities) with tunnels between them in order to protect the information. There may be more than one tunnel between a single pair of nodes, which might be useful for different encryption and authentication choices.

For example, your computer department may wish to monitor machines in the financial department using SNMP. In this case, the information itself is not sensitive, but you want to be sure that it is accurate, so you could use a tunnel that provides only authentication.

However, you also want the computer department to send mail to the financial department and you would like to protect this mail from being read in the unsecure network. This would require a second tunnel providing both authentication and encryption.

Figure 114 shows a typical tunnel scenario.

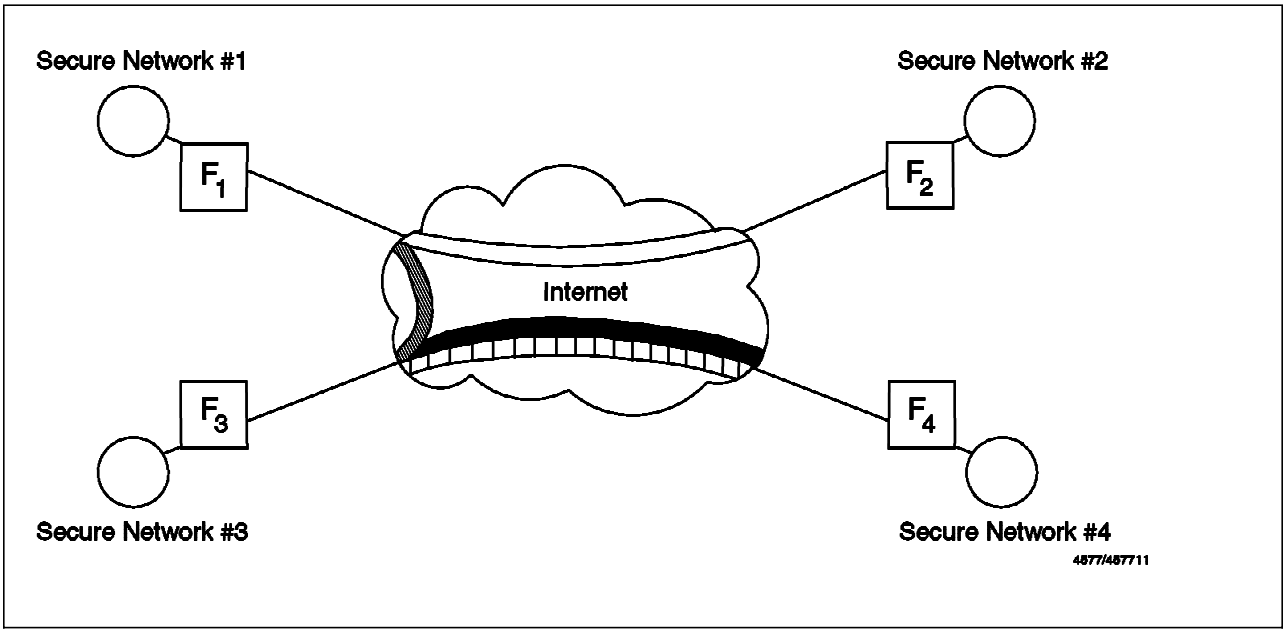


Figure 114. A Typical Secure IP Tunnel Scenario

When a packet has to go from one secure network to another secure network through the IP tunnel, the whole IP packet will be encrypted and authenticated at the first end and sent in a new IP packet to the second end of the tunnel. Note that the packet is not sent using the normal IP protocols (TCP or UDP), but using a special security protocol. In Figure 115 we show a black border around the real IP packet to show that it is being protected in the nonsecure network by the secure IP tunnel.

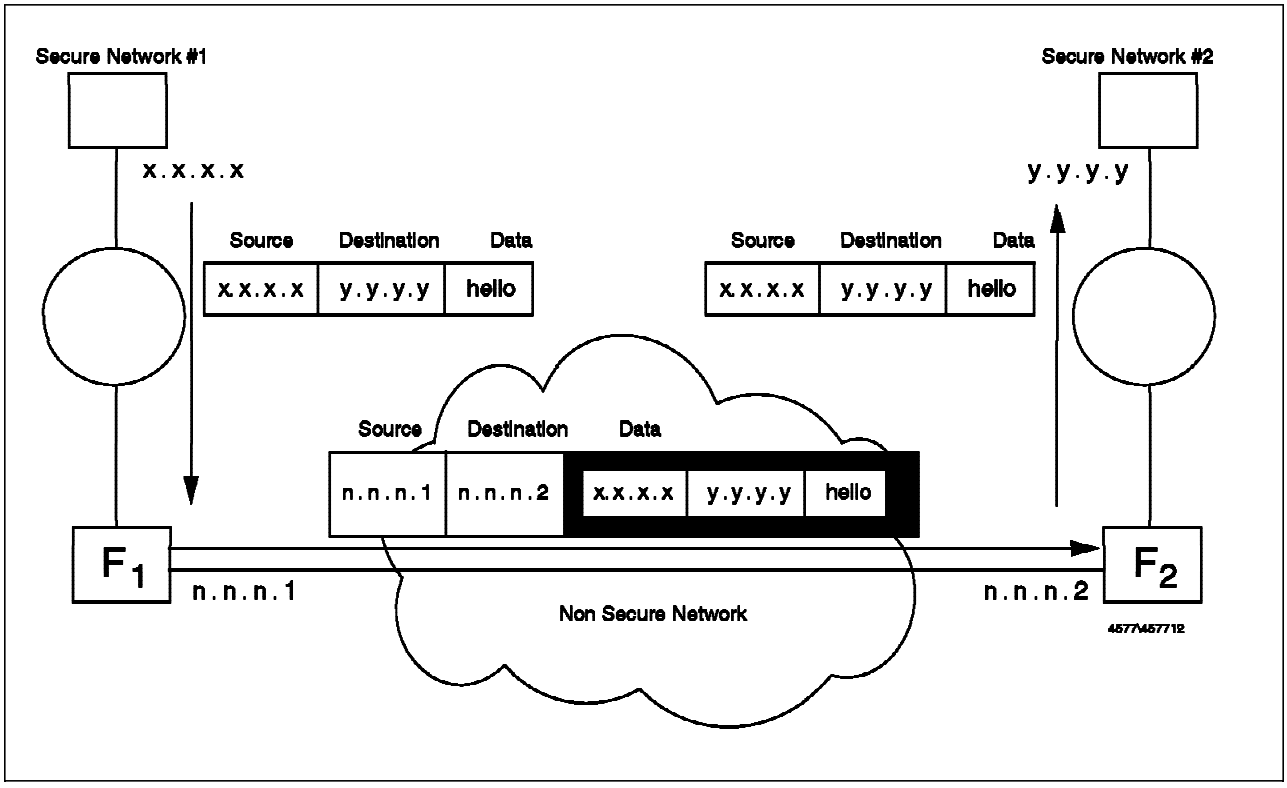


Figure 115. Operation of the Secure IP Tunnel

7.3 Implementing the Secure IP Tunnel - IBM and Manual Tunnels

In order to configure an IBM tunnel or a manual tunnel, you will have to follow these steps:

1. Add the tunnel definition in one node
2. Export the tunnel definition to a set of files
3. Transfer the tunnel definition files to the second node
4. Import the tunnel definition in the second node
5. Activate the tunnel at both ends
6. Specify which protocols you want to tunnel using filtering rules
7. Refresh the tunnel when session keys are expired (for manual tunnel only)

You also have to consider that you need the following prerequisites:

1. IP forwarding enabled in both firewalls
2. Coherent IP addresses in both secure networks (for example, you cannot use the same private IP addresses)

3. Proper routes in the clients (they point to the firewall for addresses in the other secure network)
4. Name resolution for the remote networks (this is important if you want to pass hidden DNS information through the tunnel)

We will describe each implementation step in turn.

7.3.1 Adding the Tunnel Definition in One Node

First you will have to define, using the GUI, the characteristics of the tunnel: the tunnel type, the addresses of both ends of the tunnel, the authentication/encryption desired, and the parameters for the session key. Figure 116 shows the GUI screen.

At this point it is very important to double-check the target of the tunnel. When you later import this at the second end, there is no validation against the local addresses.

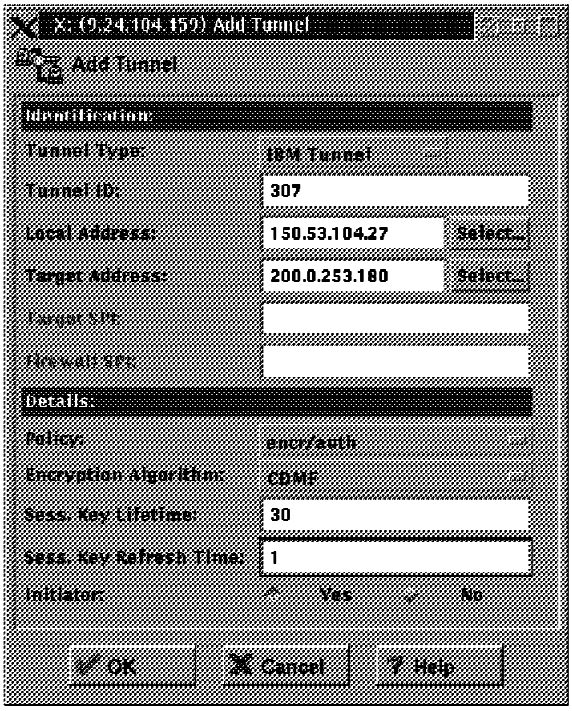


Figure 116. Adding the First Tunnel Definition

In the tunnels definition, the following fields have special importance:

- Local Address: IP address of the nonsecure interface of the local firewall.
- Target Address: IP address of the nonsecure interface of the remote firewall.
- Tunnel Type: if you defined an IBM tunnel, the other firewall must be an IBM firewall. However, if the other firewall is an IPSec compliant host, you should choose the manual tunnel.
- Target SPI: this field is required only if the Tunnel Type is Manual Tunnel. The definition of SPI is described in RFC 1825. Basically, the SPI in conjunction with the target address will uniquely identify the set of security information (such as encryption key(s), key lifetime, etc.) for your tunnel

partner. You should check with the tunnel target and obtain an unassigned SPI from it.

- Session Key Lifetime: since the session key for a manual tunnel remains the same throughout the tunnel lifetime, this value should not be too large. It is because the longer the key remains the same, the higher the chances for someone to crack the key.

7.3.2 Export the Tunnel Definition to a File

Next you will have to export this tunnel definition to a file using the GUI. In the tunnel list, you select the tunnels that you want to export. It will generate in the specified directory three files: fwexppolicy, fwesppolicy.3.1 and fwexpmctx.manual.

It should be noted that the specified directory has to be an empty directory. Otherwise, the export operation will fail. So if you are going to have tunnels between different pairs of nodes, you should create different directories for each pair of nodes.

Figure 117. Exporting the Tunnel Definition

7.3.3 Import the Tunnel Definition in the Second Node

Now you have to take the files from the first firewall to the second firewall. Currently, IBM Firewall does not provide any mechanism to do this transfer. The files contain the encryption key for the secure tunnel, so you should devise a secure way to transmit them. For example, we show an example of sending the files within an encrypted mail message, using Pretty Good Privacy (PGP) (see 7.6, “Using PGP to Distribute the Tunnel Definitions” on page 158).

When you have received the files, place them in a directory and import the definitions (see Figure 118).

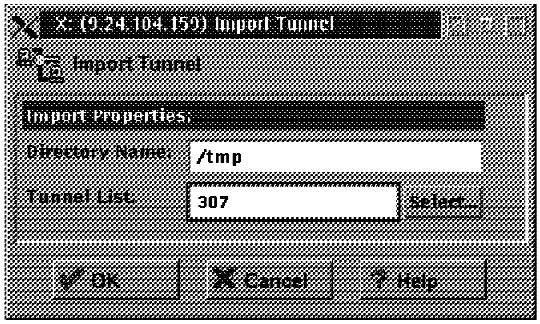


Figure 118. Importing the Tunnel Definition

The import function swaps the source and destination addresses.

7.3.4 Activate/Deactivate the Tunnel at Both Ends

Now you can activate the tunnel at both ends using the GUI. Just specify the tunnel that you would like to activate (see Figure 119).

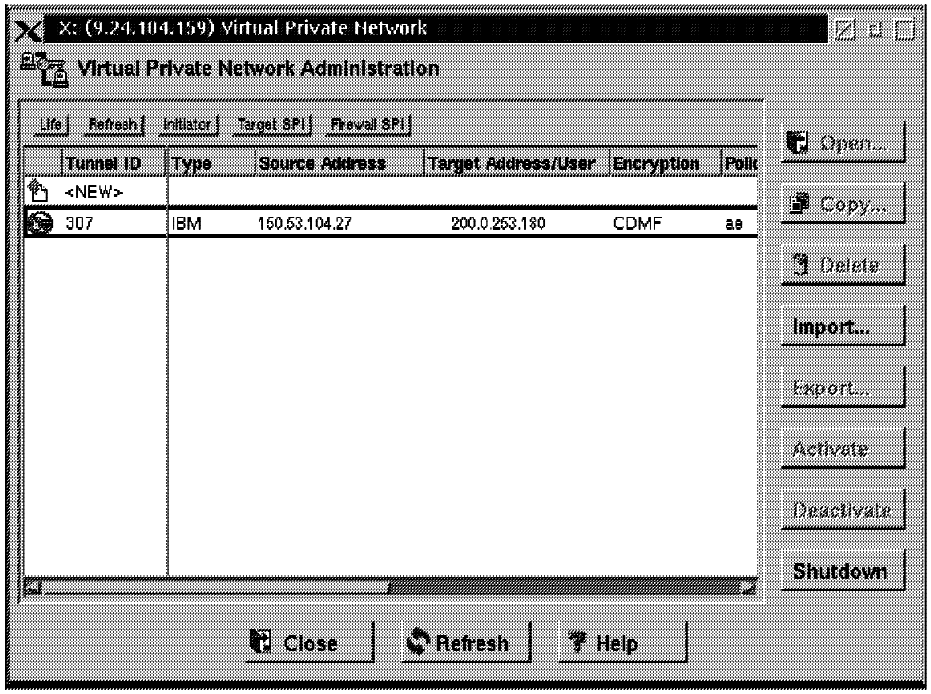


Figure 119. Tunnels List

Figure 119 shows the tunnel list before activating the tunnel, and Figure 120 on page 144 shows the active tunnel. You can see that the icon at the left of the tunnel ID is different after the tunnel is activated.

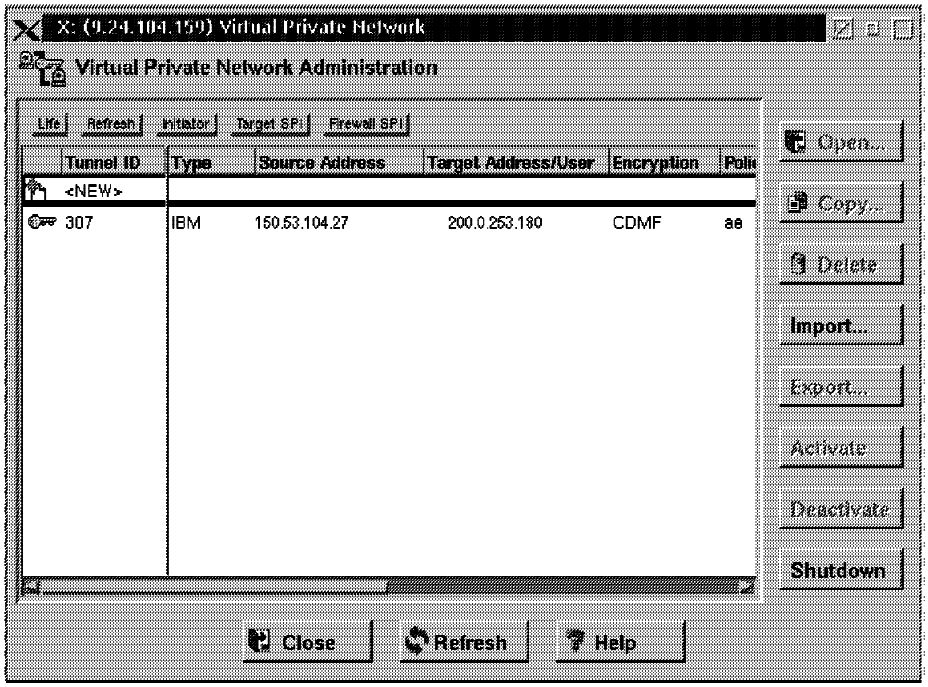


Figure 120. Activating the Secure Tunnel

Activate just enables the code; the tunnel will be marked active even if the other end is not running or connected.

In addition you have a couple of options to deactivate the tunnels. The first one **deactivate**, stops one tunnel activity, but the second one **shutdown** deactivates immediately all the tunnels. This option is used for security reasons.

7.3.5 Specify Which Protocols You Want to Tunnel Using Filtering Rules

In order to specify which nodes and protocols will use each tunnel, you will have to configure special filter rules. These rules will be like normal rules (with source, target, protocol, ports and port operations), but they will also have a tunnel ID. So when a packet must be transfered, the IBM Firewall will search the filtering rules. If it matches a rule, and this rule has a specific tunnel ID, the packet will be sent according to the authentication/encryption rules specified in this specific tunnel.

Figure 121 on page 145 shows the configuration that we used in our tests.

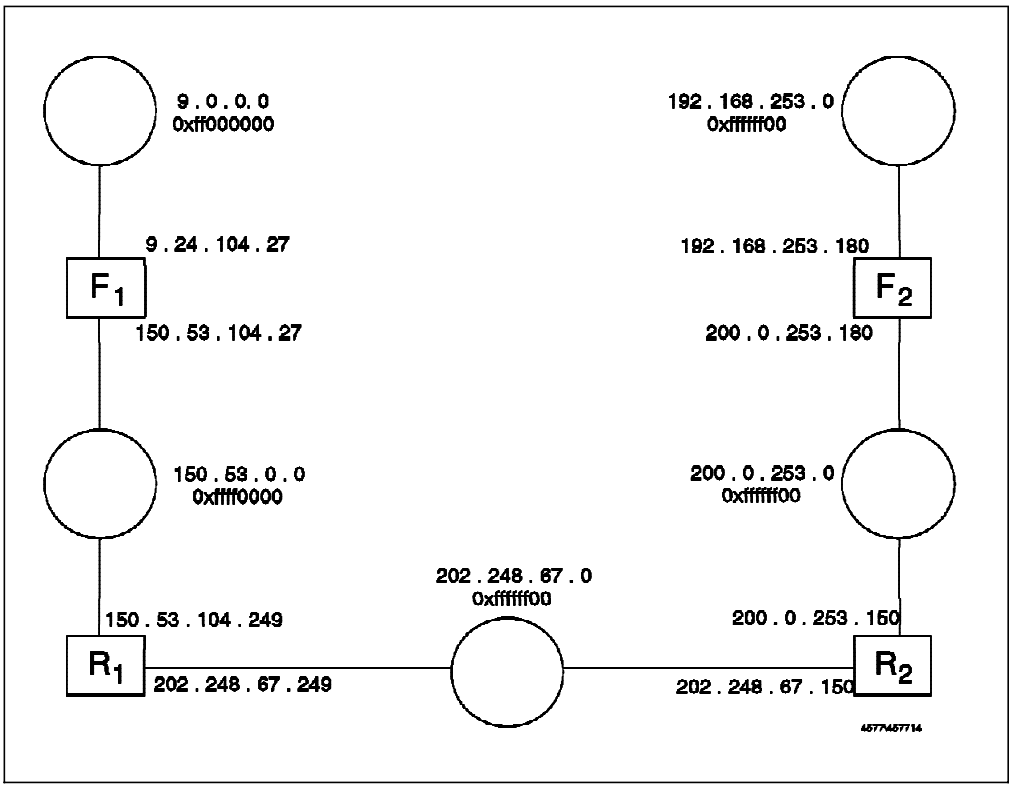


Figure 121. Test Network for Secure Tunnels

The test network has two firewalls, protecting the secure networks 9.0.0.0 and 192.168.253.0. Their nonsecure IP addresses are 150.53.104.27 and 200.0.253.180, respectively.

We defined two tunnels between the firewalls. Through tunnel 307 we will send TCP and UDP packets authenticated and encrypted and through tunnel 308 we will send ICMP packets authenticated but not encrypted. Figure 122 shows the tunnel configuration.

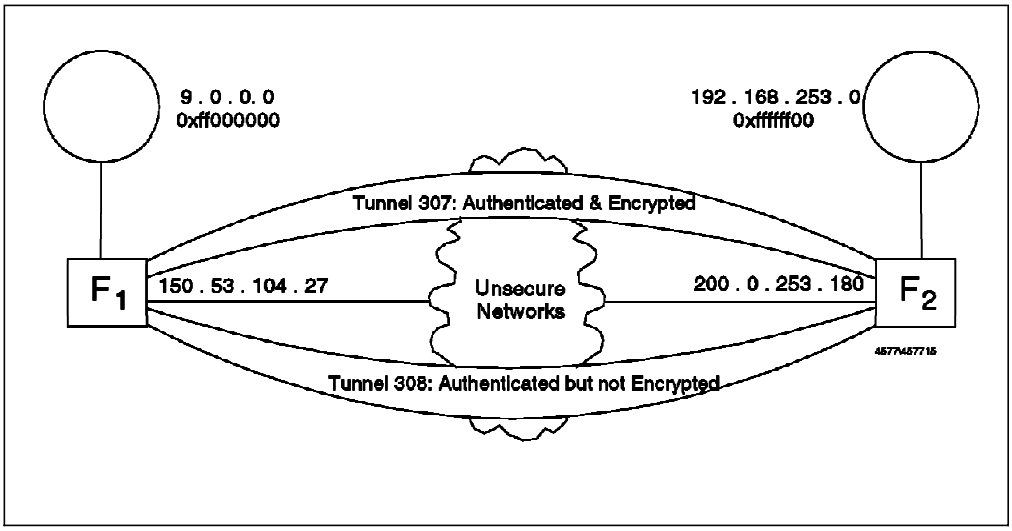


Figure 122. Tunnel Configuration for Testing

Following the procedure we described earlier, we first defined the tunnels in F1, exported the definitions, moved them to F2 and imported them there. Next we

defined filtering rules at both ends in order to specify which packets will go through each tunnel.

In order to be able to understand the rules better, we use the symbolic names (FW1 and FW2) instead of the actual IP addresses (150.53.104.27 and 200.0.253.180) in the rules illustrated below.

The filtering rules have to cater for three types of traffic: UDP packets between ports 4001 for the session key update protocol, ESP for encryption of the real data and AH for authentication of the real data.

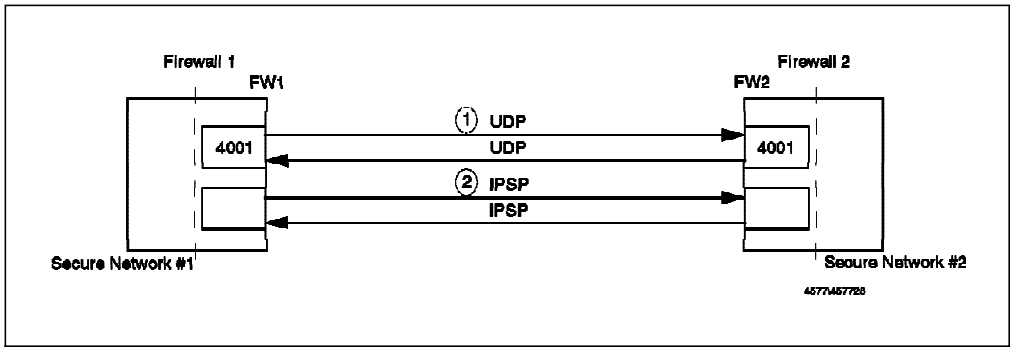


Figure 123. Protocols Used by Secure Tunnel

For each tunnel, packets come in the clear from the secure side, then they are sent through one of the tunnels, received at the second end of the tunnel and go again in the clear to the final destination.

Figure 124 shows the filter rules that we used in *Firewall 1* for the tunnels.

```
# UDP Packets between Tunnel End Points, for the Session Key Update
permit FW1 0xffffffff FW2 0xffffffff udp eq 4001 eq 4001 nonsecure local outbound f=n
permit FW2 0xffffffff FW1 0xffffffff udp eq 4001 eq 4001 nonsecure local inbound  f=n

# ESP Packets between Tunnel End Points
permit FW1 0xffffffff FW2 0xffffffff esp any 0 any 0 nonsecure local outbound f=n
permit FW2 0xffffffff FW1 0xffffffff esp any 0 any 0 nonsecure local inbound  f=n

# AH Packets between Tunnel End Points
permit FW1 0xffffffff FW2 0xffffffff ah any 0 any 0 nonsecure local outbound f=n
permit FW2 0xffffffff FW1 0xffffffff ah any 0 any 0 nonsecure local inbound  f=n

# ICMP Packets, send them Authenticated but Non Encrypted (Tunnel 308)
permit  SN1 SM1 SN2 SM2 icmp any 0 any 0 secure    both inbound  f=n
permit  SN1 SM1 SN2 SM2 icmp any 0 any 0 nonsecure both outbound f=n t=308

# ICMP Packets, receive them Authenticated but Non Encrypted (Tunnel 308)
permit  SN2 SM2 SN1 SM1 icmp any 0 any 0 nonsecure both inbound  f=n t=308
permit  SN2 SM2 SN1 SM1 icmp any 0 any 0 secure    both outbound f=n

# Other Packets (TCP, UDP), send them Authenticated and Encrypted (Tunnel 307)
permit  SN1 SM1 SN2 SM2 all any 0 any 0 secure    both inbound  f=n
permit  SN1 SM1 SN2 SM2 all any 0 any 0 nonsecure both outbound f=n t=307

# Other Packets (TCP, UDP), send them Authenticated and Encrypted (Tunnel 307)
permit  SN2 SM2 SN1 SM1 all any 0 any 0 nonsecure both inbound  f=n t=307
permit  SN2 SM2 SN1 SM1 all any 0 any 0 secure    both outbound f=n
```

Figure 124. Filter Rules Used on Firewall 1 for the Tunnels

The first pair of rules allows the session key update protocol between the firewalls. The second pair of rules allows the ESP protocol between the firewalls for encryption of packets. The third pair of rules allows the AH protocol between

the firewalls to authenticate the packets. The fourth pair of rules allows ICMP packets from Secure Network 1 to go to Secure Network 2 through tunnel 308 (they are received in the clear through the secure interface and sent through the tunnel on the nonsecure interface). The fifth pair allows incoming ICMP packets from Secure Network 2 through tunnel 308. The sixth pair allows other IP packets (TCP, UDP) to be sent from Secure Network 1 to Secure Network 2 using tunnel 307. Finally the seventh pair of rules allows incoming replies from Secure Network 2 through tunnel 307.

The rules for the other end of the tunnel are a mirror image. Figure 125 shows the filter rules that we used in *Firewall 1* for the tunnels.

```
# UDP Packets between Tunnel End Points, for the Session Key Update
permit FW2 0xffffffff FW1 0xffffffff udp eq 4001 eq 4001 nonsecure local outbound f=n
permit FW1 0xffffffff FW2 0xffffffff udp eq 4001 eq 4001 nonsecure local inbound  f=n

# ESP Packets between Tunnel End Points
permit FW2 0xffffffff FW1 0xffffffff esp any 0 any 0 nonsecure local outbound f=n
permit FW1 0xffffffff FW2 0xffffffff esp any 0 any 0 nonsecure local inbound  f=n

# AH Packets between Tunnel End Points
permit FW2 0xffffffff FW1 0xffffffff ah any 0 any 0 nonsecure local outbound f=n
permit FW1 0xffffffff FW2 0xffffffff ah any 0 any 0 nonsecure local inbound  f=n

# ICMP Packets, receive them Authenticated but Non Encrypted (Tunnel 308)
permit  SN1 SM1 SN2 SM2 icmp any 0 any 0 nonsecure both inbound  f=n t=308
permit  SN1 SM1 SN2 SM2 icmp any 0 any 0 secure    both outbound f=n

# ICMP Packets, send them Authenticated but Non Encrypted (Tunnel 308)
permit  SN2 SM2 SN1 SM1 icmp any 0 any 0 secure    both inbound  f=n
permit  SN2 SM2 SN1 SM1 icmp any 0 any 0 nonsecure both outbound f=n t=308

# Other Packets (TCP, UDP), receive them Authenticated and Encrypted (Tunnel 307)
permit  SN1 SM1 SN2 SM2 all any 0 any 0 nonsecure both inbound  f=n t=307
permit  SN1 SM1 SN2 SM2 all any 0 any 0 secure    both outbound f=n

# Other Packets (TCP, UDP), send them Authenticated and Encrypted (Tunnel 307)
permit  SN2 SM2 SN1 SM1 all any 0 any 0 secure    both inbound  f=n
permit  SN2 SM2 SN1 SM1 all any 0 any 0 nonsecure both outbound f=n t=307
```

Figure 125. Filter Rules Used on Firewall 2 for the Tunnels

7.3.6 Refresh Manual Tunnel When Key Expired

This step is required for manual tunnels only. When the tunnel lifetime is reached, the tunnel will cease operation until it is refreshed. Figure 126 on page 148 shows the VPN Administration screen with an expired manual tunnel.

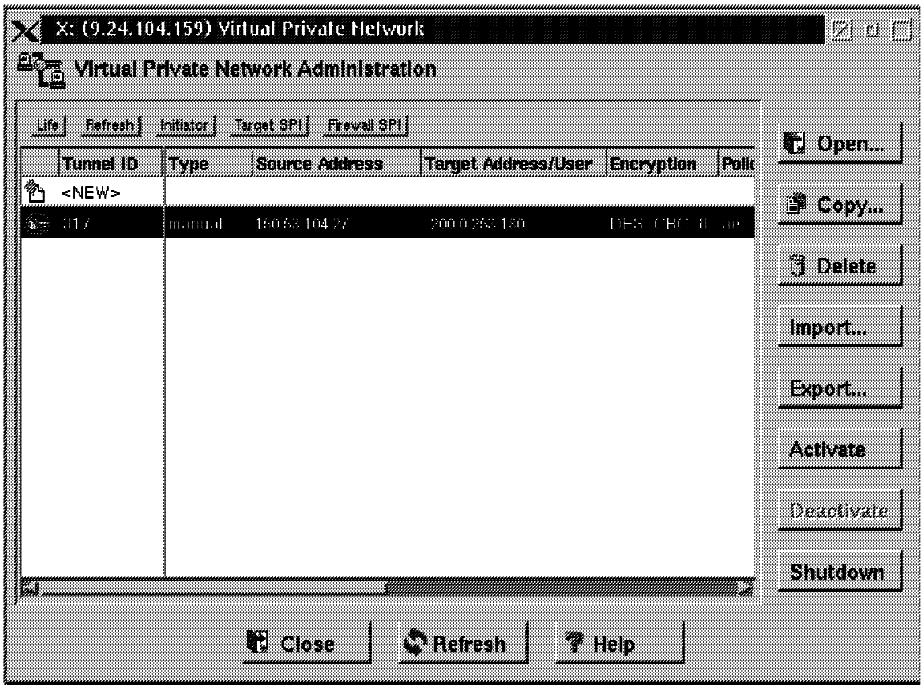


Figure 126. Re-activating the Secure Tunnel

To refresh the tunnel, simply select the tunnel and click the **Activate** button. However, refreshing a tunnel would only re-activate it to an operational state. The keys used in the tunnel remain the same. To re-establish the tunnel with new session keys, you need to delete the tunnel, and then add a new tunnel with the same tunnel ID and characteristics back into the firewall. After that, you export the new tunnel definition to the tunnel partner. New session keys are stored inside the definition files. Your tunnel partner is also required to delete the existing tunnel and then re-import the new definition.

7.3.7 Summary

The following list is a summary of the steps to create and activate a tunnel:

1. Create a firewall object for the non-secure interface of remote firewall.
2. Create a Network object for the secure network of remote firewall (or for the specific hosts to be connected with)
3. Create the tunnel itself, local address=non-secure interface of local firewall and target address=remote firewall object from 1).
4. Export definitions, transport to remote firewall, import definitions (which automatically switches local and remote addresses).
5. Add connection, services for both VPN encapsulation plus VPN key exchange, source=non-secure interface of my firewall, destination=remote firewall object from 1). This allows the firewall-firewall communication of encapsulated data.
6. Copy the VPN 2/2 rule to a new rule called "VPN traffic 2/2 tunnel xx" and set the tunnel ID in that rule.
7. Create a connection, services "VPN traffic 1/2" and "VPN traffic 2/2 tunnel xx" from 6). Source=my secure network (or the set of hosts allowed to use VPN), destination=secure network of remote firewall from 2).

8. Ensure "no -o ipforwarding=1" is set.
9. Repeat 1,2,5,6,7,8 at remote firewall. (Note "remote" and "local" are relative to that firewall.)
10. Activate rulesets
11. Activate tunnel at both ends.
12. Try to ping between the networks. It should work!

7.4 Using the AIX IPSec Client

The AIX IPSec client is supplied with the IBM Firewall 3.1 for AIX. By installing the IPSec client software in an AIX host, the host can securely communicate with IBM Firewall or another AIX IPSec client through a secure tunnel.

The installation procedure is described in *IBM Firewall For AIX User's Guide Version 3.1.1*, GC31-8419-00. However, besides installing the AIX IPSec client, you need to install the latest fixes for AIX on the machine as well. Otherwise, the AIX IPSec client would not function properly. You can find instructions for installing AIX fixes in 4.2.1.2, " Install AIX Fixes" on page 38.

The configuration steps for AIX IPSec client are very similar to the IBM Firewall secure tunnel implementation described in 7.3, "Implementing the Secure IP Tunnel - IBM and Manual Tunnels" on page 140. These steps are:

- Add a tunnel
- Export the tunnel definition
- Transfer the tunnel definition to the tunnel partner
- Import the tunnel definition in the tunnel partner
- Activate the tunnel at both ends
- Refresh the tunnel when session keys expired

7.4.1 Adding Tunnel

The screen for adding a tunnel under SMIT is shown in Figure 127 on page 150.

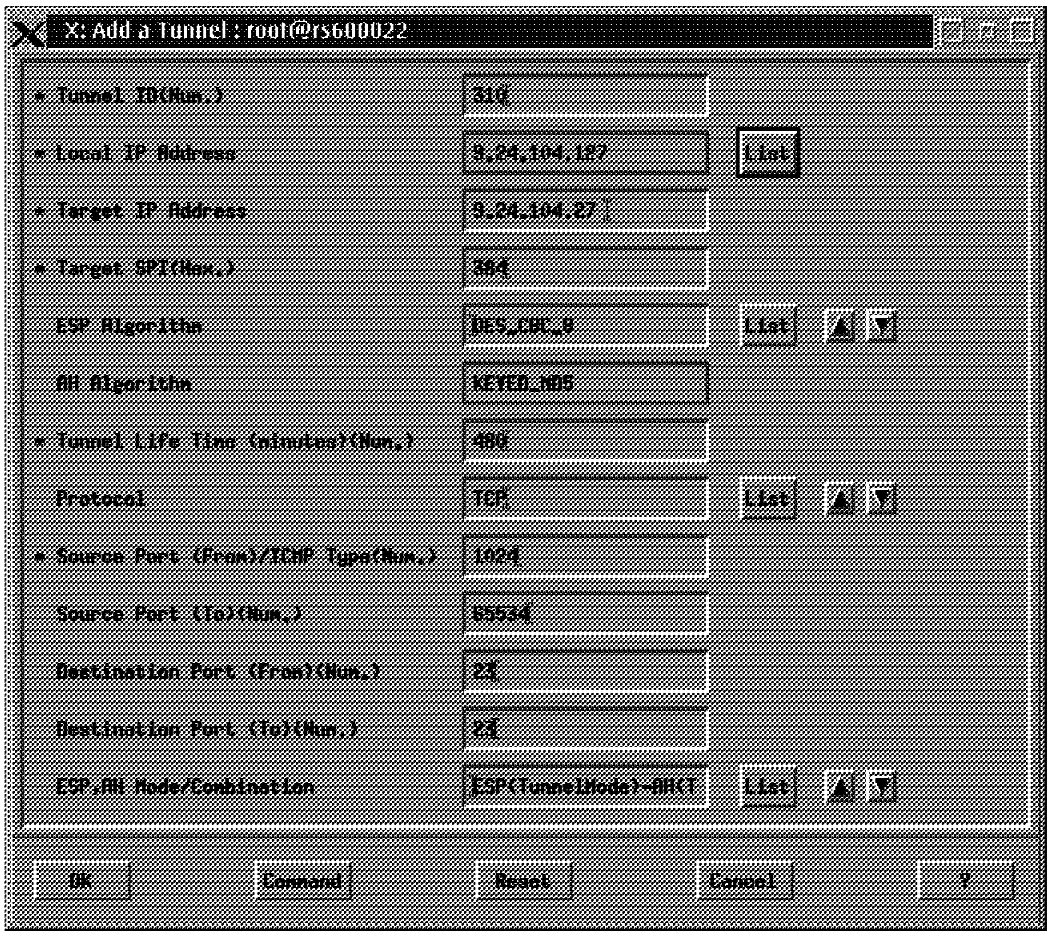


Figure 127. Adding Tunnel in AIX IPSec Client

The importance of Target SPI, ESP Algorithm and Tunnel Life Time have been explained in the IBM Firewall secure tunnel implementation (see 7.3.1, “Adding the Tunnel Definition in One Node” on page 141). However, the following new fields are introduced to AIX IPSec client:

- Protocol
- Source Port From
- Source Port To
- Destination Port From
- Destination Port To
- ESP,AH Mode/Combination

They collectively define the scope within which the secure tunnel mechanism will be applied. You can treat them as a tunnel filtering rule in AIX IPSec client host. For example, the data shown in Figure 127 defines an IPSec tunnel for TCP traffic originating from the node 9.24.104.127 (port greater 1023) to the node 9.24.104.159 (port 23). The traffic for this tunnel is encrypted in tunnel mode first, then authenticated in tunnel mode. Any other traffic will be sent outside this tunnel.

7.4.2 Exporting the Tunnel Definition

Figure 128 shows the tunnel export screen under SMIT. You need to specify the tunnels that you want to export, and the directory where the export files will be placed. Unlike the IBM Firewall secure tunnel implementation, the specified directory needs not be an empty directory. So if there are existing tunnel definition files in that directory, they will be over-written.

The AIX IPSec client will generate five files in that directory. They are:

- fwexpmctx.manual
- fwexppolicy
- fwexppolicy.3.1
- expmctx
- exppolicy

We found that the generated tunnel definition files are world readable. Since the session keys of the tunnel are stored in those files, you should change their permission to be readable by root only.

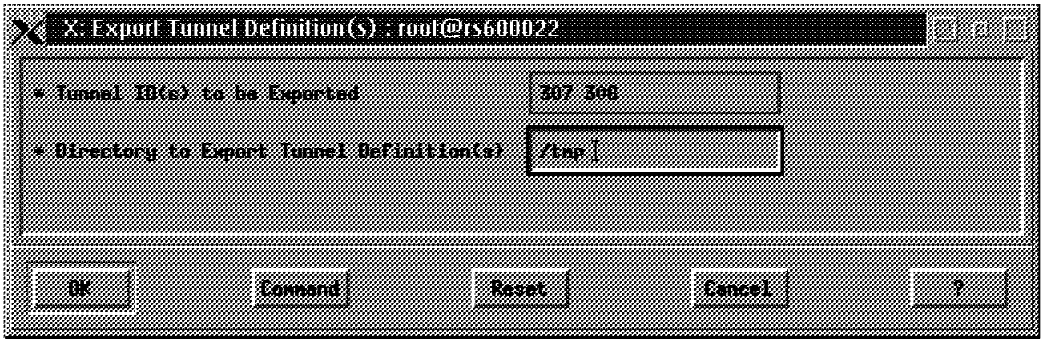


Figure 128. Exporting Tunnel in AIX IPSec Client

7.4.3 Importing the Tunnel Definition

Before you import the tunnel definition into the tunnel partner, they have to be transferred to the other end first. For transferring the tunnel definition files to a tunnel partner, see 7.6, “Using PGP to Distribute the Tunnel Definitions” on page 158.

When you have received the files, place them in a directory and import the definitions (see Figure 129).

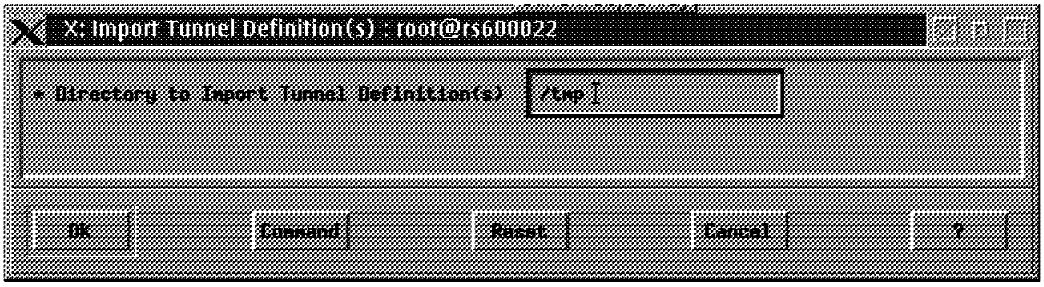


Figure 129. Importing Tunnel in AIX IPSec Client

AIX IPSec client will automatically import *all* tunnels from the definition files placed in the specified directory. Compared with the IBM Firewall, you do not have the option to choose a subset of tunnels to be imported.

7.4.4 Activating the Tunnel

Activating a tunnel is straight forward. You are provided with an option to select which tunnel(s) to activate. Figure 130 shows the Activate/Refresh Tunnel(s) screen from SMIT.

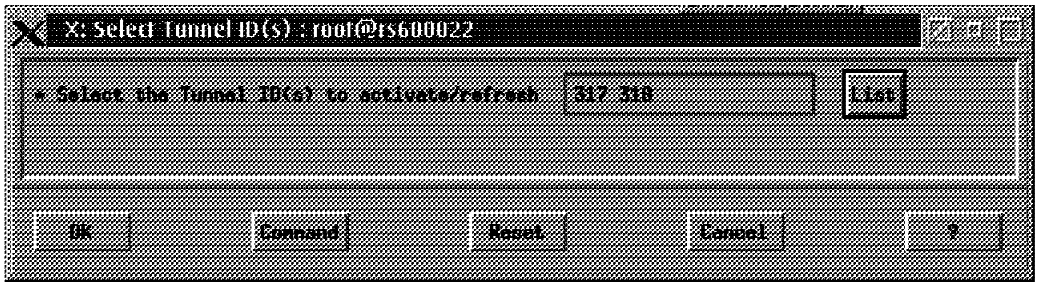


Figure 130. Activating Tunnel in AIX IPSec Client

It should be noted that if your tunnel partner is an IBM Firewall, you have to set up the filtering rules in the firewall to allow the tunnel traffic. The steps to set up filtering rules for a secure tunnel are described in 7.3.5, “Specify Which Protocols You Want to Tunnel Using Filtering Rules” on page 144.

7.4.5 Refreshing the Tunnel When Key Expired

Just like the manual tunnel in an IBM Firewall, the IPSec tunnel for AIX client needs to be refreshed when its keys are expired. Note that the tunnel actviation screen shown in Figure 130 is also used for refreshing the tunnel.

Just like the manual tunnel in an IBM Firewall, the session keys of the IPSec tunnel remain the same after the tunnel is refreshed. To re-establish the tunnel with new session keys, see 7.3.6, “Refresh Manual Tunnel When Key Expired” on page 147.

7.5 Examples

In order to understand how the tunnels work, we will show two examples, one using tunnel 307 (authentication and encryption) and one using tunnel 308 (only authentication). We will start with the tunnel 308 example, as it is easier.

7.5.1 Authentication Example

For this example we will ping address 192.168.253.222 (in secure network 2) from 9.24.104.241 (in secure network 1). We use the -p option of the ping command which allows you to specify the content of the packet, which makes it easier to trace the real packet later and locate the data inside it.

This is the ping command:

```
ping -c 1 -p 6162636465666768696a 192.168.253.222
```

7.5.1.1 No IP Tunnel, Just Plain IP Routing

Figure 131 shows the output of iptrace on the nonsecure interface of F1. Here you will see both packets in the clear (search for the 616263 sequence in the packet content).

```
18:22:13.638477952 9.24.104.241 > 192.168.253.222: icmp: echo request
                        ----------------------------------------
                         4500 0054 ab1d 0000 ff01 dffa 0918 68f1
           HEADERS       c0a8 fdde 0800 2bb9 79ee 0000 3121 1dc4
                         0000 85dd ------------------------------
                        ---------- 6162 6364 6566 6768 696a 6162
                         6364 6566 6768 696a 6162 6364 6566 6768
           PING DATA     696a 6162 6364 6566 6768 696a 6162 6364
                         6566 6768
                        ----------------------------------------

18:22:13.650748288 192.168.253.222 > 9.24.104.241: icmp: echo reply
                        ----------------------------------------
                         4500 0054 c9ce 0000 fb01 c549 c0a8 fdde
           HEADERS       0918 68f1 0000 33b9 79ee 0000 3121 1dc4
                         0000 85dd ------------------------------
                        ---------- 6162 6364 6566 6768 696a 6162
                         6364 6566 6768 696a 6162 6364 6566 6768
           PING DATA     696a 6162 6364 6566 6768 696a 6162 6364
                         6566 6768
                        ----------------------------------------
```

Figure 131. Ping Packet on a Clear Connection (No Tunnel)

As you would expect, when we are using plain routers, the packet has the same sender, destination and data contents as it goes through the second interface (nonsecure network) of the router.

```
18:22:13.639013248 9.24.104.241 > 192.168.253.222: icmp: echo request
                        ----------------------------------------
                         4500 0054 ab1d 0000 fe01 e0fa 0918 68f1
           HEADERS       c0a8 fdde 0800 2bb9 79ee 0000 3121 1dc4
                         0000 85dd ------------------------------
                        ---------- 6162 6364 6566 6768 696a 6162
                         6364 6566 6768 696a 6162 6364 6566 6768
           PING DATA     696a 6162 6364 6566 6768 696a 6162 6364
                         6566 6768
                        ----------------------------------------
18:22:13.650139008 192.168.253.222 > 9.24.104.241: icmp: echo reply
                        ----------------------------------------
                         4500 0054 c9ce 0000 fc01 c449 c0a8 fdde
           HEADERS       0918 68f1 0000 33b9 79ee 0000 3121 1dc4
                         0000 85dd ------------------------------
                         --------- 6162 6364 6566 6768 696a 6162
                         6364 6566 6768 696a 6162 6364 6566 6768
           PING DATA     696a 6162 6364 6566 6768 696a 6162 6364
                         6566 6768
                        ----------------------------------------
```

Figure 132. Ping Packet Is Unchanged

7.5.1.2 Secure IP Tunnel

Now we activate the secure tunnels. Ping is an ICMP request, so in this case the tunnel only uses authentication. We therefore expect to see the packet content in both the secure network and the nonsecure network. When we look at the packet through the secure interface of F1, we will see the packet in the clear, as before.

```
18:40:43.72561536 9.24.104.241 > 192.168.253.222: icmp: echo request
                        ----------------------------------------
                         4500 0054 c985 0000|ff|01|c192|0918 68f1
                                            ----  ------
           HEADERS       c0a8 fdde 0800 8d37 75 80 0000 3121 2219
                         0007 2471 ------------------------------
                        ---------- 6162 6364 65 66 6768 696a 6162
                         6364 6566 6768 696a 61 62 6364 6566 6768
           PING DATA     696a 6162 6364 6566 67 68 696a 6162 6364
                         6566 6768
                        ----------------------------------------
18:40:43.85301888 192.168.253.222 > 9.24.104.241: icmp: echo reply
                        ----------------------------------------
                         4500 0054 c9d8 0000|fd|01|c33f|c0a8 fdde
                                            ----  ------
           HEADERS       0918 68f1 0000 9537 75 80 0000 3121 2219
                         0007 2471 ------------------------------
                        ---------- 6162 6364 65 66 6768 696a 6162
                         6364 6566 6768 696a 61 62 6364 6566 6768
           PING DATA     696a 6162 6364 6566 67 68 696a 6162 6364
                         6566 6768
                        ----------------------------------------
```

Figure 133. Ping Packet in the Originating Network

However, when we look at the nonsecure adapter of F1, we can see that the packet that is sent is different. Figure 134 on page 155 shows the first packet that goes from the nonsecure interface of F1 to the nonsecure interface of F2 and its reply. Now we do not see ICMP messages, we just see IPSP messages (indicated by ip-proto-253, fd in hex instead of 01 for ICMP). Another point that is important is that as we are using this tunnel just for authentication, people on the nonsecure network will be able to look at the packet content (again search for the 616263 pattern in the packet). If you look carefully at the enclosed packet, you will see that it is exactly the same except for the 8-bit TTL that is decremented by one and the 16-bit checksum that is adjusted because of the TTL change.

```
18:40:43.73537280 150.53.104.27 > 200.0.253.180: ip-proto-253 112
                  ----------------------------------------
           NEW HEADERS   4500 0084 5e38 0000 fe fd 993e 9635 681b
                         c800 fdb4 0100 0003 00 01 0070 0000 00f6
                        ----------------------------------------
                                           TTL  Checksum
                        ----------------------------------------
           ORIGINAL      4500 0054 c985 0000|fe|01|c292|0918 68f1
                                            ----  ------
           HEADERS       c0a8 fdde 0800 8d37 75 80 0000 3121 2219
                         0007 2471 ----------------------------
                        ---------- 6162 6364 65 66 6768 696a 6162
                         6364 6566 6768 696a 61 62 6364 6566 6768
           PING DATA     696a 6162 6364 6566 67 68 696a 6162 6364
                         6566 6768 ----------------------------
                        ---------- 2967 c744 20 97 2b34 ad15 1e41
           AUTHENTIC.    552e bef4
                        ----------------------------------------
18:40:43.84263168 200.0.253.180 > 150.53.104.27: ip-proto-253 112
                  ----------------------------------------
           NEW HEADERS   4500 0084 c8b6 0000 fc fd 30c0 c800 fdb4
                         9635 681b 0100 0003 00 01 0070 0000 01ce
                        ----------------------------------------
                                           TTL  Checksum
                        ----------------------------------------
           ORIGINAL      4500 0054 c9d8 0000|fe|01|c23f|c0a8 fdde
                                            ----  ------
           HEADERS       0918 68f1 0000 9537 75 80 0000 3121 2219
                         0007 2471 ----------------------------
                        ---------- 6162 6364 65 66 6768 696a 6162
                         6364 6566 6768 696a 61 62 6364 6566 6768
           PING DATA     696a 6162 6364 6566 67 68 696a 6162 6364
                         6566 6768 ----------------------------
                        ---------- 81b4 1e54 92 fa d857 916b ee0e
           AUTHENTIC.    a4c2 de6f
                        ----------------------------------------
```

Figure 134. Ping Packet in Authenticated Tunnel

7.5.2 Encryption Example

For the encryption example we will show a TCP based service. As we again want to keep the example simple, we will use the *echo* service. This service just echoes the input that it receives on a TCP port (it uses port 7). As this is a TCP based service, it will be (according to our filter rules) routed through tunnel 307 using encryption and authentication. We will show traces with plain routers first and then with the IP tunnel in place. In both cases we will send the packet from secure network 1.

Figure 135 shows the echo command and response.

```
> telnet 192.168.253.180 echo
Trying...
Connected to 192.168.253.180.
Escape character is '¬]'
abcdefghijabcdefghij
abcdefghijabcdefghij
```

Figure 135. Using the TCP/IP Echo Service

7.5.2.1 Plain Routing without a Secure Tunnel

In this case we can see on all the interfaces the user input in the clear (packet #4) and the echoed string in the clear (packet #5). The trace is more complex than that of the previous example because of the TCP handshake and acknowledgments. Figure 136 and Figure 137 on page 157 show the trace in the originating secure network and in the nonsecure network, respectively.

```
11:46:27.906422912 9.24.104.241.1169 > 192.168.253.180.7: S 1681600001:1681600001(0)
                                       win 16384 <mss 512>
                         4500 002c 7b63 0000 3c06 d302 0918 68f1
Packet #1                c0a8 fdb4 0491 0007 643b 2e01 0000 0000
                         6002 4000 949f 0000 0204 0200
11:46:27.910948096 192.168.253.180.7 > 9.24.104.241.1169: S 1344959489:1344959489(0)
                                     ack 1681600002 win 16384 <mss 512>

                         4500 002c b6d3 0000 3906 9a92 c0a8 fdb4
Packet #2                0918 68f1 0007 0491 502a 7401 643b 2e02
                         6012 4000 d062 0000 0204 0200
11:46:27.914418688 9.24.104.241.1169 > 192.168.253.180.7: . ack 1 win 16384
                         4500 0028 7b64 0000 3c06 d305 0918 68f1
Packet #3                c0a8 fdb4 0491 0007 0000 0001 0000 0001
                         5010 4000 e46b 0000
11:46:45.759564800 9.24.104.241.1169 > 192.168.253.180.7: P 1:33(32) ack 1 win 16384
                         4500 0048 7bbc 0000 3c06 d28d 0918 68f1
Packet #4                c0a8 fdb4 0491 0007 0000 0001 0000 0001
abcd entered  ---->      5018 4000 e639 0000 6162 6364 6566 6768
                         696a 6162 6364 6566 6768 696a 6162 6364
                         6566 6768 696a 0d0a
11:46:45.764526592 192.168.253.180.7 > 9.24.104.241.1169: P 1:33(32) ack 33 win 16384
                         4500 0048 b6d4 0000 3906 9a75 c0a8 fdb4
Packet #5                0918 68f1 0007 0491 0000 0001 0000 0021
abcd echoed   ---->      5018 4000 e619 0000 6162 6364 6566 6768
                         696a 6162 6364 6566 6768 696a 6162 6364
                         6566 6768 696a 0d0a
11:46:45.814578432 9.24.104.241.1169 > 192.168.253.180.7: . ack 33 win 16384
                         4500 0028 7bbd 0000 3c06 d2ac 0918 68f1
Packet #6                c0a8 fdb4 0491 0007 0000 0021 0000 0021
                         5010 4000 e42b 0000
```

Figure 136. No Encryption, Secure Network Trace

```
11:46:27.906934272 9.24.104.241.1169 > 192.168.253.180.7: S 1681600001:1681600001(0)
                                     win 16384 <mss 512>
                        4500 002c 7b63 0000 3b06 d402 0918 68f1
Packet #1               c0a8 fdb4 0491 0007 643b 2e01 0000 0000
                        6002 4000 949f 0000 0204 0200
11:46:27.910315520 192.168.253.180.7 > 9.24.104.241.1169: S 1344959489:1344959489(0)
                                    ack 1681600002 win 16384 <mss 512>
                        4500 002c b6d3 0000 3a06 9992 c0a8 fdb4
Packet #2               0918 68f1 0007 0491 502a 7401 643b 2e02
                        6012 4000 d062 0000 0204 0200 6976
11:46:27.914952192 9.24.104.241.1169 > 192.168.253.180.7: . ack 1 win 16384
                        4500 0028 7b64 0000 3b06 d405 0918 68f1
Packet #3               c0a8 fdb4 0491 0007 0000 0001 0000 0001
                        5010 4000 e46b 0000
11:46:45.760137216 9.24.104.241.1169 > 192.168.253.180.7: P 1:33(32) ack 1 win 16384
                        4500 0048 7bbc 0000 3b06 d38d 0918 68f1
Packet #4               c0a8 fdb4 0491 0007 0000 0001 0000 0001
abcd entered  ---->     5018 4000 e639 0000 6162 6364 6566 6768
                        696a 6162 6364 6566 6768 696a 6162 6364
                        6566 6768 696a 0d0a
11:46:45.763902208 192.168.253.180.7 > 9.24.104.241.1169: P 1:33(32) ack 33 win 16384
                        4500 0048 b6d4 0000 3a06 9975 c0a8 fdb4
Packet #5               0918 68f1 0007 0491 0000 0001 0000 0021
abcd echoed   ---->     5018 4000 e619 0000 6162 6364 6566 6768
                        696a 6162 6364 6566 6768 696a 6162 6364
                        6566 6768 696a 0d0a
11:46:45.815131008 9.24.104.241.1169 > 192.168.253.180.7: . ack 33 win 16384
                        4500 0028 7bbd 0000 3b06 d3ac 0918 68f1
Packet #6               c0a8 fdb4 0491 0007 0000 0021 0000 0021
                        5010 4000 e42b 0000
```

Figure 137. No Encryption, Nonsecure Network Trace

7.5.2.2 Secure IP Tunnel with Encryption

Next we activated the secure tunnel again. As Figure 138 shows, we can see the input and the echo clear in secure network 1.

```
13:44:24.923953792 9.24.104.241.1215 > 192.168.253.180.7: S 2592384001:2592384001(0)
                                     win 16384 <mss 512>
                        4500 002c 0707 0000 3c06 475f 0918 68f1
Packet #1               c0a8 fdb4 04bf 0007 9a84 a401 0000 0000
                        6002 4000 e827 0000 0204 0200
13:44:24.931274752 192.168.253.180.7 > 9.24.104.241.1215: S 2251135489:2251135489(0)
                                     ack 2592384002 win 16384 <mss 512>
                        4500 002c b75f 0000 3b06 9806 c0a8 fdb4
Packet #2               0918 68f1 0007 04bf 862d 9a01 9a84 a402
                        6012 4000 c7e7 0000 0204 0200
13:44:24.934672896 9.24.104.241.1215 > 192.168.253.180.7: . ack 1 win 16384
                        4500 0028 0708 0000 3c06 4762 0918 68f1
Packet #3               c0a8 fdb4 04bf 0007 0000 0001 0000 0001
                        5010 4000 dbf0 0000
13:44:33.391491456 9.24.104.241.1215 > 192.168.253.180.7: P 1:23(22) ack 1 win 16384
                        4500 003e 0736 0000 3c06 471e 0918 68f1
Packet #4               c0a8 fdb4 04bf 0007 0000 0001 0000 0001
abcd entered  ---->     5018 4000 d8c8 0000 6162 6364 6566 6768
                        696a 6162 6364 6566 6768 696a 0d0a
13:44:33.399595008 192.168.253.180.7 > 9.24.104.241.1215: P 1:23(22) ack 23 win 16384
                        4500 003e b760 0000 3b06 97f3 c0a8 fdb4
Packet #5               0918 68f1 0007 04bf 0000 0001 0000 0017
abcd echoed   ---->     5018 4000 d8b2 0000 6162 6364 6566 6768
                        696a 6162 6364 6566 6768 696a 0d0a
13:44:33.552369536 9.24.104.241.1215 > 192.168.253.180.7: . ack 23 win 16384
                        4500 0028 0738 0000 3c06 4732 0918 68f1
Packet #6               c0a8 fdb4 04bf 0007 0000 0017 0000 0017
                        5010 4000 dbc4 0000
```

Figure 138. Secure Network Trace with Encrypted Tunnel

However, as expected, in the trace of the nonsecure adapter in Figure 139 on page 158 we cannot see the user data (look at packets #4 and #5, and you will not be able to find the sequence 6162636465). Look again at the packet headers to see that the IPSP protocol is in use, instead of TCP (ip-proto-253). The real TCP packets form the encrypted payload of the IPSP packets.

```
13:44:24.925197696 150.53.104.27 > 200.0.253.180: ip-proto-253 80
                         4500 0064 ed6e 0000 3bfd cd28 9635 681b
                         c800 fdb4 0100 0005 0000 0050 0000 01c8
Packet #1                0000 0000 0000 001d 3ef1 e9aa f9bf b71b
                         0e35 d281 b1a8 b038 05d6 e45e 337b 9181
                         8f6b a5f5 9335 183f c052 c9cb a25d e438
                         3421 a0b0 dbdc d868 dbb4 f3f7 b24c 979b
                         3425 2ff5
13:44:24.929874816 200.0.253.180 > 150.53.104.27: ip-proto-253 80
                         4500 0064 c93b 0000 3afd f25b c800 fdb4
                         9635 681b 0100 0005 0000 0050 0000 0079
Packet #2                0000 0000 0000 208f 883d 3fcf dd4a cceb
                         8c7c c7a8 eaf6 32ad 2784 c739 2a92 dd88
                         96d8 0a2b ada5 f7e4 ce27 1498 a26e 9983
                         ce00 0193 91f6 9215 4377 29bd 4ed9 2689
                         cebf 58ce
13:44:24.935858048 150.53.104.27 > 200.0.253.180: ip-proto-253 76
                         4500 0060 ed6f 0000 3bfd cd2b 9635 681b
                         c800 fdb4 0100 0005 0000 004c 0000 01c8
Packet #3                0000 0000 0000 001e 3189 5ea1 c522 1537
                         41b1 7f34 323a 9f67 ab59 efc1 8f95 95cf
                         d6d8 7e30 149f 35cb 1a65 e8b5 144c 6da3
                         c294 9bdd f744 5dce 14c2 bb7d ed32 ff23
13:44:33.392776192 150.53.104.27 > 200.0.253.180: ip-proto-253 98
                         4500 0076 ed70 0000 3bfd cd14 9635 681b
                         c800 fdb4 0100 0005 0000 0062 0000 01c8
Packet #4                0000 0000 0000 001f 513d 0e6b 3f01 0478
abcd entered is          bb1b 80b7 af21 85ea d29e 6737 dbcc 2d3b
encrypted and            2bd2 3596 6b94 064b eb66 759b fa7b 285e
authenticated            462f da1b c3f5 618e fb8b 60da 9e83 7b3f
                         428c aadb dc79 ea64 c43e 16a1 df4c bf58
                         0b4a 7002 d165
13:44:33.398115584 200.0.253.180 > 150.53.104.27: ip-proto-253 98
                         4500 0076 c93c 0000 3afd f248 c800 fdb4
                         9635 681b 0100 0005 0000 0062 0000 0079
Packet #5                0000 0000 0000 2090 570b 999b 2b68 b974
abcd echoed is           18c0 8b0d f108 10ad 142f 6167 1836 ce42
encrypted and            0d66 ca70 c272 5e84 ff3b 5b0d ef3a 53ee
authenticated            f1bb 2b51 ec87 fcae 9697 f9ac b993 3097
                         9a55 19a2 d3f4 a38f 5983 5ac2 1642 b79c
                         5a73 5e81 997e
13:44:33.553585920 150.53.104.27 > 200.0.253.180: ip-proto-253 76
                         4500 0060 ed71 0000 3bfd cd29 9635 681b
                         c800 fdb4 0100 0005 0000 004c 0000 01c8
Packet #6                0000 0000 0000 0020 6445 7ebf afb7 8d89
                         0020 5f1b 4f2a 077b 6386 e1f4 3faa 5bf8
                         4793 f204 a012 5657 5d92 b39c ef74 13b1
                         f0bc ef02 8f1c a5a8 0039 bdc8 06b1 195e
```

Figure 139. Nonsecure Network Trace with Encrypted Tunnel

7.6 Using PGP to Distribute the Tunnel Definitions

One weakness of any symmetric-key encryption mechanism is the need to copy the key from one side to the other. In the case of the IBM Firewall secure tunnel, this is performed by exporting the tunnel definition, including the key, and importing it at the other end.

In this section we will discuss one technique to make this key exchange secure, using Pretty Good Privacy (PGP) to encrypt the exported tunnel definition. PGP

is a freely available program which uses public-key cryptography to create encrypted and authenticated messages.

First, you have to decide which version of PGP you must use, the USA version or the International version. You can get both versions via anonymous FTP from concert.cert.dfn.de in the directory /pub/tools/crypt/pgp/unix. Having received the package, you should then compile the PGP program using the command `make rs6000`. Note that this is a good reason for *not* installing PGP on the firewall machine, because you do not want to have a compiler there if you can avoid it. If all goes well, you will end up with an executable file called pgp (in our case it compiled without any modification).

Create a directory /.pgp for storing the keys. Before you start to create your key pair, run these tests (to test that PGP is correctly installed):

- Create a 512-bit public/secret key pair (enter `test` as userid/password).

  ```
  pgp -kg
  ```

- Add the keys from the file keys.asc to the public keyring. PGP will ask if you want to sign the keys you are adding. Answer yes for at least one key.

  ```
  pgp -ka keys.asc
  ```

- Do a keyring check

  ```
  pgp -kc
  ```

- Encrypt pgpdoc1.txt

  ```
  pgp -e pgpdoc1.txt test -o testfile.pgp
  ```

- Decrypt this file

  ```
  pgp testfile.pgp
  ```

This will produce a file called testfile, which should be identical to pgpdoc1.txt. If everything went well, install the pgp program in a bin directory.

Create the directory /usr/local/lib/pgp. Place the documentation, pgpdoc1.txt and pgpdoc2.txt, config.txt, language.txt and the help files (*.hlp) in /usr/local/lib/pgp (this step *is* necessary, because without the documentation it will not run). Then create a subdirectory somewhere in your home directory hierarchy to hold your public and private keyrings and anything else pgp might need (such as the language.txt file). The default name PGP assumes is ~/.pgp. If you want to use a different name, you must set the PGPPATH environment variable to point to it before you use the system. This directory cannot be shared as it will contain your personal private keys.

Now you are ready to use PGP in earnest. First create your public/secret key pair:

```
pgp -kg
```

The private key will stay locked in your keyring file, but you can send the public key to anyone. Anything encrypted using the private key can only be decrypted with the public key, and vice versa. Extract your public key:

```
pgp -kxa user1 user1.asc
```

Send the public key to a node at the other end of the secure tunnel.

Now you need to move to the other end of the secure tunnel. First install PGP, as described earlier, and then receive the public key and load it into your keyring:

```
pgp -ka user2.asc
```

Now move to the local firewall and create a tar file containing the exported tunnel definitions:

```
tar cvf tunnel.tar fwexpmctx fwexppolicy
```

Encrypt the tar file using the public key from the other end of the secure tunnel, which you received earlier. You will have to save it in a suitable way for sending it through SMTP mail (in radix-64 format):

```
pgp -esa tunnel.tar user2 -u user1
```

You can now send the encrypted file in a mail message:

```
mail user2@firewall2 <tunnel.tar.asc
```

Finally, the user at the other end of the tunnel receives the mail and saves it. Then they decrypt it using their private key and their pass phrase:

```
pgp tunnel.tar.asc
```

7.7 Windows 95 IPSec Client

The Windows 95 IPSec Client software will allow you to create a secure tunnel from a PC connected to the nonsecure network and IBM Firewall 3.1.

Windows 95 IPSec Client differs from a standard IPSec tunnel in the following ways:

- It cannot be used on the secure network to tunnel to the firewall.
- Access is given via dynamic filter rules. These override the static rules (due to the unpredictable IP address of the client). Rule permit's are logged automatically; this cannot be changed. Debug logging is sent to Local2.
- Packets decrypted by the firewall are not routed using the firewall's routing entries.
- The Windows 95 IPsec client will NOT run on Windows NT.

7.7.1 Installing and Configuring the Windows 95 IPSec Client

For complete instructions on installing the Windows 95 IPSec Client see "Chapter 14, Using the Windows 95 Secure Remote Client" in *IBM Firewall 3.1 User's Guide*, GC31-8419-00.

A prerequisite for installing the Windows 95 IPSec Client is the Microsoft ISDN Accelerator Pack 1.1 (this does not mean that you need an ISDN connection). The link for obtaining this pack has changed from the one specified in the User' Guide. To obtain this software go to:

```
http://www.microsoft.com/windows/windows95/info/isdn4w95.htm
```

Note: See 7.7.2, “Troubleshooting” on page 168 for information on Microsoft Dial-Up Networking 1.2.

7.7.1.1 IBM Firewall 3.1 Configuration

Several changes are required on the firewall before the Windows 95 IPSec Client will work properly. Here are the steps we went through to configure our firewall:

1. First we create a proxy user, `miner` (see Figure 140) and change the Authentication for `Nonsecure IP` to password.

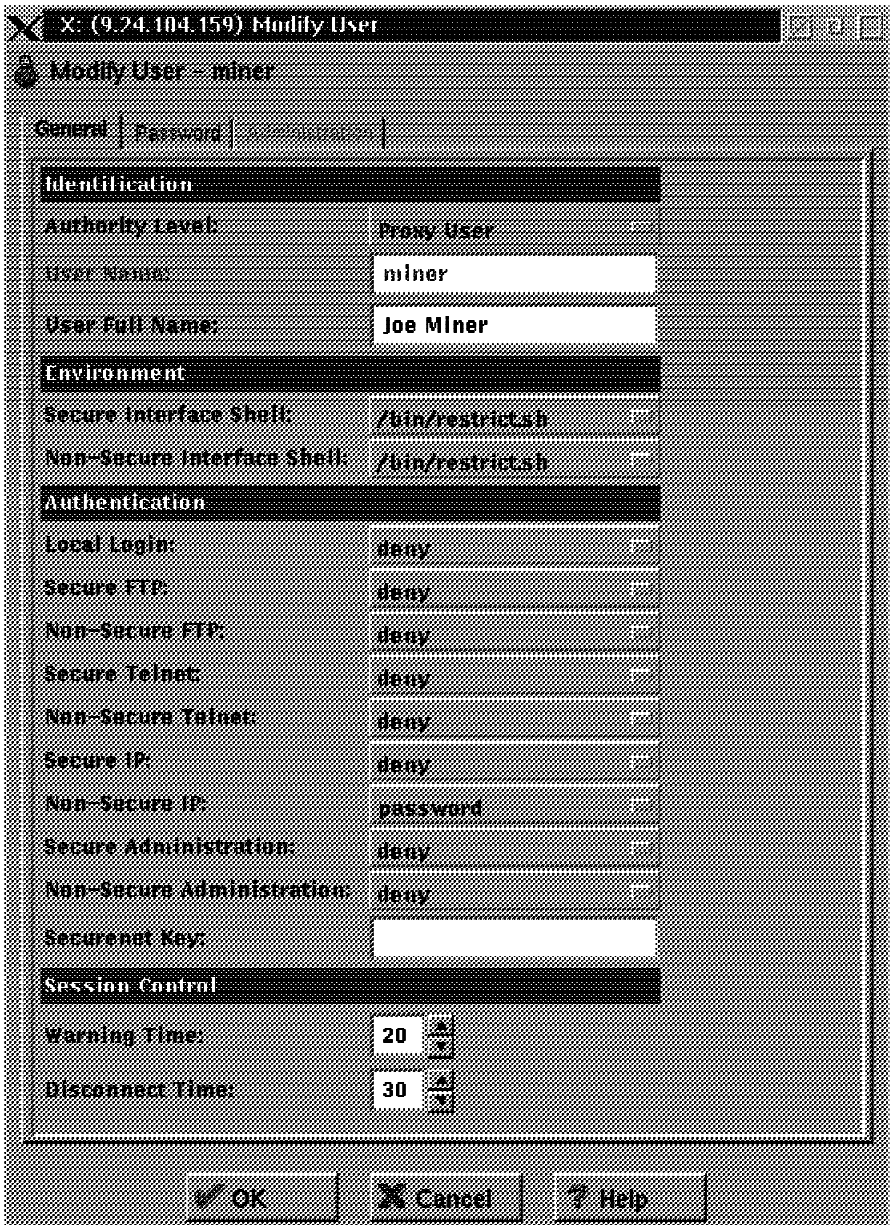


Figure 140. New Proxy User for Tunnelling

2. Next we create a network object referring to `miner`, see Figure 141 on page 162, to associate this user with the tunnel. We specified a short filter lifetime here for testing.

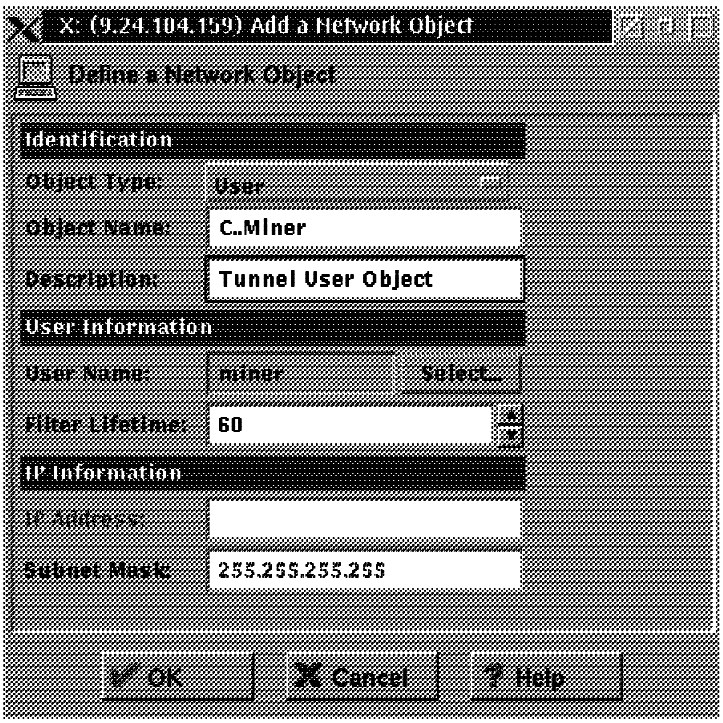


Figure 141. New Tunnel User Network Object

3. then we went into Virtual Private Network and created a new tunnel; see Figure 142 on page 163. We generated a Tunnel ID and Target SPI randomly.

   For a large number of tunnels define a numbering standard to keep track of your tunnels. We also specified the Nonsecure adapter address and the miner network object.

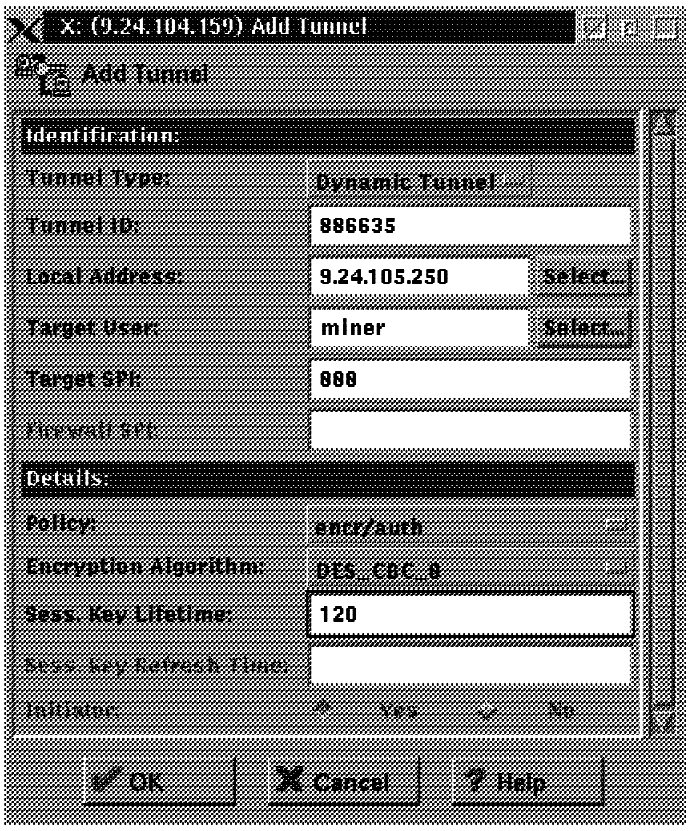


Figure 142. Creating a New Dynamic Tunnel

4. We then added a new connection to allow SSL from The World to the nonsecure adapter; see Figure 143 on page 164.

Figure 143. Creating the SSL Connection

5. Now we need a keyfile to enable SSL negotiation. We had already done this in 4.3.3.5, “Generate an SSL Key” on page 43. Windows 95 IPSec Client uses SSL to exchange keys before entering IPsec itself.
6. We checked to make sure that the process `/usr/sbin/sslrctd` was running. Without this process the Windows 95 IPSec Client cannot use SSL to negotiate a tunnel. It will automatically start on reboot only if you have generated a set of SSL keys.

7.7.1.2 Windows 95 IPSec Client Installation

We installed the Windows 95 IPSec Client as per the instructions in the IBM Firewall User's Guide. This was quite straight forward as each of the packages are self-installing executables. Configuring them after installation was the tricky part. We installed Windows 95 from scratch on a PC and then installed:

1. Microsoft ISDN Accelerator Pack 1.1
2. Windows 95 IPSec Client
3. IbmIsdn Software Network Adapter

If you are installing Windows 95 IPSec Client from a zip file, uncompress the file in a temporary directory. This will also create the driver sub-directory where the IbmIsdn driver is located.

As we were using the IbmIsdn driver for PPP dialup via a modem, we left the options to configure the ISDN connection blank.

7.7.1.3 Configure Windows 95 IPSec Client

We will show how to configure Windows 95 IPSec Client using the screen captures we used during our installation.

Figure 144. Configure Windows 95 IPSec Client (Part 1 of 2)

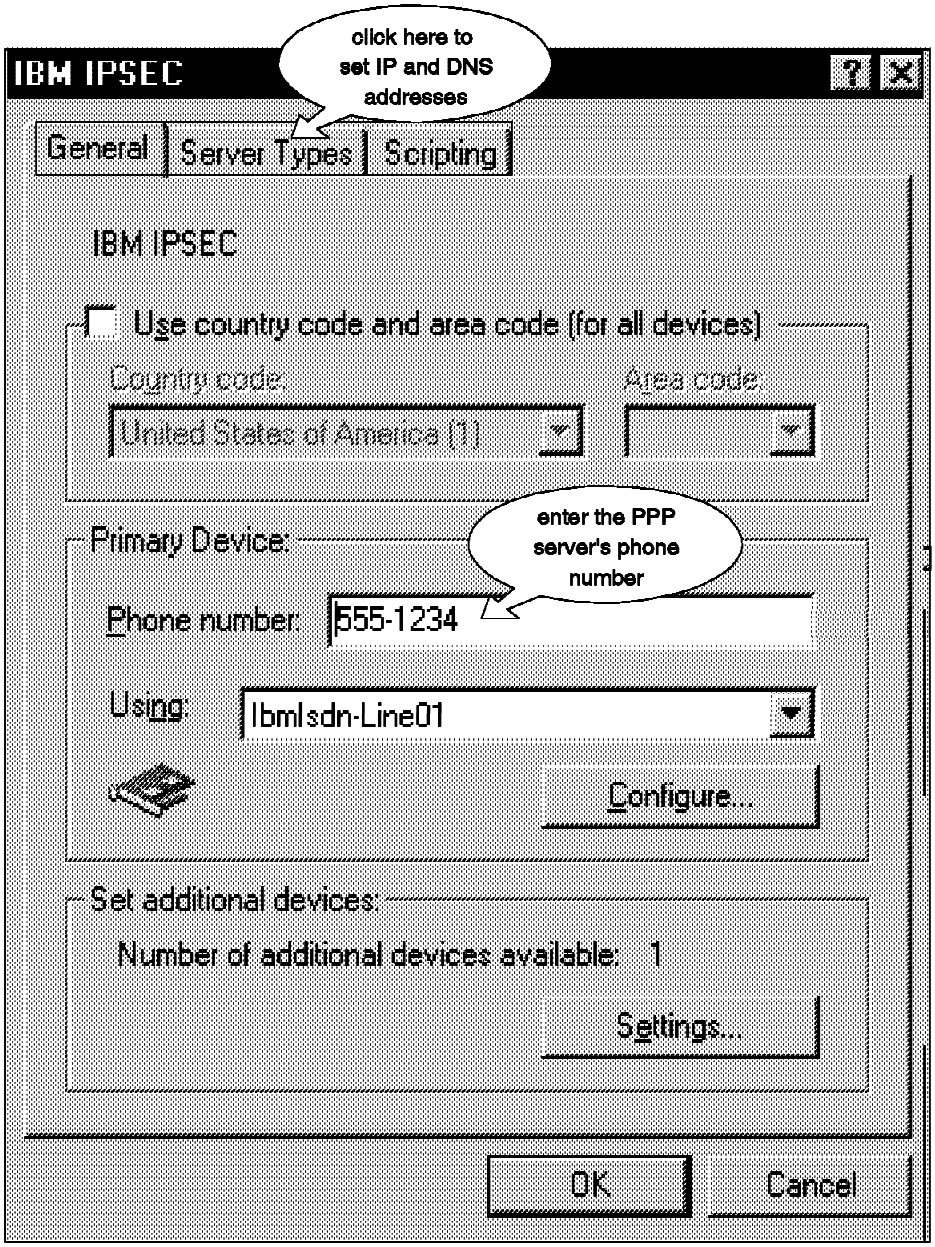


Figure 145. Configure Windows 95 IPSec Client (Part 2 of 2)

7.7.1.4 Using Windows 95 IPSec Client

We will show how to start Windows 95 IPSec Client using the screen captures we used in our installation.

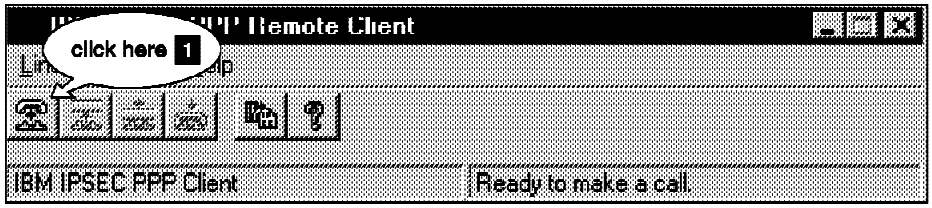


Figure 146. Starting Windows 95 IPSec Client (Part 1 of 8)

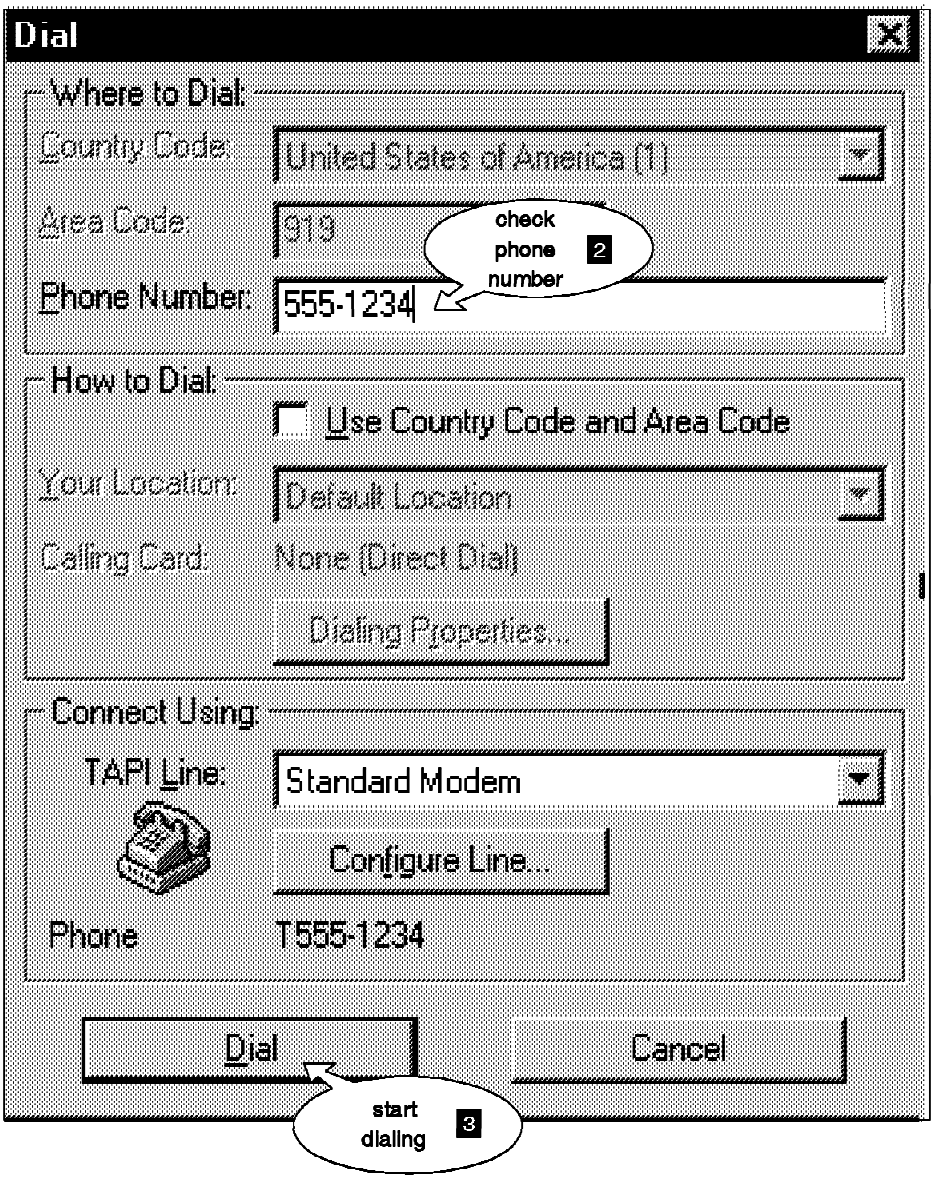


Figure 147. Starting Windows 95 IPSec Client (Part 2 of 8)

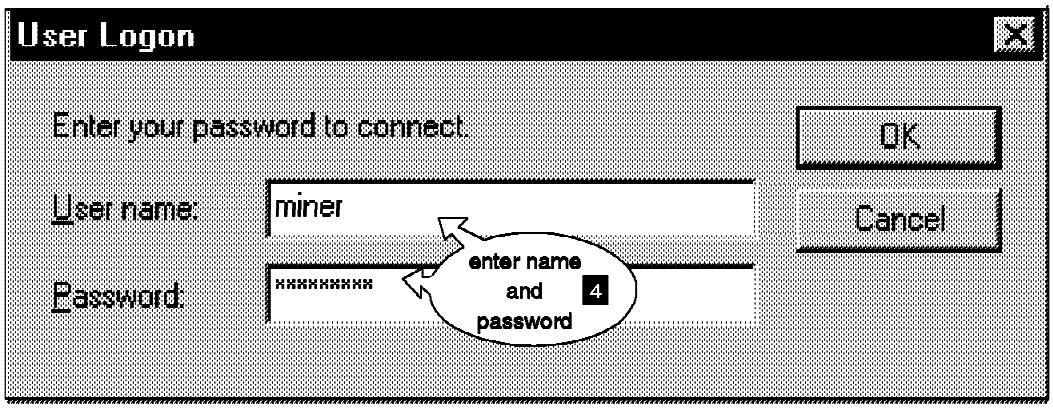


Figure 148. Starting Windows 95 IPSec Client (Part 3 of 8)

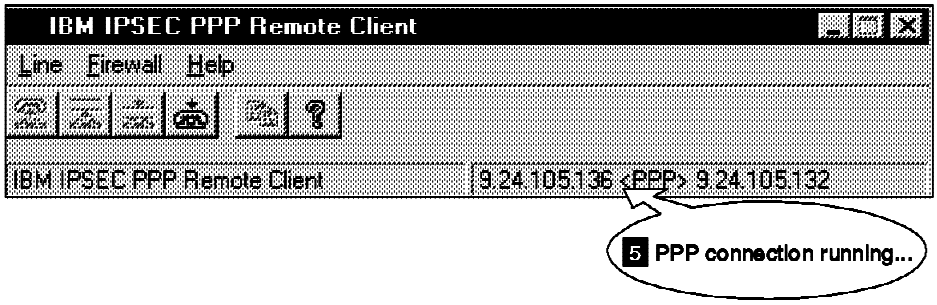


Figure 149. Starting Windows 95 IPSec Client (Part 4 of 8)

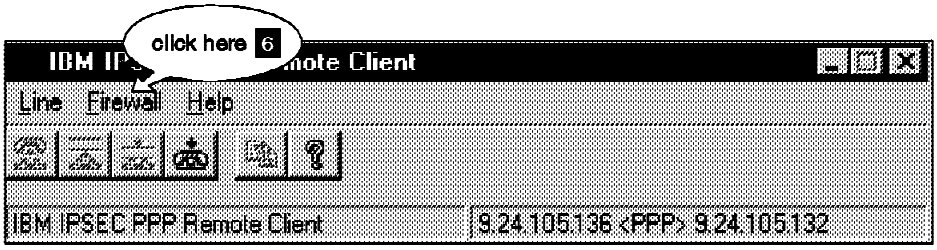


Figure 150. Starting Windows 95 IPSec Client (Part 5 of 8)

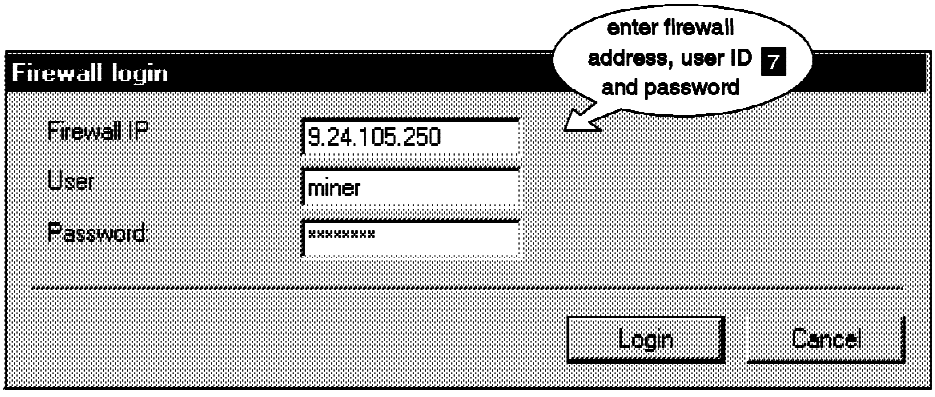


Figure 151. Starting Windows 95 IPSec Client (Part 6 of 8)

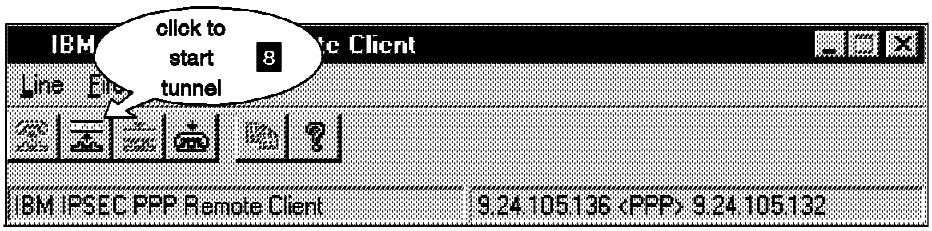


Figure 152. Starting Windows 95 IPSec Client (Part 7 of 8)

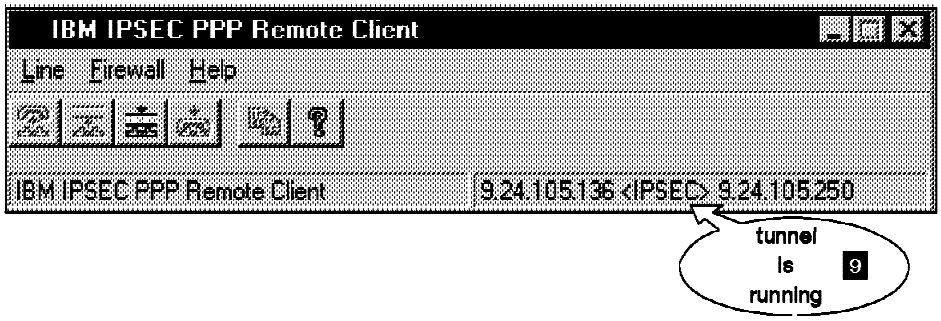


Figure 153. Starting Windows 95 IPSec Client (Part 8 of 8)

7.7.2 Troubleshooting

This section describes how we resolved the few problems we had installing Windows 95 IPSec Client.

7.7.2.1 Cannot Create Dial-Up Networking Entry

Installing Windows 95 IPSec Client after a failed attempt can sometimes result in a IBM IbmIsdn Software Adapter that is named slightly differently from what pppsec.exe expects. A successful install of pppsec.exe will create a Dial-Up networking connection for you. As it doesn't recognize the slight name difference it will not work and give you an error.

To work around this you can manually create your own IBM IPSec Dial-Up connection using Windows 95 Dial-Up Networking Accessory. When selecting a device you should see:

```
IbmIsdn-Line01 Software Adapter
```

The number after `Line` might be slightly different.

Note:

If the IbmIsdn entry cannot be seen when you select a device while attempting to make a new connection, then you may have a driver problem.

To correct this we completed the following steps:

1. Removed the TCPIP protocol from the network configuration in Windows 95.
2. Removed the IbmIsdn Software Network Adapter.
3. Reinstalled Microsoft ISDN Accelerator Pack 1.1.

4. Reinstalled the IbmIsdn Software Network Adapter.

When prompted by Windows if we wanted to install drivers, we always kept the latest drivers. The most significant driver seemed to be ndis.vxd 4.1.1010. We accidentally replaced this with an older version and this caused the IbmIsdn Software Network Adapter to vanish.

7.7.2.2 Microsoft Dial-Up Networking 1.2

We tried to install Microsoft Dial-Up Networking 1.2 but it failed.

Despite numerous attempts we could not get the `IBM IbmIsdn Software Adapter` device to become visible to the Create Dial-Up Adapter screen. The driver was visible to the System Devices view but that was it. We removed Microsoft Dial-Up Networking 1.2 and replaced the ndis.vxd and other drivers with older versions during the install of Microsoft ISDN Accelerator Pack 1.1.

7.7.2.3 SSL Error: Cannot Initiate Control Session to Firewall

Every attempt to start an IPSec Tunnel resulted in the error:

```
SSL error
Cannot initiate control session to firewall.
Contact your firewall administrator.
SSL Init error code: 1
```

This was because the firewall was not running the process `/usr/sbin/sslrctd`.

We had forgotten to modify /etc/security/rcsfile.cfg to point to our keyfile.kyr. This caused the sslrctd process to fail after a few minutes. We corrected the sslfile entry in rcsfile.cfg and ran sslrctd manually.

7.7.2.4 Useful Commands

There are a couple of commands that will help you with the dynamic rules set by the firewall:

- The first is the dfdump command. It is undocumented but quite useful. On the firewall, typing dfdump in an xterm window will show all dynamic filter rules in the kernel in stanza format.
- The dfclr command will remove ALL dynamic filter rules from the kernel.

7.7.2.5 Rule Numbering for Dynamic Filter Rules

The rule numbering for dynamic filter rules are the 6XXX series rules in the local4 log. This is useful for debuging without using the local2 facility. Dynamic filters are checked before static rules, so every packet is passed through them and checked. The debug log on local2 can get huge quickly.

Chapter 8. Configuring Proxy Services and SOCKS

As we have discussed (2.1.2, “Proxy Servers” on page 13), address-based filters have a limited usefulness in keeping unwanted intruders away. For a more secure system, we want to break the sessions at the firewall boundary, and the proxy server gives us a technique for doing so. There are two different proxy servers: the normal proxy server and the transparent proxy server.

With the normal proxy server, users have to be predefined in the firewall as proxy users. They first have to access the firewall, authenticate themselves, then create a second session to access the target host. With the transparent proxy server, users logon to the firewall but user authenticaion is not required. Instead, users enter a special user ID format. This format is user@remote-host. The proxy server now tries to logon to the remote-host with the specified user ID. After this the user enters the password for the user ID at the remote host. There are no special accounts needed for using the transparent proxy. However, the transparent proxy is less secure, because it does not require user ID password authentication at the firewall. For this reason, the transparent proxy is only allowed from the secure interface.

We still need to use IP filters to enforce acceptable modes of access to the firewall. However, these filter rules can be much more restrictive than rules designed to allow traffic directly through the firewall.

8.1 User Administration

Each user, who will use the normal proxy servers, needs to have an account on the IBM Firewall system. These are normal AIX user accounts, except that they have a restrictive login shell. Note that the FTP proxy replaces the standard FTP daemon and it does not allow access to the firewall itself, as it only accepts `quote site` commands.

Adding users is simply done by selecting the users option in the GUI. Figure 154 on page 172 shows the general user screen.

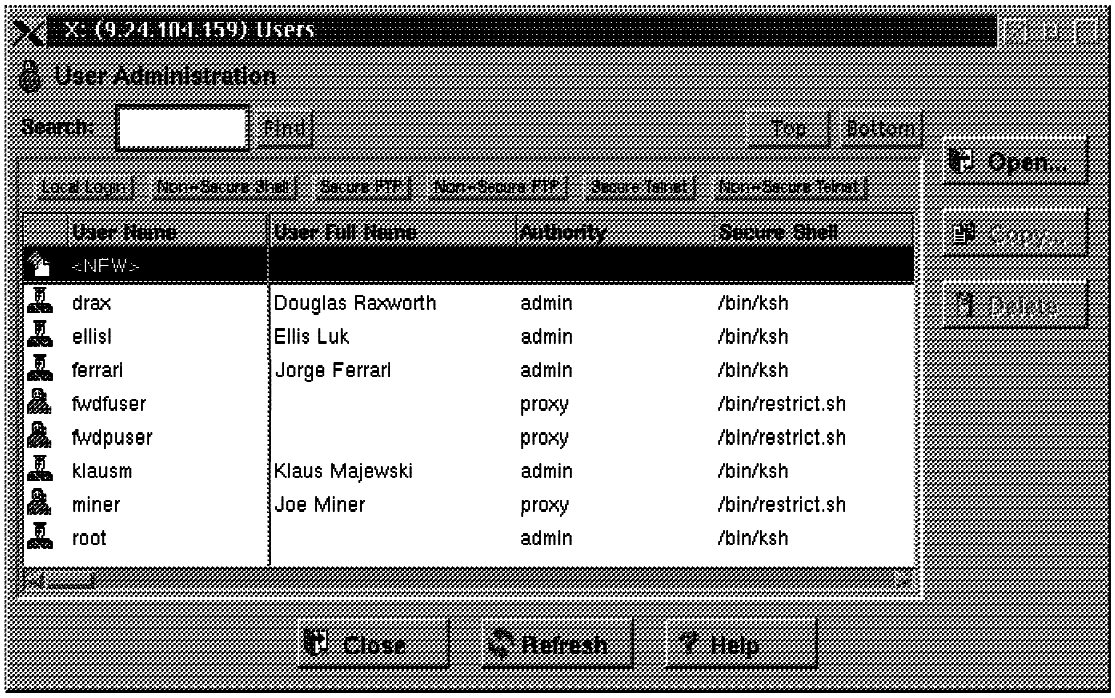


Figure 154. List User Screen

From the list users screen we can define, modify or delete users. Figure 155 on page 173 shows the dialog for adding a user.

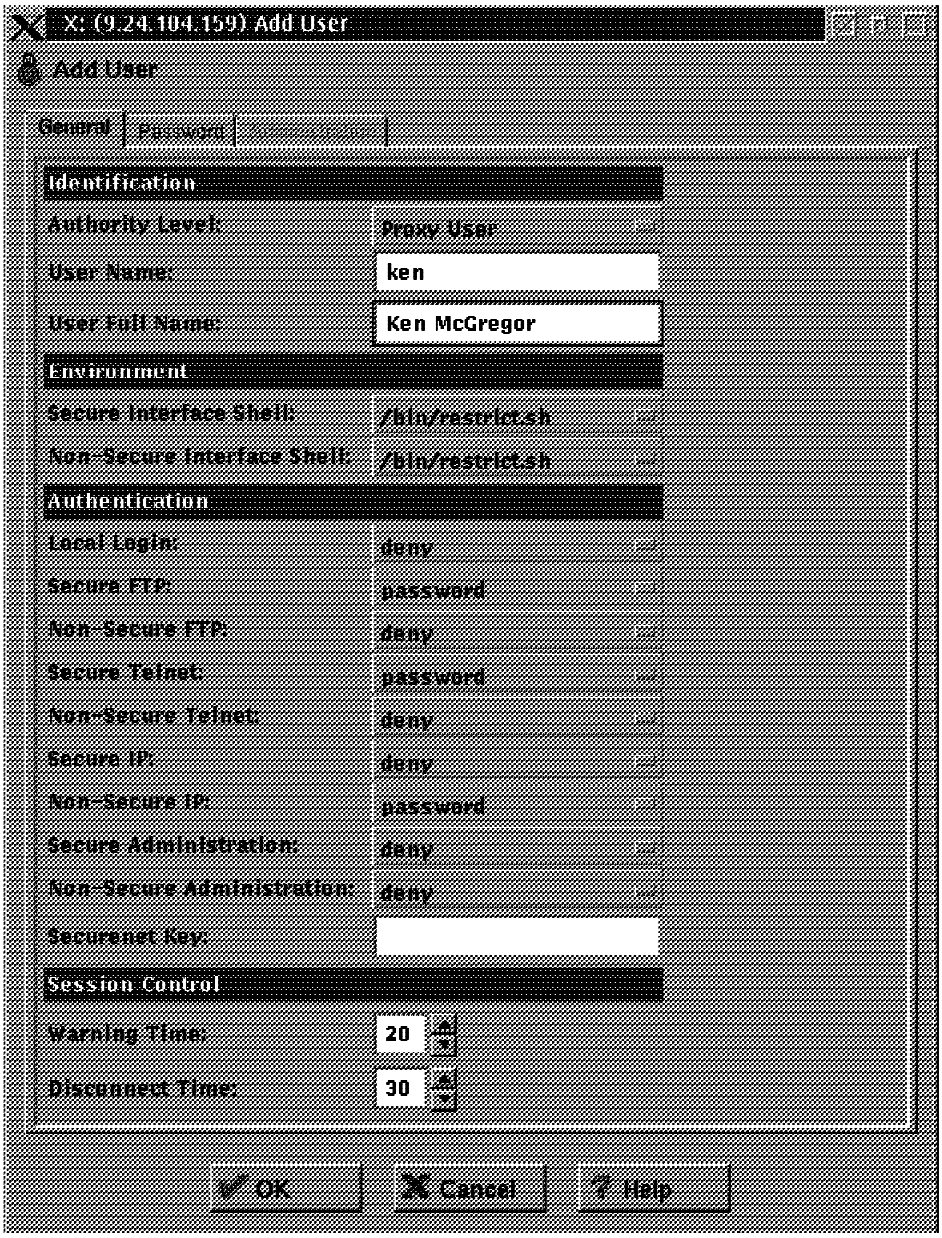


Figure 155. Administer General Screen

The options you can specify for each user are as follows:

Authority Level is either Administrator or Proxy User.

In this chapter we explain only the proxy user option.

User Name is the ID that the user will use to log in to the IBM Firewall.

User Full Name is the textual name of the user (for documentation purposes).

Secure Interface Shell is the command shell that the user will see when they log in using Telnet from the secure side of the firewall. The available shells are:

- restrict.sh - only gives commands for establishing sessions (telnet, gopher, ping, mosaic, finger, etc.)

- oneact.sh - only permits the user to establish a further Telnet session
- csh, ksh, bsh - the normal C, Korn and Bourne shells

The default secure interface shell is restrict.sh.

Nonsecure Interface Shell is the command shell that the user will see when they log in from the nonsecure side of the firewall. The same shells are available as for the secure side.

Local Login is the method to authorize login from the console. The options are:

- Password - normal AIX password validation
- SecureNet card - one-time pass key using Axent SecureNet
- SecureID card - one-time pass key using Security Dynamics SecureID
- User Supplied - use authentication method provided by you (for example, Bellcore's S/Key)
- Deny - prevent FTP login from the secure interface (default)
- None - no security required

Secure FTP is the method to use to authenticate the user when they log in to FTP from the secure side of the firewall. The same options as for previous field.

Non-Secure FTP is the method to use to authenticate the user when they log in to FTP from the nonsecure side of the firewall. The same options as for Local Login.

Secure Telnet is the method to use to authenticate the user when the user logs into Telnet from the secure side of the firewall. The same options as for Local Login.

Non-Secure Telnet is the method to use to authenticate the user when the user logs into Telnet from the nonsecure side of the firewall. The same options apply as for Secure Telnet.

Secure IP is the method to use to authenticate the user when the user logs in from the secure side of the firewall. The options are:

- Deny - prevent FTP login from the secure interface (default)
- Password - normal AIX password validation

Non-Secure IP is the method to use to authenticate the user when the user logs in from the nonsecure side of the firewall. The same options apply as for Secure IP.

Local Administration is the method to use to authenticate the administrator when he/she logs in from the secure network interface. The same options as for Local Login apply.

Remote Administration is the method to use to authenticate the administrator when he/she logs in from the nonsecure network interface. The same options as for Local Login apply.

Securenet Key enter the same key code that will be used to prime the Axent SecureNet Key.

Warning Time in minutes, is the maximum time of user idle before a warning message to disconnect the user.

Disconnect Time in minutes, is the maximun time of user idle before being disconnected. This time must be more than warning time.

You define the user as shown in Figure 154 on page 172 and select **OK**, but you first need set to the values of the password (Figure 156).

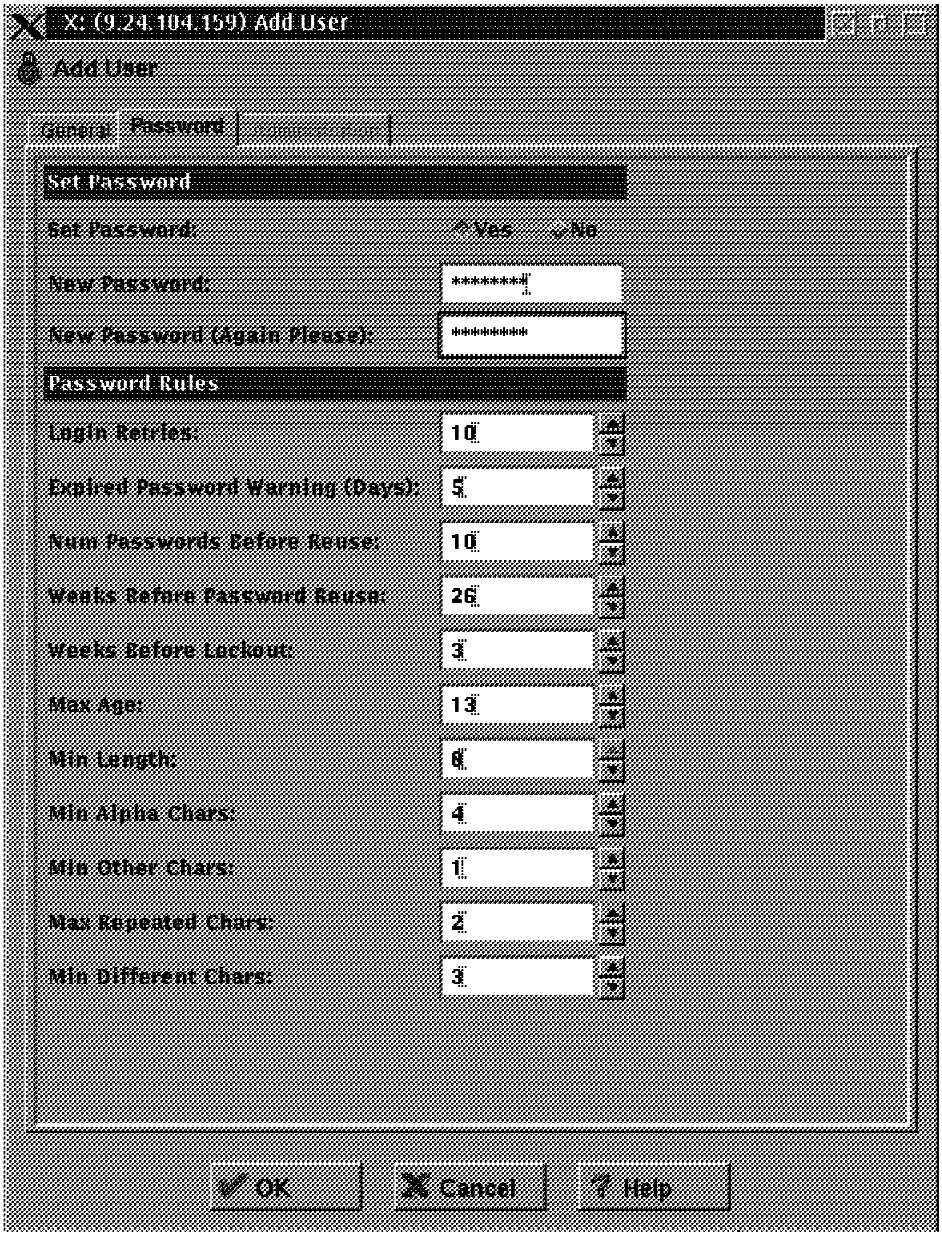


Figure 156. Defining the Password for New User

The options that you can specify are:

Set Password specify yes to assign the new password.

New Password provide a password.

New Password (Again Please) provide a password again, just to verify.

Login Retries maximum number of fail retries before lockout. 0 disables this option and 20 is the maximum number of login retries.

Expired Password Warning (Days) number of days before the password expires to receive a warning.

Num Passwords Before Reuse number of passwords before reuse; the maximum allowed is 20.

Weeks Before Password Reuse number of weeks before a password can be reused; 26 is recommended and 52 is the maximum.

Weeks Before Lockout number of week between password expiration and provide a new password to lockout the user. -1 disables this option; maximum is 26.

Max Age maximum password live time in weeks; 0 disables this option, and 52 is the maximum allowed.

Min Length minimum length of the password. 0 is no minimum and 8 is the maximum allowed.

Min Alpha Chars minimum of alphabetic characters in the password. 0 is no minimum and 8 is the maximum allowed.

Min Other Chars minimum of nonalphabetic characters in the password. 0 is no minimum and 8 is the maximum allowed.

Max Repeated Chars maximum of repeated characters in the password. 0 is no maximum; 8 is the maximum allowed.

Min Different Chars minimum of different characters between the new password and the old password. 0 disables this option and 8 is the maximum allowed.

The only user ID that you do not have to add in this way is root.

If you have many users to enter, the GUI may be a rather slow method to do it. You may want to consider using the command *fwuser*; see 4.3.7, “Command Line Proxy User Generation” on page 50 for details.

8.2 Transparent Proxy

Setting up the transparent proxy server is quite easy. From the main GUI screen, select **System Administration**, and then **Security Policy**, now you can choose to enable the transparent Telnet and/or FTP proxy. We enabled both in Figure 157 on page 177.

Figure 157. Enable Transparent Telnet and FTP Proxy

8.3 Filter Rules for Proxy Services

There is no point in setting up a proxy service if users can bypass it by routing through the firewall. We have already shown the filter rules for the Telnet proxy server (see “Telnet Using Proxy” on page 82), but we show them again here for completeness.

```
# Telnet from secure network to the Firewall
permit s.s.s.s sm.sm.sm.sm s.s.s.1 0xffffffff tcp     gt 1023 eq 23 secure local inbound
permit s.s.s.1 0xffffffff s.s.s.s sm.sm.sm.sm tcp/ack eq 23 gt 1023 secure local outbound

# Telnet from Firewall to the Nonsecure Network
permit n.n.n.1 0xffffffff 0 0 tcp     gt 1023 eq 23 nonsecure local outbound
permit 0 0 n.n.n.1 0xffffffff tcp/ack eq 23 gt 1023 nonsecure local inbound
```

Figure 158. Filter Rules for Telnet Proxy

The first pair of rules allows any access to the firewall from a node inside the secure network (you need this to get to the proxy server in the first place). The second pair of rules allows Telnet sessions started from the firewall to the nonsecure network.

These rules only allow access from Telnet clients inside the secure network to servers outside it. As we explained previously, there are serious security implications if you provide Telnet access from the nonsecure network to the secure network (see 2.1.2, “Proxy Servers” on page 13).

8.4 Examples of Using the Proxy Servers

Figure 159 is an example of a transparent Telnet proxy session. The user only starts a telnet session to the firewall and enters root@192.168.253.222 at the login prompt. After this only the password for root at 192.168.253.222 has to be entered.

```
# telnet rs60004
Trying...
Connected to rs60004.itso.ral.ibm.com.
Escape character is '[]'.

telnet (rs60004.itso.ral.ibm.com)

login: root@192.168.253.222
Trying...
Connected to 192.168.253.222.
Escape character is '[]'.

telnet (192.168.253.222)
AIX Version 4
(C) Copyrights by IBM and by others 1982, 1996.
login: root's Password:
*******************************************************************************
*                                                                             *
*                                                                             *
*  Welcome to AIX Version 4.2!                                                *
*                                                                             *
*                                                                             *
*  Please see the README file in /usr/lpp/bos for information pertinent to    *
*  this release of the AIX Operating System.                                  *
*                                                                             *
*                                                                             *
*******************************************************************************
Last login: Thu Mar 27 10:29:07 EST 1997 on /dev/pts/1 from aix1

x4
:root#
```

Figure 159. Transparent Telnet Proxy Session

Figure 160 on page 179 is an example of a user session using the FTP proxy. The user uses the ftp command to connect to the firewall (rs60004). Once there, they are authenticated, then they use quote site in order to reach their final destination.

```
# ftp rs60004
Connected to rs60004.itso.ral.ibm.com.
220-********************************************************************************
220-*                                                                              *
220-*                                                                              *
220-*  Welcome to AIX Version 4.1!                                                 *
220-*                                                                              *
220-*                                                                              *
220-*  Please see the README file in /usr/lpp/bos for information pertinent to     *
220-*  this release of the AIX Operating System.                                   *
220-*                                                                              *
220-*                                                                              *
220-********************************************************************************
220  rs60004.itso.ral.ibm.com FTP GATEWAY (Version 1.2 12/06/94 22:49:31) ready.
Name (rs60004:root): ken
Password:
230  To specify destination, type "quote site remote.host.com"
ftp> quote site ftp.cert.org
220 cert.org FTP server (Version wu-2.4(1) Mon Apr 3 16:53:11 EDT 1995) ready.
ftp> user anonymous
331 Guest login ok, send your complete e-mail address as password.
Password:
230-CERT Coordination Center FTP server.
230-
230 Guest login ok, access restrictions apply.
ftp> dir
200 PORT command successful.
150 Opening ASCII mode data connection for /bin/ls.
total 5
-rw-r--r--  1 cert     cert          454 Jun 13  1995 .message
drwxr-xr-x  2 cert     cert          512 Mar  3  1993 bin
drwxr-xr-x  3 cert     cert          512 Apr  3  1995 etc
drwxr-x--x  7 cert     cert          512 Mar 24  1995 incoming
drwxr-xr-x 15 cert     cert         1024 Feb  8 18:56 pub
226 Transfer complete.
ftp> quit
221  Goodbye.
#
```

Figure 160. FTP Proxy Session

In order to use the Telnet proxy, the user enters the telnet command to connect to the firewall (rs60004). They are authenticated by IBM Firewall, and then again use Telnet to reach the final destination (note that Telnet is a generic terminal emulator, so it could also be used to access other TCP services on different ports). Figure 161 on page 180 shows a typical session.

```
# telnet rs60004
Trying...
Connected to rs60004.itso.ral.ibm.com.
Escape character is '[]'.

telnet (rs60004.itso.ral.ibm.com)

login: ken
Password:
*******************************************************************************
*                                                                             *
*                                                                             *
*  Welcome to AIX Version 4.1!                                                *
*                                                                             *
*                                                                             *
*  Please see the README file in /usr/lpp/bos for information pertinent to    *
*  this release of the AIX Operating System.                                  *
*                                                                             *
*                                                                             *
*******************************************************************************
Currently logged in from rs600020.itso.ral.ibm.com
imported TERM=aixterm
Please enter what you would like set as TERM [aixterm]
Your TERM variable has been set to aixterm
Your DISPLAY variable has been set to rs600020.itso.ral.ibm.com:0
rs60004.itso.ral.ibm.com: cat /etc/passwd
cat not valid
rs60004.itso.ral.ibm.com: telnet 150.53.104.12
Trying...
Connected to 150.53.104.12.
Escape character is '[]'.

HP-UX ns1 A.09.05 A 9000/715 (ttyq1)

login: root
Password:

Value of TERM has been set to "aixterm".
WARNING:  YOU ARE SUPERUSER !!
/ >
```

Figure 161. Telnet Proxy Session

Notice that on the firewall the user, ken, has a restricted shell so he cannot issue any damaging commands (cat is the command shown here). Notice also that this example demonstrates the most common failing of security measures, user error. The proxy server is doing a good job of protecting the secure network, but the target machine (IP address 150.53.104.12) has been compromised because ken connected to it with the root user ID. Under Telnet, the password is sent in clear, so anyone could have captured it in the nonsecure network.

8.5 Results of Configuring the Proxy Servers

Having added the proxy user ID and activated the filters shown earlier, we found the following expected results:

- Telnet from any machine in the nonsecure network to the firewall machine failed with a time-out (because of the source address specifications in the filters)
- Telnet from the secure network to any machine in the nonsecure network and also the reverse timed-out (no filter rule to permit sessions to be routed)
- Telnet from the secure network to the firewall succeeded, and we logged on with ID ken. Then from this firewall login we found the following:

- Command `cat /etc/passwd` failed (due to the restricted shell)
- Command `telnet 150.53.104.12` succeeded (the filters allow the session to be set up)

8.6 HTTP Proxy

IBM Firewall 3.1 provides an HTTP proxy server which handles browser requests efficiently. It is a transparent proxy, so there is no user authentication required. Besides the HTTP protocol, the proxy also supports FTP, Gopher, WAIS and HTTPS (via SSL tunneling).

8.6.1 Filter Rules for HTTP Proxy

Before the HTTP proxy server can work properly, filtering rules must be set up to allow HTTP traffic. The rules for HTTP proxy server have been shown in 6.8, “HTTP - World Wide Web Sessions” on page 107. However, as we mentioned earlier, the HTTP proxy server also supports FTP, Gopher, WAIS and HTTPS. So if such services are allowed, you need to enable them on the packet filter as well. The filtering rules for these services are described in Chapter 6, “Examples of Rules for Specific Services” on page 71. Note that when these additional services are used, they are encapsulated in the HTTP connection between the internal client and the proxy.

Figure 162 on page 182 and Figure 163 on page 183 show the HTTP proxy connections required in the secure and nonsecure network respectively.

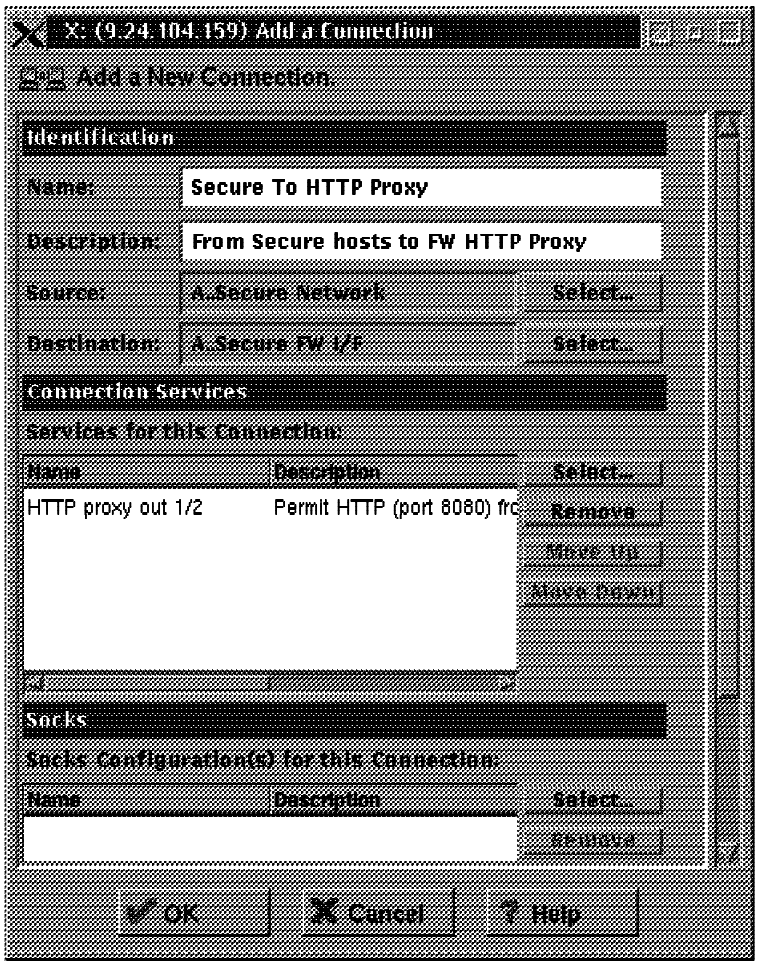


Figure 162. HTTP Proxy Connection on Secure Network

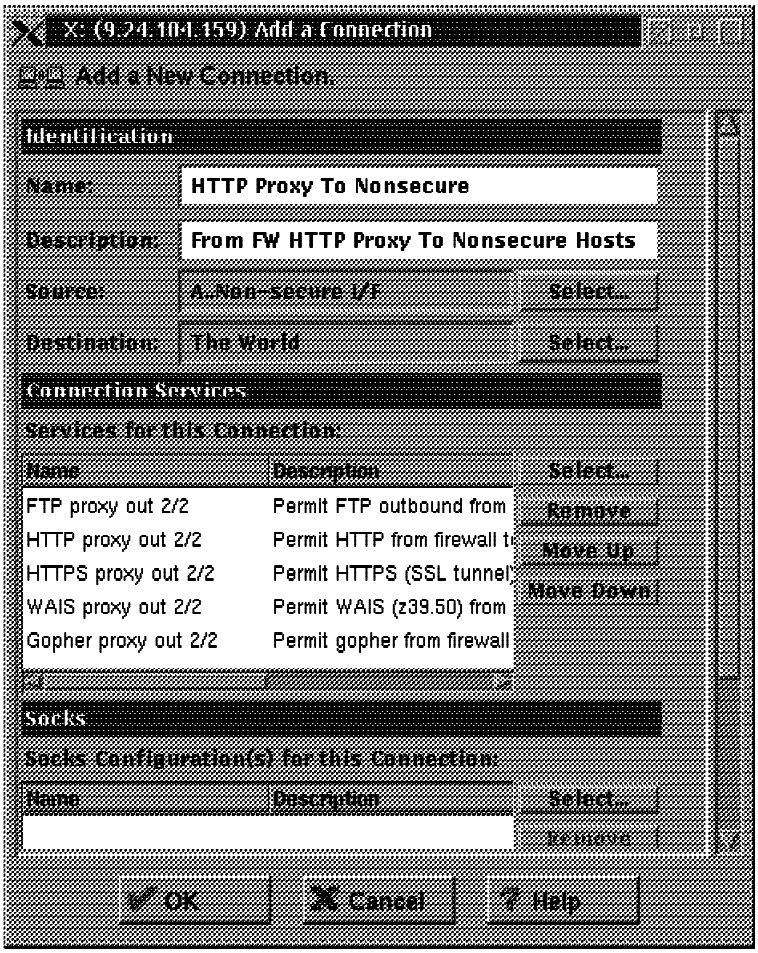


Figure 163. HTTP Proxy Connection on Nonsecure Network

8.6.2 Configuration of HTTP Proxy

The HTTP proxy server, by default, is not started up automatically at boot time. To make it start up at boot time, you need to modify the /etc/rc.tcpip file, and uncomment the following line:

```
#/usr/sbin/phttpd
```

You should also make sure that DNS queries are permitted from the firewall. Otherwise, the HTTP proxy cannot resolve any hostnames in the browser requests.

The configuration parameters of the proxy server can be modified using the GUI. Figure 164 on page 184 shows the HTTP proxy configuration screen.

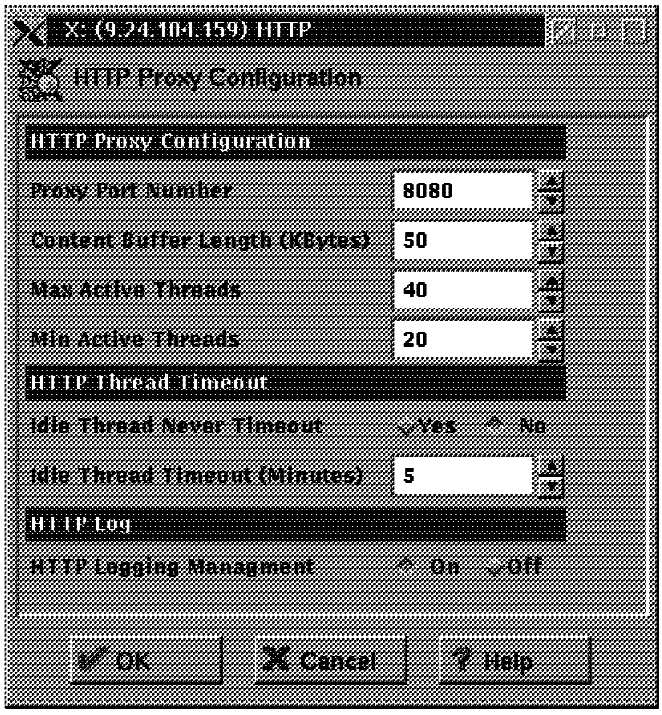


Figure 164. HTTP Proxy Configuration

The parameters that you can modify are:

Proxy Port Number

The port number the proxy should listen to for requests. If you change the port number, you must configure your filters to allow or disallow flow through the ports

Content Buffer Length

The size of the buffer for dynamic data generated by the server. Dynamic data is output from CGI programs, server-side includes, and API programs. It is data that does not come from a proxy.

Max Active Threads

The maximum number of threads that you want to have active at one time. If the maximum is reached, the proxy holds new requests until another request finishes and threads become available. Generally, the more power a machine has, the higher the value you should use for this parameter

Min Active Threads

The minimum number of threads that you want the proxy to have available for use. The server will not close threads below this minimum even if the threads are idle.

Idle Thread Never Timeout

The time for which the proxy should keep an idle thread available.

HTTP Logging Management

If this is on, the proxy logs startup/shutdown and all proxy requests to the syslog. Events are logged in local4.log.

8.7 Idle Proxy Connections

In this section, we discuss idle proxy connections. The purpose of a proxy server is to provide access to outside networks for internal users (or vice versa). There is no reason to allow users to establish these connections and then do nothing. Idle users tie up resources on the firewall. The idle proxy process disconnects all non-interactive sessions to the IBM Firewall after a specific period of idle time.

8.7.1 Setting Up Idle Proxy

We will describe how to set up idle proxy with examples. In order to set up this function, we should follow the next 3 steps:

1. Create a log file
2. Check out an idle privileged user file
3. Set up idle proxy as a cron job

8.7.1.1 Create a Log File

The idle proxy uses the syslog facility to write a log record. It uses facility name *local4*. If you want to record the syslog messages from idle proxy, you need to add an entry to the /etc/syslog.conf file, as follows:

```
local4.debug            /var/adm/messages
```

See Chapter 14, “Logging” on page 251 for more information about setting up syslog recording.

8.7.1.2 Update the Idle Privileged User File

When you install IBM Firewall, the system automatically creates a default /etc/security/fwpriv.users file. The file includes the following line:

```
DEFAULT 20 30
```

The records in this file have three fields, as follows:

- The first field is for user ID.
- The second field indicates the warning time, which is the maximum idle time in minutes before the user gets a warning message.
- The third field indicates the disconnect time, which is the maximum idle time in minutes before the user is disconnected.

Note that this default value specified in the /etc/security/fwpriv.users file is only applicable to non-root users. By default, connections started up by root would not timeout.

You can add entries to this file for specific user IDs, but the DEFAULT entry *must* exist in the file. You must not delete the DEFAULT entry, but you can change the idle time values that it specifies.

If you want to add a specific user's idle time, you can add a new entry for the user ID to the existing entries, either by editing the file or by using a SMIT dialog. We recommend the latter course. To do this, enter SMIT and select:

```
IBM Firewall V3R1 for AIX
     Users
          Idle Users
               Add
```

The SMIT display is shown in Figure 165.

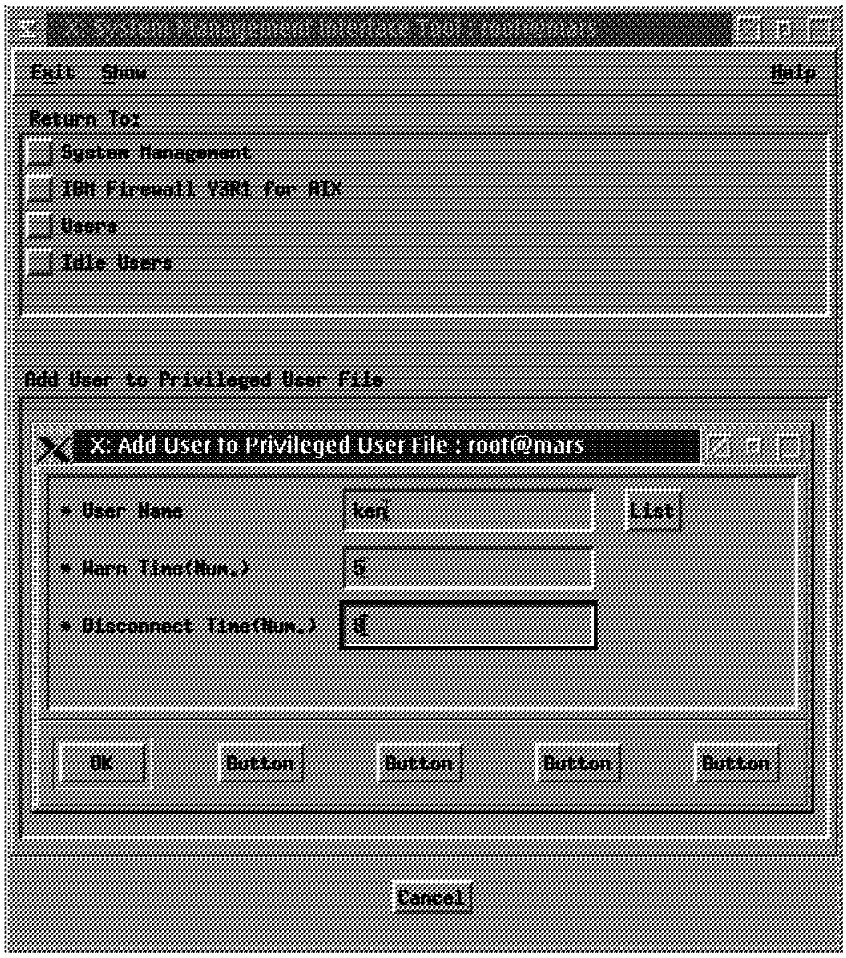


Figure 165. Add a User to the Privileged User File

When you click on **List** you are presented with a list of all the user IDs defined for proxy services. In Figure 165, we selected user ID ken with a warn time of 5 minutes and a disconnect time of 8 minutes. .

This caused the /etc/security/fwpriv.users file to be updated as follows:

```
DEFAULT 20 30
ken 5 8
```

After you have added a user, you should validate the /etc/security/fwpriv.users file. You can use the `fwidleout -v` command or select **Validate** in SMIT to do this (see Figure 166 on page 187).

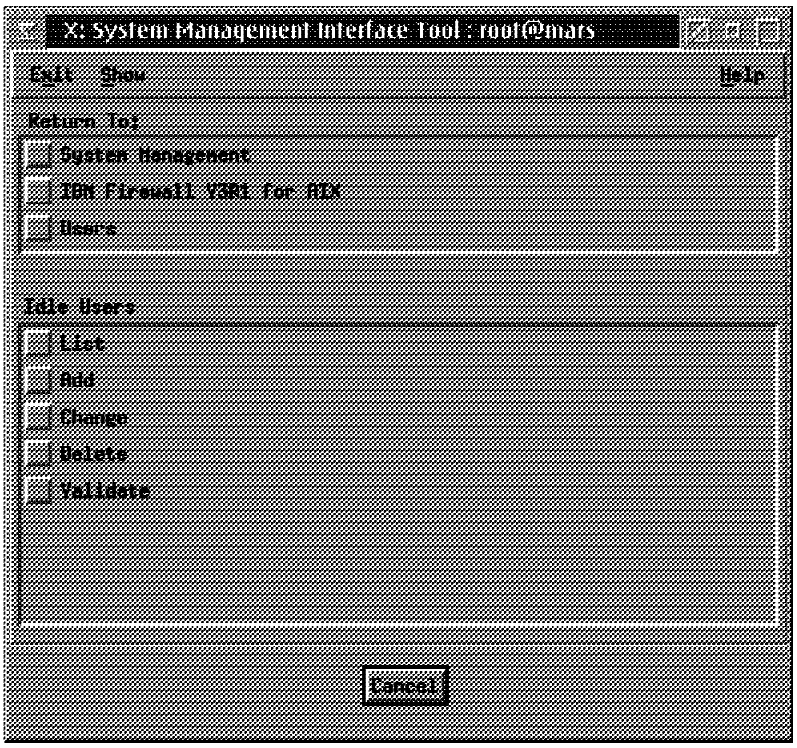


Figure 166. Validate Privileged User File

If you get any error messages, you should check your entries in /etc/security/fwpriv.users. To avoid any mistakes, we recommend that you use SMIT to add or change the privileged user file.

8.7.1.3 Set Up Idle Proxy as a Cron Job

So far we have simply defined the time-out configuration for idle proxy. The program that actually implements the time-out actions (sending messages and cancelling sessions) is called fwidleout.

We recommend that you run the idle proxy process as a cron job instead of issuing the fwidleout command from the command line. Figure 167 shows an example of a cron table entry for the root user ID (you can use the `crontab -e` command to edit root's cron table and `crontab -l` command to verify your changes).

```
0,5,10,15,25,30,35,40,45,50,55 * * * * /usr/bin/fwidleout
```

Figure 167. Cron Table Example for Idle Timeout

This entry sets up cron to run idle proxy every 5 minutes for every day of the year.

8.7.2 Sample Idle Proxy Environment

Because of interaction between the polling cycle for idle proxy and the activity of the user, the behavior may not always be what you would expect. We will show four examples of idle proxy behavior to illustrate this.

We will use the fwpriv.users file and cron table entry described previously in the examples.

8.7.2.1 Idle Proxy Example 1

User Activity	Time	Idle Proxy Activity
User ken logs in to IBM Firewall	10:00	
Ken enters ping command	10:02	
	10:05	fwidleout command executed by cron. Sees idle time for ken is 3 minutes (10:02-10:05)
Ken enters traceroute command	10:08	
	10:10	fwidleout command executed by cron. Sees idle time for ken is 2 minutes (10:08-10:10)
Ken logs out from IBM Firewall	10:14	

Notice that in this case, when the first fwidleout command was issued by a cron job, the idle time was only 3 minutes. Therefore, the idle proxy process did not send any messages or cancel Ken's session. However, between the first user command (ping) and the second command (traceroute), the idle time was actually six minutes. In fact, because the time between each user command and the next cron polling interval was below the warning threshold of five minutes, no action was taken. Similarly, because Ken logged out before the next scheduled poll (at 10:15), idle proxy did not notice that he was inactive for six minutes between 10:08 and 10:14.

8.7.2.2 Idle Proxy Example 2

User Activity	Time	Idle Proxy Activity
User ken logs in to IBM Firewall	10:00	
Ken enters ping command	10:04	
	10:05	fwidleout command executed by cron. Sees idle time for ken is 1 minute (10:04-10:05)
	10:10	fwidleout command executed by cron. Sees idle time for ken is 6 minutes (10:04-10:10). Sends a warning message to ken.
Ken enters traceroute command	10:13	
	10:15	fwidleout command executed by cron. Sees idle time for ken is 2 minutes (10:13-10:15).
Ken logs out from IBM Firewall	10:17	

Between the first user command (ping, at 10:04) and the first fwidleout command by cron (10:05), there was only one minute, so Ken did not get any messages. However, Ken did not enter any further command before the next time cron issued fwidleout, so it registered 6 minutes idle time. Therefore the idle proxy process sent a warning message to user ID ken. The warning message would look like this:

```
WARNING:!!! Spotted idle for 6 minutes. Time left before disconnection:2 minutes.
```

Next, Ken entered the traceroute command (at 10:13). Between the two commands there were 9 minutes idle time, which exceeds the disconnect time.

However, when the third fwidleout command was issued by cron, it registered an idle time of only 2 minutes (10:13-10:15). Therefore, Ken did not receive any messages and was not disconnected.

8.7.2.3 Idle Proxy Example 3

User Activity	Time	Idle Proxy Activity
User ken logs in to IBM Firewall	10:00	
Ken enters ping command	10:01	
	10:05	fwidleout command executed by cron. Sees idle time for ken is 4 minutes (10:01-10:05)
User ID ken disconnected by IBM Firewall	10:10	fwidleout command executed by cron. Sees idle time for ken is 9 minutes (10:01-10:10).

In this case, Ken was disconnected without any warning messages.

When the first fwidleout command was issued by cron, the idle time was 4 minutes. Therefore, Ken did not get any warning message. However, when the second fwidleout command was issued the idle time was 9 minutes which exceeds the maximum disconnect time of 8 minutes. Therefore, user ID ken was disconnected.

The disconnect message would look like this:

```
Disconnecting your session. Idle for 9 minutes.
Connection closed.
```

Idle proxy would also place an entry in syslog, as follows:

```
Jan 22 10:10:10 rs60004 : ICA2077i: Telnet Session ended for pid 8418 (9.24.104.241).
Jan 22 10:10:11 rs60004 : Disconnected ken at Mon Jan 22 16:55:10 CST 1996
```

8.7.2.4 Idle Proxy Example 4

User Activity	Time	Idle Proxy Activity
User ken logs in to IBM Firewall	10:00	
Ken enters ping command	10:03	
	10:05	fwidleout command executed by cron. Sees idle time for ken is 2 minutes (10:03-10:05)
	10:10	fwidleout command executed by cron. Sees idle time for ken is 7 minutes (10:03-10:10). Sends warning message.
User ID ken disconnected by IBM Firewall	10:15	fwidleout command executed by cron. Sees idle time for ken is 12 minutes (10:03-10:15).

In this case, the behavior is what you might expect; the user receives a warning message (10:10) and then is disconnected when he remains idle at 10:15.

What these examples illustrate is that if you want the behavior of idle proxy to be consistent, you should be careful when choosing idle timeout limits and the cron polling frequency. Even if you do choose reasonable values, the total idle time

before disconnection can vary considerably, depending on the time between the last user action and the following polling interval.

8.8 Using the SOCKS Server

When configuring IBM Firewall for the proxy application servers, we had to define user IDs, since it is the *user* who is being authenticated. With SOCKS, the authentication is primarily based on the *address* of the session source. For this reason, configuring SOCKS is very much like configuring IP filters.

However, in addition to authentication based on IP addresses, SOCKS can optionally check the user ID using the *Identification Protocol* as defined in RFC1413. This is a protocol that allows a client host to ask a server whether a user ID is valid. For a positive response, the user ID has to have activated a specific session between the same client and server hosts. If we consider user ken on node 9.24.104.241 attempting to start a Telnet session with node 150.53.104.12, the sequence would be as shown in Figure 168.

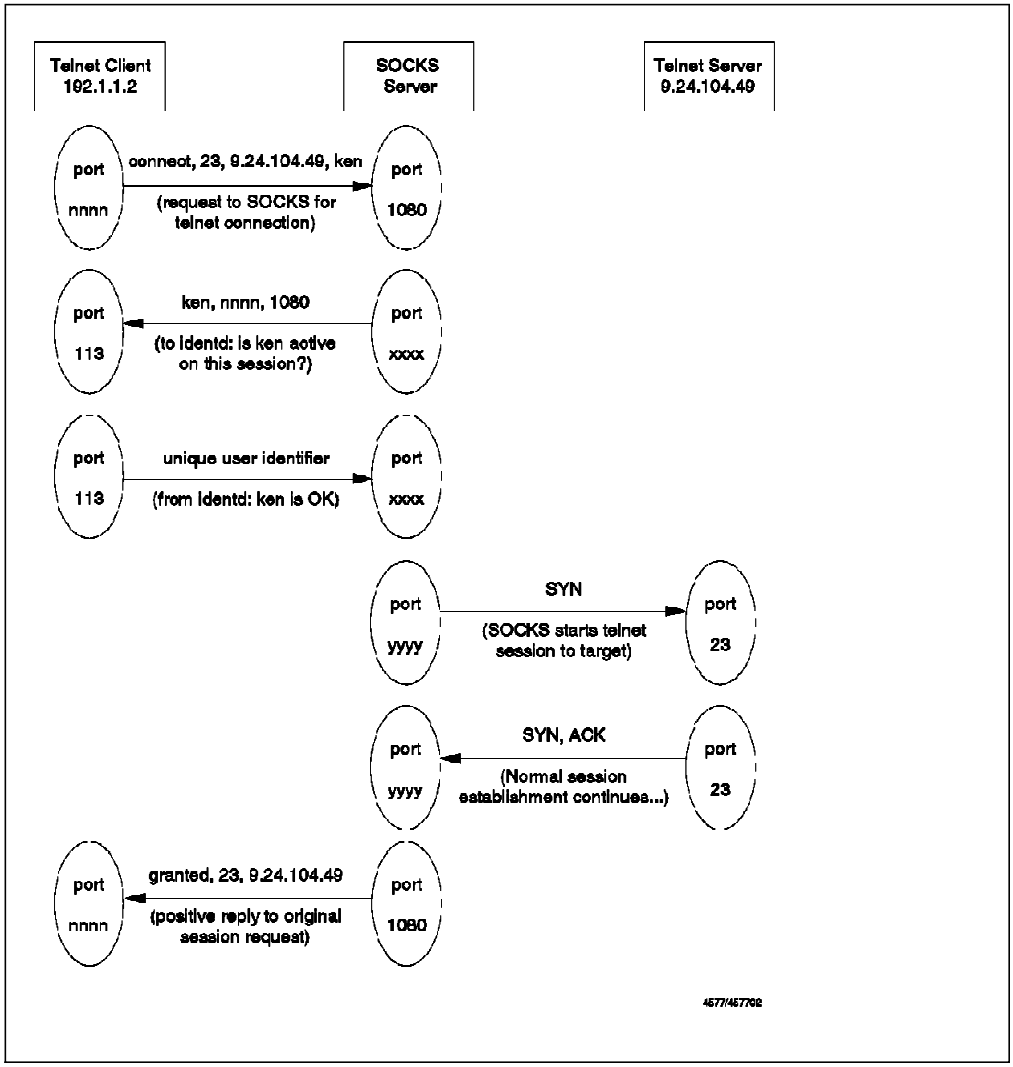


Figure 168. SOCKS Session Initialization with Ident Checking

Note that the ident request does not specify the nodes between which ken's session must exist. The ident daemon will take the address pair from the ident request itself.

You should be aware that using the ident option is not possible in every case, since not all TCP/IP implementations support it (in particular, single-user systems such as OS/2, DOS and Windows).

8.9 Configuration of SOCKS Server

SOCKS configuration entries are placed in file /etc/sockd.conf. The configuration file for the SOCKS client has a similar name (/etc/socks.conf), so be careful not to get them confused. You can build the server configuration file using a SMIT dialog, but we will explain the way to configure using the GUI.

To set up SOCKS services on the IBM Firewall, you need to:

1. Build filtering rules for connections from the secure network to the IBM Firewall SOCKS server.

   Figure 169 shows the GUI screen to build these connections.

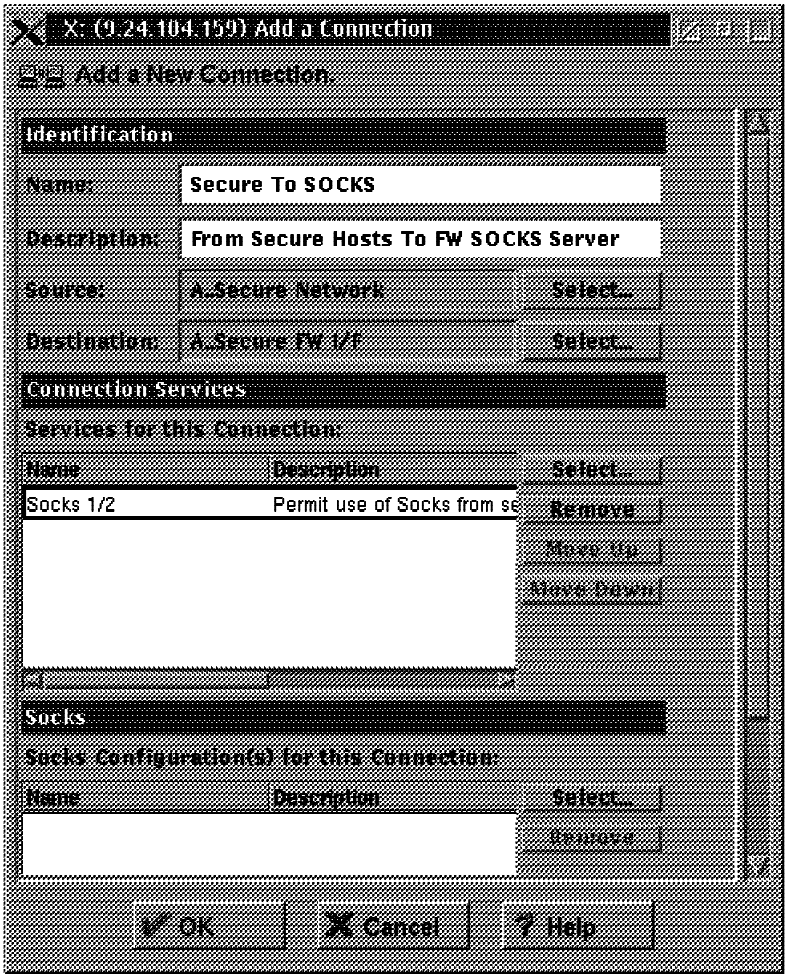


Figure 169. Connection from Secure Network to SOCKS Server

In this example, the entire secure network is allowed to connect to the firewall SOCKS server on port 1080.

2. Build filtering rules for connections from the IBM Firewall SOCKS server to the nonsecure network.

   Figure 170 shows the GUI screen to build these SOCKS outbound connections.

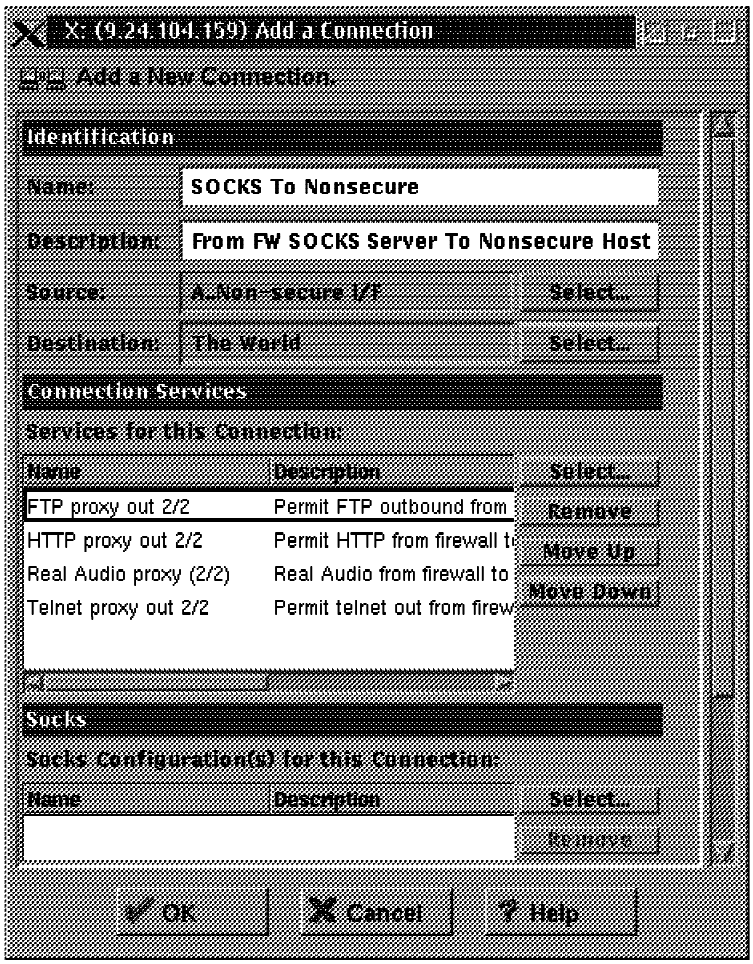


Figure 170. Connection from SOCKS Server to Nonsecure Network

   In this example, the permitted outbound proxy services are HTTP, Telnet, FTP and Real Audio.

3. Specify SOCKS server configuration for connections between the secure network and the nonsecure network.

   You build a connection with the secure network as the source object, the nonsecure network as the destination object, and the SOCKS templates as the services. Figure 171 on page 193 shows the GUI screen for such a connection.

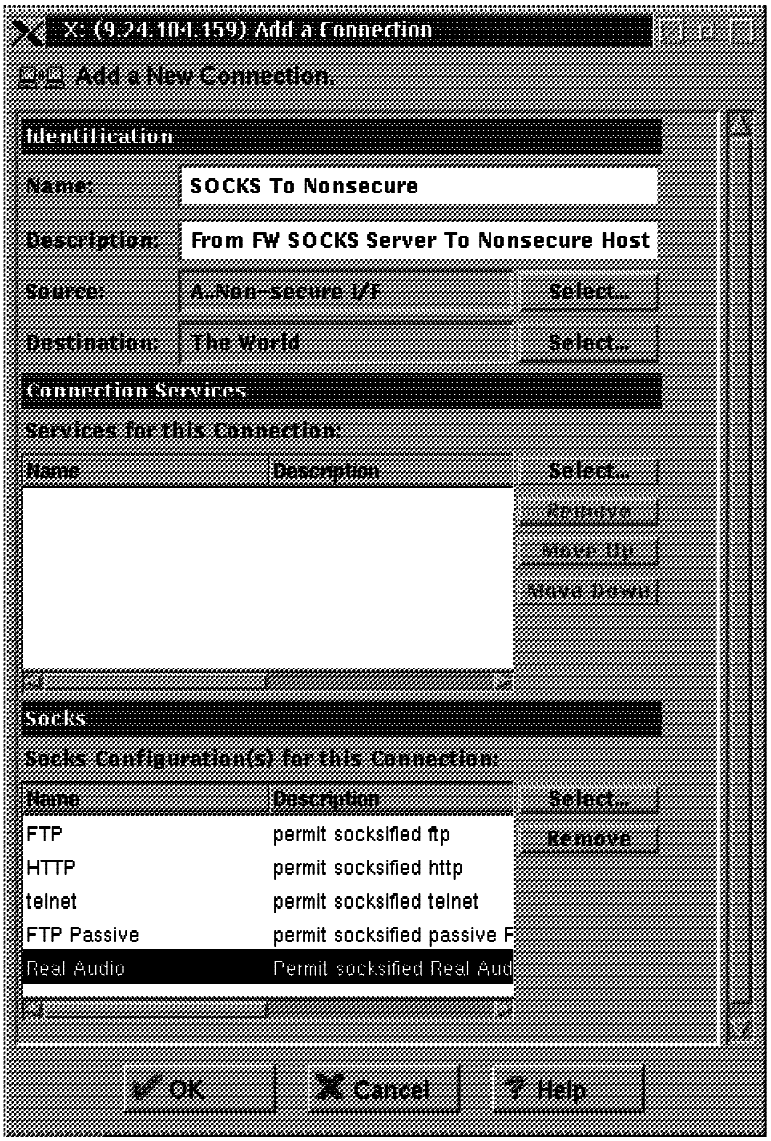


Figure 171. SOCKS Configuration from Secure to Nonsecure Network

You click on **List** to get a list of defined SOCKS templates. In this example, we selected five templates (FTP, HTTP, Telnet, FTP Passive and Real Audio).

The IBM Firewall 3.1 provides three standard SOCKS templates - HTTP, Telnet and FTP. However, the FTP Passive and Real Audio templates shown in Figure 171 are created by us. You can expand the SOCKSified services by creating new templates. The procedure to create templates is similar to creating filters, the only difference is the definition of the SOCKS template.

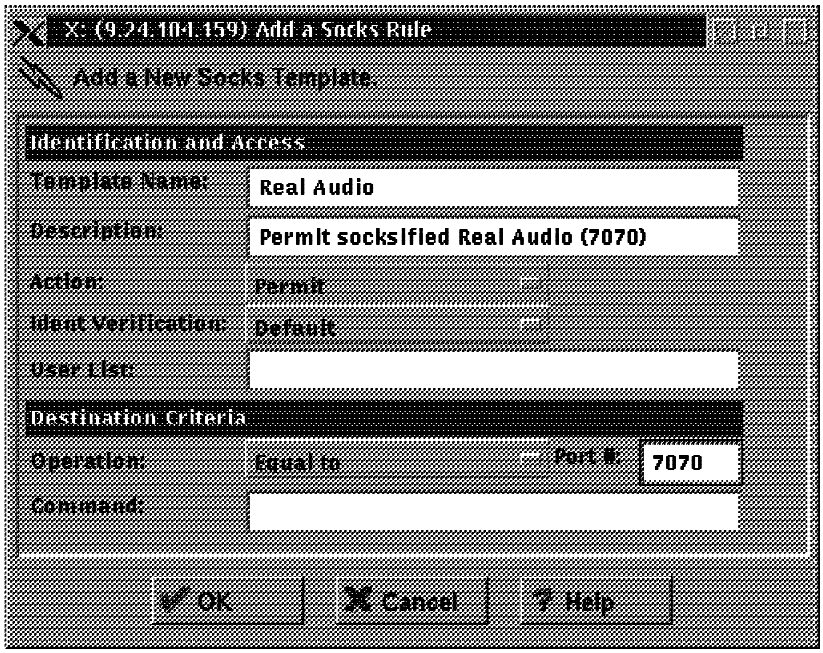


Figure 172. Adding a SOCKS Template

As you can see in Figure 172, the fields to be entered are similar to those for the filter rule definition. The meaning of these fields are as follows:

Template Name is the name of the template. Remember: use a naming convention.

Description for describes the function of the template.

Action is the action to take if a session request matches the conditions in the filter definition. Possible values are permit (allow the session) or deny (refuse session establishment).

Ident Verification specifies whether or not to use RFC1413 user ID checking (as discussed earlier).

User list lists user IDs that this configuration applies to, or the name of a file containing a list of IDs. These are the IDs on the originating host, and they must be listed separated by commas and without blanks.

Operation / Port Number is the acceptable destination port (same format as for the IP filters, see Table 5 on page 58).

Command in addition to the permit or deny actions taken when the criteria in the SOCKS configuration entry are met, it is possible to execute a command. *IBM Firewall For AIX User's Guide Version 3.1.1*, GC31-8419-00, shows some examples of using this to notify the user of failed login attempts.

8.9.1 SOCKS Server Files

The SOCKS server uses two files, /etc/sockd.route and /etc/sockd.conf. The sockd.route file is generated when you install IBM Firewall. If you later modify or add some interface, you will have to modify this file manually.

8.10 Configuration of SOCKS Client

The client configuration file, /etc/socks.conf, is simpler than /etc/sockd.conf on the server. The /etc/socks.conf file describes, for each destination, whether to use a SOCKS server, normal connection, or to prevent the connection attempt. It can also define the SOCKS server to use for a particular connection, the user ID(s) for which a particular profile applies and a command option, like the command option in the SOCKS server configuration.

For our environment (see Figure 121 on page 145), we created a simple two-line configuration file on the secure network machine:

```
direct 9.24.104.0 255.255.255.0
sockd @=9.24.104.27 0.0.0.0 0.0.0.0
```

This says to use direct TCP connections when connecting to addresses on the 9.24.104 network. For every other connection, use the SOCKS server at the IP address 9.24.104.27

8.11 Using SOCKS Services

In order to make use of a SOCKS server, you need to have a modified *SOCKSified* client program that will direct the session to the SOCKS port on the server and handle the connect request/response sequence. In general, WWW browsers (such as Netscape Navigator, NCSA Mosaic or Microsoft Internet Explorer) provide built-in SOCKS support. SOCKSified versions of many other application clients are available from various Internet sites, for example:

```
*AIX       rxgopher 1.3.1   ftp.nec.com/pub/security/socks.cstc/socks4/client
*AIX       rMosaic 2.0      ftp.nec.com/pub/security/socks.cstc/socks4/client
Unix       finger           ftp.www.nec.com/pub/security/socks.cstc/socks4/socks.cstc.4.2.2.tar.gz
Unix       ftp              ftp.www.nec.com/pub/security/socks.cstc/socks4/socks.cstc.4.2.2.tar.gz
Unix       telnet           ftp.www.nec.com/pub/security/socks.cstc/socks4/socks.cstc.4.2.2.tar.gz
Unix       whois            ftp.www.nec.com/pub/security/socks.cstc/socks4/socks.cstc.4.2.2.tar.gz
OS/2 Warp  Any TCP/IP App   Retrieve Software Updates from the Internet Connection Folder
Windows    Any Winsock App  www.socks.nec.com/SOCKSCap.html
SGI/Sun    rMosaic 2.0      ftp.nec.com/pub/security/socks.cstc/soc      ks4/client

Unix = AIX, HP-UX, Solaris, SunOS, SCO/ODT, Linux, and others
* = tested with IBM Firewall
```

Figure 173. Some Sources for SOCKSified Client Code

Also, several manufacturers of TCP/IP implementations are incorporating SOCKS support into their products (see E.1, “Aventail AutoSOCKS” on page 288 for more detail). For general information about SOCKS, consult the SOCKS Home Page at http://www.socks.nec.com/.

To invoke the AIX clients, we did the following:

1. Downloaded the files with the FTP proxy
2. Renamed telnet, ftp and finger programs to stelnet, sftp and sfinger
3. We had to change the file mode for these programs as follows:

   ```
   secured:/usr/bin > ls -al telnet
   -rwxr--r--   1 root     system    160270 Feb  8 13:49 telnet
   secured:/usr/bin > chmod 755 telnet
   ```

Using the SOCKSified clients was then simple. We just entered the normal command (for example, `telnet 150.53.104.12`) and had access to the host.

8.12 Creating a SOCKSified Client Application

It is simply a matter of recompiling the code with aliases for the common TCP system calls. Use the aliases from the following list that are appropriate for your code.

- -Dconnect=Rconnect
- -Dgetsockname=Rgetsockname
- -Dbind=Rbind
- -Daccept=Raccept
- -Dlisten=Rlisten
- -Dselect=Rselect

For example, if your code includes the TCP *listen* function, recompile it with a -Dlisten=Rlisten flag.

You also have to provide the linker with access to the libsocks library, located in /usr/lib. The full compile command is:

```
cc -o socks client.c -lbsd -lsocks -Dconnect=Rconnect (plus other aliases)
```

Having created the executable program, you need to make sure that the SOCKS configuration in /etc/socks.conf is set correctly.

The final step to make this a secure environment is to define filters that prevent direct routing of the connection through the firewall. We have already shown the filter rules for the Telnet proxy server in the previous chapter (see “Telnet Using SOCKS” on page 84), but we show them again here for completeness:

```
# Connection from the client to the SOCKS Server
permit s.s.s.s sm.sm.sm.sm s.s.s.1 0xffffffff tcp     gt 1023 eq 1080 secure local inbound
permit s.s.s.1 0xffffffff s.s.s.s sm.sm.sm.sm tcp/ack eq 1080 gt 1023 secure local outbound

# Telnet from Firewall to the Nonsecure Network
permit n.n.n.1 0xffffffff 0 0 tcp     gt 1023 eq 23 nonsecure local outbound
permit 0 0 n.n.n.1 0xffffffff tcp/ack eq 23 gt 1023 nonsecure local inbound
```

Figure 174. Telnet from Secure Network to Nonsecure Network using SOCKS

The first pair of rules allows the SOCKS clients to contact the SOCKS server on port 1080. The second pair of rules allows the firewall to contact the external Telnet server.

Chapter 9. RealAudio Support

In this section we will discuss the IBM Firewall 3.1 support of the RealAudio protocol. IBM Firewall 3.1 is able to identify RealAudio connections and support them in a secure way, in order to receive live or on-demand audio from the Internet without exposing the security of the private network.

9.1 RealAudio Protocol

RealAudio was developed by Progressive Networks in early 1994. It is unusual in that it is a hybrid protocol, a combination of both TCP and UDP. When a user clicks on a RealAudio link from a Web browser, a connection is established using TCP/IP to the RealAudio server. The server tells the player (implemented as a helper application on the browser) what stream to play, the player opens a UDP port for reception of the audio stream and then requests the server to start sending it.

The objective of RealAudio is to provide streaming audio over a limited bandwidth connection. That is why it uses UDP; a datagram service has a much lower overhead than a connection-oriented service, which makes it practical for use over low-speed modem connections.

RealAudio uses both TCP and UDP ports, as follows:

1. TCP port 7070 for the control connection
2. UDP ports greater than 1023 for incoming traffic

RealAudio also supports TCP-Only and HTTP-Only modes if you cannot allow the UDP traffic.

9.2 RealAudio and IBM Firewall 3.1

You have four different possibilities to configure the RealAudio traffic through the IBM Firewall 3.1:

1. Route the RealAudio control connection through the IBM Firewall 3.1 and use Network Address Translation to hide the internal IP adresses. The IBM Firewall 3.1 will automatically create dynamic filter rules for incoming UDP packets.
2. Use RealAudio's free proxy from http://www.real.com.
3. SOCKSify the client's TCP/IP stack and use SOCKS to get through the IBM Firewall 3.1; see 8.11, “Using SOCKS Services” on page 195.
4. Use HTTP protocol to go through IBM Firewall HTTP proxy.

Each of these approaches will hide the secure network's IP adresses.

9.2.1 RealAudio Routed through the IBM Firewall 3.1

In order to handle the RealAudio traffic, the IBM Firewall 3.1 has to permit two connections to be routed through. The first one is the TCP connection from a client's high port to the RealAudio server's port 7070. The second one is a UDP connection for incoming streams from the RealAudio server to the client. Because these connections are routed through the IBM Firewall 3.1, we have to hide the internal network adresses using Network Address Translation.

The UDP connection does not need to be configured, because when IBM Firewall 3.1 detects a RealAudio TCP control connection it interprets the data stream and dynamically creates a temporary UDP filter rule for the RealAudio stream. When the TCP control connection is closed, this temporary rule for UDP packets is deleted from the filter configuration. Figure 175 illustrates the connections.

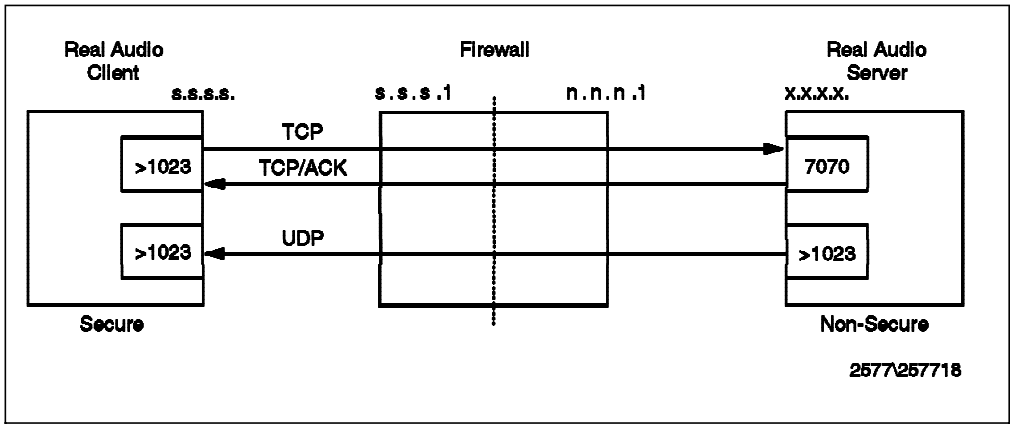


Figure 175. Connection across IBM Firewall 3.1

At the time of testing we noticed that dynamic filter rules did not work due to changes in the RealAudio protocol. This is fixed by APAR IR37027.

The RealAudio must first be enabled in the IBM Firewall 3.1. The RealAudio configuration in IBM Firewall 3.1 can be done using SMIT or the new GUI interface. First it is necessary to set up the default values:

Server Port Number This is the port used to establish the RealAudio control connection over TCP. 7070 is the standard port to use. This has to match the configuration of the RealAudio server.

Maximum Concurrent Sessions This is the maximum number of connections (simultaneous connections) that IBM Firewall 3.1 will allow for RealAudio.

To set up these values, select the following sequence from the SMIT main menu:

```
IBM Firewall V3R1 for AIX
     System Administration
          RealAudio
```

Or you can select the RealAudio option, inside the system administration folder in the new GUI Interface. Figure 176 on page 199 shows the resulting screen in the GUI.

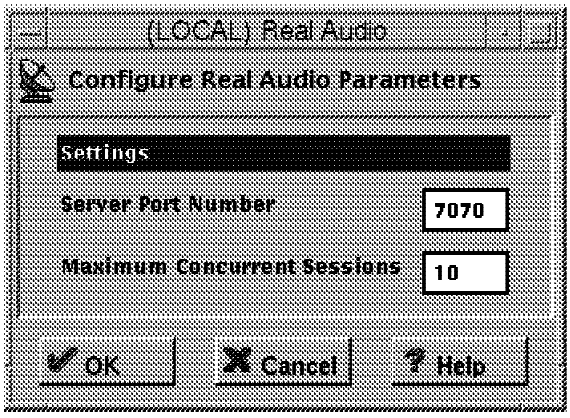


Figure 176. RealAudio Defaults Values

After setting up the default values, press Enter.

Now you need to permit users to establish connections to the RealAudio server. For this it is necessary to add filter rules to permit TCP connection using port 7070. To add this connection you need to select the source object (Secure Network object) and the destination object (World object). Additionally you need to select the service named "RealAudio", which has 4 rules:

```
permit  tcp      gt 1023 eq 7070   secure      route   inbound   l=n  f=y
permit  tcp      gt 1023 eq 7070   non-secure  route   outbound  l=n  f=y
permit  tcp/ack  eq 7070 gt 1023   non-secure  route   inbound   l=n  f=y
permit  tcp/ack  eq 7070 gt 1023   secure      route   outbound  l=n  f=y
```

Remember to regenerate the connection rules and activate them.

You should have your Network Address Translation configured and activated before you do the following steps to add your RealAudio client's IP address to Network Address Translation tables. See Chapter 10, “Network Address Translation” on page 205 for instructions how to configure and activate the Network Address Translation.

Check if your RealAudio client's IP address is translated. If it is not, then add your RealAudio client's IP address to the Network Address Translation tables.

1. Choose **NAT** from the GUI main screen.
2. Choose **Setup.**
3. Select **new object** and push the **Open** button.
4. Choose **Type of NAT** from the pull-down menu.
5. Push **Select** to choose Registered IP Address.
6. Select your RealAudio client from the Network Object List and push **OK**.
7. Push **OK** to accept values.
8. Push **Activate**.
9. Push **Execute** to Activate/Update Configuration.
10. Push **Close**.
11. Push **Close**.

Configure your internal network routing so that all the packets from the RealAudio client will go through the IBM Firewall 3.1. We used The RealAudio Client Version 4.0. This is how we configured the RealAudio client.

1. Select **View** pull-down menu from the RealAudio Client.
2. Select **Preferences** from the View pull-down menu.
3. In the Preferences window, click the **Transport** tab.
4. Click the **Use specified transports** option.
5. Click **Specify transports** button.
6. Click **Use TCP to Connect to Server** option.
7. Check **Attempt to use UDP for static content, and for live content not available via Multicast**.
8. Check **Attempt to use TCP for all Content**. (If UDP connection cannot be established client will use TCP.)
9. Push **OK**.
10. In the UDP Port box, check **Use specific UDP port:7070**.
11. Push **OK**.

9.2.2 RealAudio with RealAudio Proxy

There is no RealAudio proxy in the IBM Firewall 3.1. You can download the free source code for the RealAudio proxy from the URL http://www.real.com. While reading RealAudio's Web pages look for the service and support pages. There is an entry for the frequently asked questions and from there click on one of the RealAudio players. Then you will find information about the Firewall Administrator's Proxy Kit and the source code. The source code is free if you use it only for RealAudio traffic. All the information you need for the installation is in those Web pages. The RealAudio Proxy is a transparent proxy, so you do not have to create proxy users in order to use it. Figure 177 illustrates the connections needed for RealAudio to traverse through IBM Firewall 3.1.

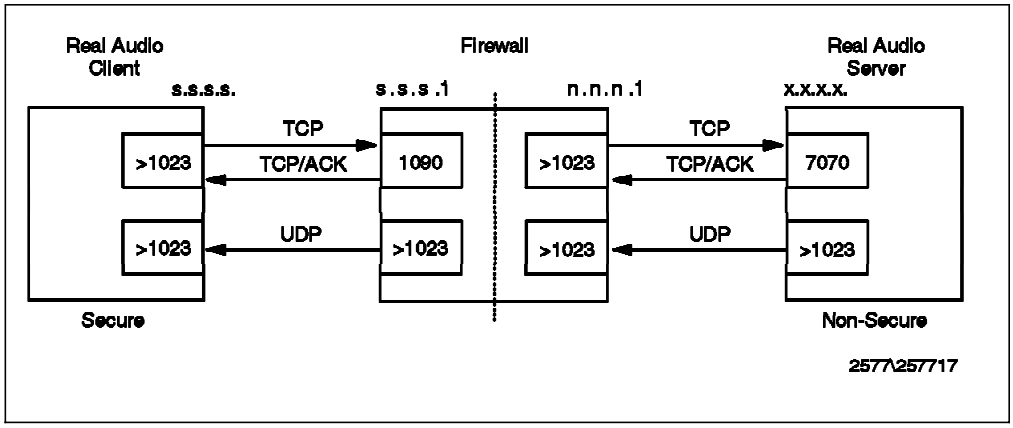


Figure 177. Connection across IBM Firewall 3.1 Using RealAudio Proxy

Now you need to permit users to establish connections to the RealAudio server. You need to make two connections. The first connection is from the RealAudio client to the firewall secure adapter. The second connection is from Firewall's nonsecure adapter to World. For the first connection add filter rules to permit the TCP connection from the secure network to the firewall using port 1090. You can use service HTTP proxy out (1/2) and rules that belong to it as a model for

your new service. Additionally you must allow the RealAudio client to receive UDP packets from higher ports of the IBM Firewall 3.1. Write three new rules:

```
permit  tcp      gt 1023 eq 1090   secure      local   inbound   l=n  f=y
permit  tcp/ack  eq 1090 gt 1023   secure      local   outbound  l=n  f=y
permit  udp      gt 1023 gt 1023   secure      local   outbound  l=n  f=y
```

Make a new service called RealAudio proxy (1/2) and put in it the rules you just created. Remember that the direction of the flow is important in these rules as shown in Figure 178.

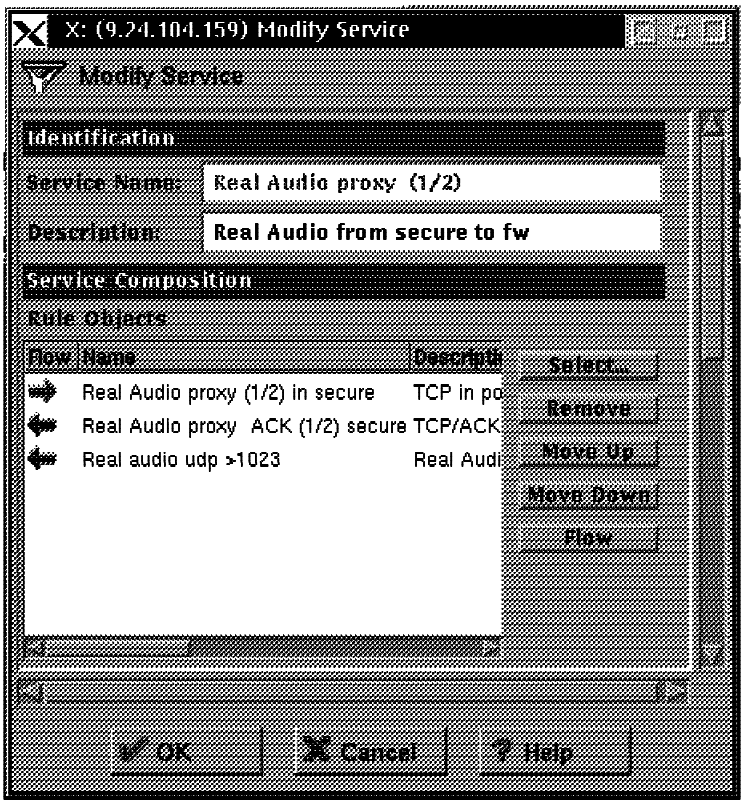


Figure 178. Rules and Their Flow in Service Real Audio Proxy (1/2)

To add the connection you need to select the source object (Secure Network object) and the destination object (the secure interface of the firewall object). Additionally you need to select the new service you just created.

For the second connection add filter rules to permit TCP connection from the nonsecure network to The World. You can use service HTTP proxy out (2/2) and rules that belong to it as a model for your new service. Additionally you must allow the IBM Firewall 3.1 to receive UDP packets from higher ports of the RealAudio server. Write three new rules:

```
permit  tcp      gt 1023 eq 7070   non-secure  local   outbound  l=n  f=
permit  tcp/ack  eq 7070 gt 1023   non-secure  local   inbound   l=n  f=
permit  udp      gt 1023 gt 1023   non-secure  local   inbound   l=n  f=
```

Make a new service called "RealAudio proxy (2/2)" and put in it the rules you just created. To add the second connection you need to select the source object (nonsecure interface of the firewall object) and the destination object (The World object).

Configure your RealAudio client to use the RealAudio proxy for RealAudio content. We did it in the following steps:

1. In your RealAudio client, click **Preferences** from the View menu.
2. In the Preferences window, click the **Transport** tab.
3. Click the **Use Specified Transport** option.
4. Click the **Specify Transports** button.
5. Click **Use TCP to Connect to Server** option.
6. Check **Attempt to use UDP for static content, and for live content not available via Multicast**.
7. Check **Attempt to use TCP for all Content**. (If UDP connection cannot be established client will use TCP.)
8. Push **OK**.
9. Click **Proxy** tab.
10. Click **Use Proxy** option.
11. Write your firewall's secure interface IP address into the **RealPlayer Proxy** field. Specify the port number 1090 in the **Port** field.
12. Push **OK**.

9.2.3 RealAudio with SOCKS

RealAudio client does not support SOCKS. You should socksify your RealAudio client's TCP/IP stack in order to get through the IBM Firewall 3.1. There are several software packages that will socksify a TCP/IP stack. See 8.11, “Using SOCKS Services” on page 195 to get the list of packages.

Create a new SOCKS object that will allow the traffic into port 7070. This requires creating a new connection from the secure network to The World. Do not specify any services to that connection. Instead, you will put in the SOCKS section of that connection the SOCKS object you just created.

In addition, you need to create two new connections very similar to 9.2.2, “RealAudio with RealAudio Proxy” on page 200. The IBM Firewall 3.1 SOCKS does not support UDP packets, so we have to configure the RealAudio client to use only TCP. Figure 179 illustrates the connections.

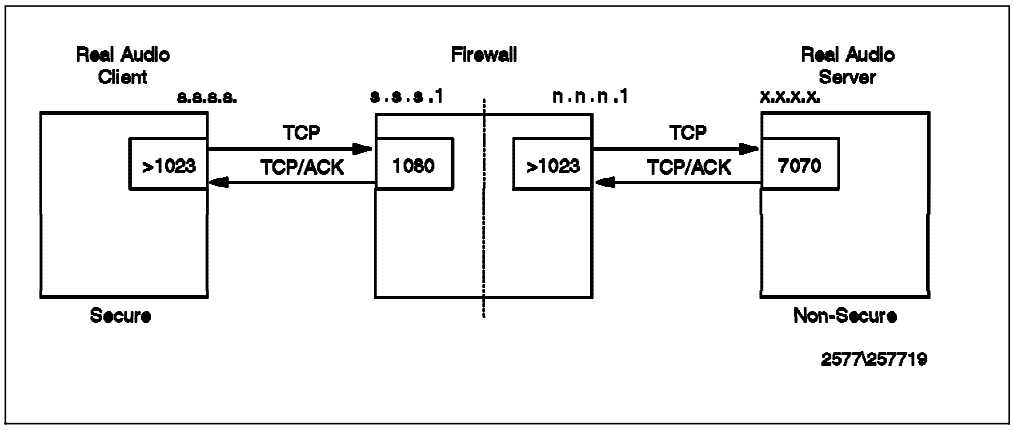


Figure 179. Connection across IBM Firewall 3.1 Using RealAudio with SOCKS

Configure your RealAudio client to use TCP only to access RealAudio content. Do it this way:

1. In your RealAudio client, click **Preferences** from the View menu.
2. In the Preferences window, click the **Transport** tab.
3. Click the **Use Specified Transport** option.
4. Click the **Specify Transports** button.
5. Click **Use TCP to Connect to Server**.
6. Check box **Attempt to use TCP for all content**.
7. Push **OK**.
8. In the Preferences window, click the **Proxy** tab.
9. Make sure that **Use Proxy** box is not checked.
10. Push **OK**.

9.2.4 RealAudio with HTTP

This case is the same as 8.6, “HTTP Proxy” on page 181. Figure 180 illustrates the connections.

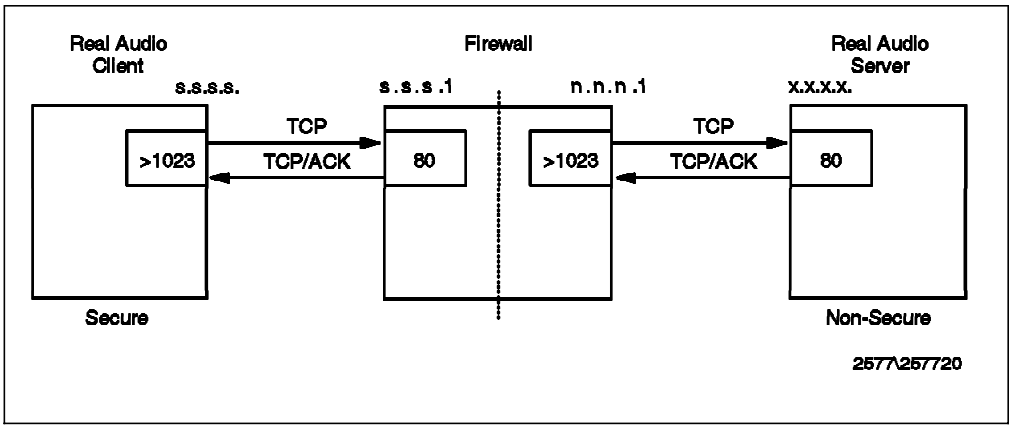


Figure 180. Connection across IBM Firewall 3.1 Using RealAudio with HTTP

Configure your RealAudio client to use the HTTP protocol and HTTP proxy for RealAudio content. We did it in the following steps:

1. In your RealAudio client, click **Preferences** from the View menu.
2. In the Preferences window, click the **Transport** tab.
3. Click the **Use Specified Transport** option.
4. Click the **Specify Transports** button.
5. Click the **Use HTTP Only** option.
6. Push **OK**.
7. Click **Proxy** tab.
8. Click **Use Proxy** option.
9. Write your firewall's secure IP address or HTTP proxy IP address into the **Http Proxy** field. Specify the HTTP port in the **Port** field.
10. Push **OK**.

Chapter 10. Network Address Translation

In this section we discuss Network Address Translation (NAT). Originally NAT was suggested as a short-term solution to the problem of IP address depletion. In order to be assured of any-to-any communication in the Internet, all IP addresses have to be officially assigned by the Internet Assigned Numbers Authority (IANA). This is becoming increasingly difficult to achieve, because the number of available address ranges is now severely limited. Also many organizations have in the past used locally assigned IP addresses, not expecting to require Internet connectivity.

The idea of NAT is based on the fact that only a small part of the hosts in a private network are communicating outside of that network. So if we can devise a technique to assign official addresses to hosts that are used only when they need to communicate outside the private network, then only a small number of official addresses are required.

This is what NAT does; it takes the IP address of an outgoing packet and dynamically translates it to an official address. For incoming packets it translates the official address to an internal address. We now can use NAT for a solution for networks that have private address ranges or illegal addresses and want to communicate with hosts on the Internet.

In fact, by implementing the firewall we have already circumvented part of the problem. Clients that communicate with the Internet by using a proxy or SOCKS server do not expose their addresses to the internet, so their addresses do not have to be translated anyway. However, when we do not want, for whatever reason, to use a proxy or SOCKS server or when proxy and SOCKS are not possible we can use NAT. For example, proxy and SOCKS servers implemented on IBM Firewall 3.1 cannot be used for UDP connections.

10.1 Translation Mechanism

Here we describe how the address translation is done. For each outgoing IP packet the source address is checked by the NAT configuration rules. If a rule matches the source address, the address is translated to an official address from the address pool. The predefined address pool contains the addresses that NAT may use for translation. For each incoming packet the destination address is checked if it is used by NAT. When this is true the address is translated to the original unofficial address. Figure 181 on page 206 shows the NAT configuration.

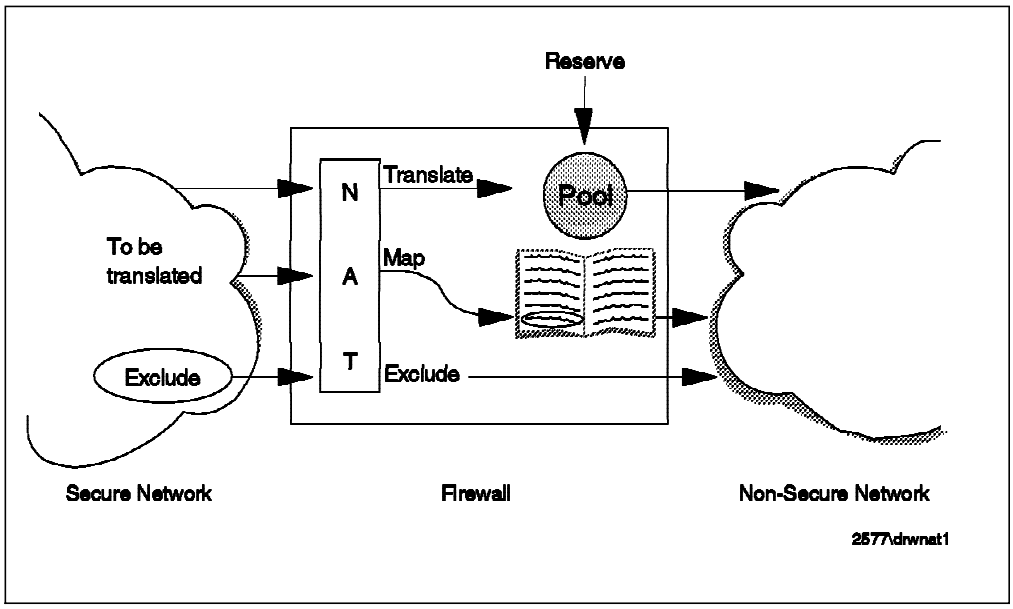


Figure 181. NAT Configuration

If NAT translates an address for a TCP packet the checksum is also adjusted. For FTP packets the task is even more difficult because the packets can contain addresses in the data of the packet. For example the FTP PORT command contains an IP address in ASCII. These addresses are also translated correctly and checksum updates and even TCP sequence and acknowledge updates are made.

Note that only TCP and UDP packets are translated by NAT. The ICMP protocol is not supported by NAT. For example pinging to the NAT addresses does not work, because ping uses the ICMP protocol.

Basically you will create filter rules that will normally route packets from a secure network to the Internet and back. NAT will take care of the address translation of the secure addresses. Figure 182 on page 207 shows how packets flow through the IBM Firewall 3.1. You will notice that NAT translation will occur for the outgoing packet after the packet has gone through both packet filters (secure and non-secure). This means that you should never mention NAT addresses in the filter rules.

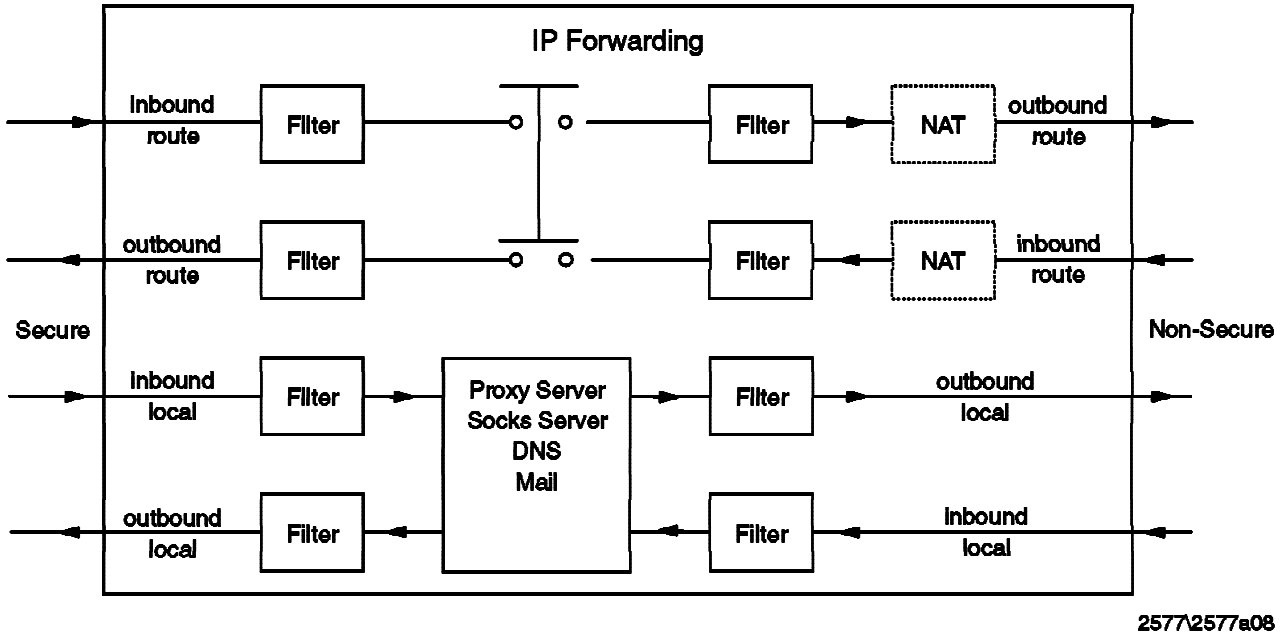


Figure 182. The Flow of the Packets inside the IBM Firewall 3.1

10.2 Configuration of NAT

The address translation is done according to NAT rules and every NAT rule starts with a keyword. The NAT keywords are visualized in Figure 183 on page 208.

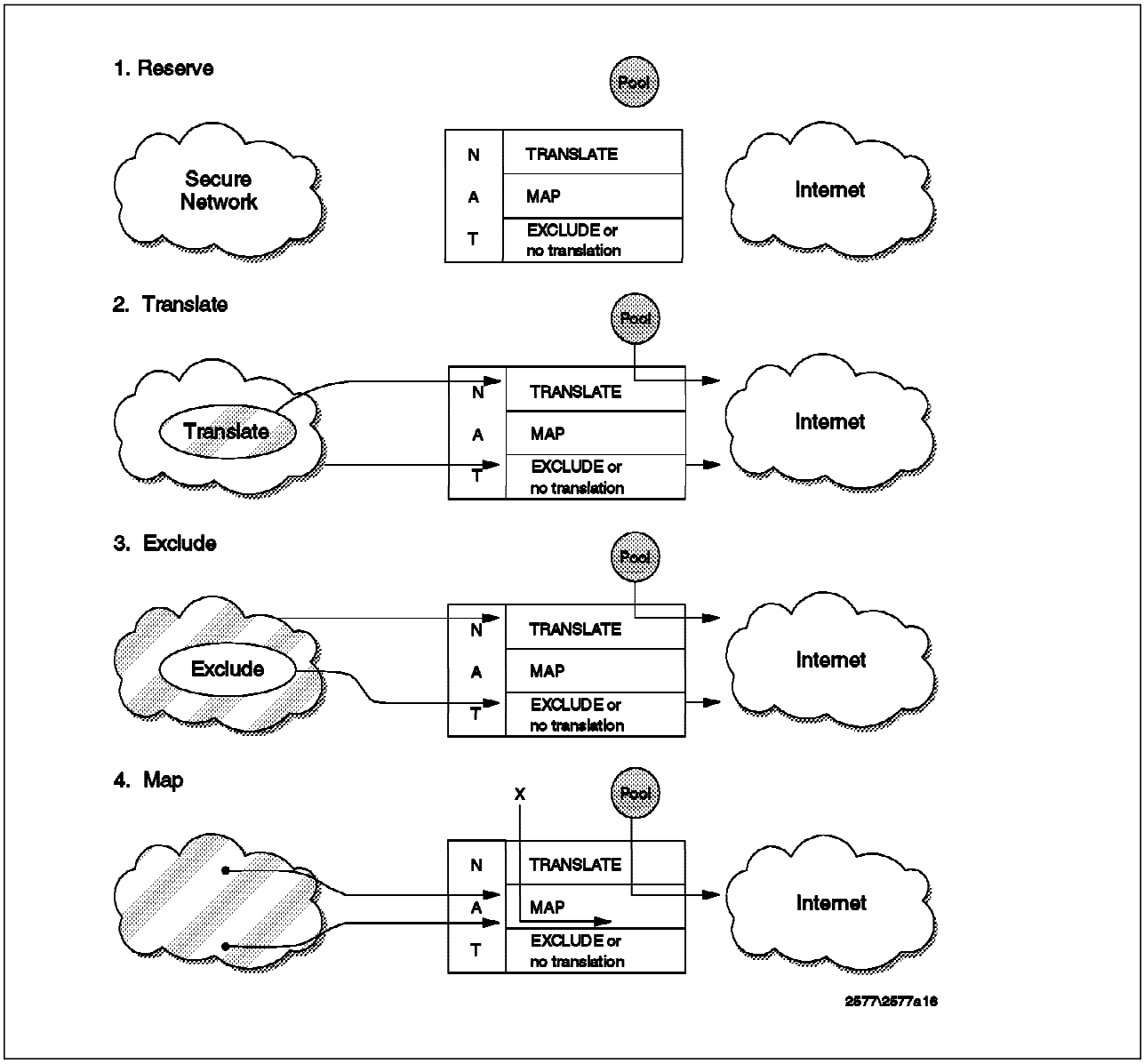


Figure 183. Network Address Translation Keywords

To set up and activate the rules for the NAT follow these rules:

1. Define the reserved address range.

   First NAT has to know which addresses it may use for the translation. Each entry consists of a network address, netmask and a timeout value. The timeout value is the number of minutes before NAT returns an unused address back to the address pool. The default value is 15 minutes and this is also the minimum value allowed. The rule starting with RESERVE defines the pool of official addresses that NAT may use.

   For example, the rule `RESERVE a.b.c.0 255.255.255.0 15` reserves the class C range a.b.c.0 to be used for NAT.

2. Define addresses to be translated.

   By default all addresses in the secure network are translated by NAT. However, you may also specify one or more ranges of addresses which must be translated.

For example, if you want the network s.s.s.0 to be translated, you define the following rule: `TRANSLATE s.s.s.0 255.255.255.0`

3. Define addresses to be excluded from translation.

   Use this step if you don't want to translate a range of addresses or you even want to exclude some addresses from the range of addresses you specified for translation.

   If you don't want to translate the network x.x.x.0, you define the following rule: `EXCLUDE x.x.x.0 255.255.255.0`

4. Define address mappings.

   An address mapping allows you to map a secure address to a specific official address. The official address may, but does not have to, be in the reserved address range. For example, if you want to give Internet users access to your internal Web site from the Internet, this is the only way to do that using NAT.

   For example, if you want the secure address a.b.c.d to be translated into w.x.y.z then define the rule: `MAP a.b.c.d w.x.y.z`

5. Activate or update the NAT configuration.

   After the initial configuration and after every change you have to activate/update the NAT configuration. You may also decide whether or not to activate the NAT logging.

The use of NAT is transparent to the filter rules. This implies that in the filter rules you have to use your untranslated secure addresses. The reserved addresses must *not* be used in the filter rules.

10.3 Configuring NAT Return Packets

The return of the NAT packets from the Internet to the IBM Firewall 3.1 must be enabled via static routing or the Address Resolution Protocol (ARP). You can use ARP entries in the firewall if you have a small pool of reserved addresses. But you must use ARP entries in the firewall if your reserved addresses are all in range of non-secure LAN. (They are within the address and mask of the non-secure adapter.) Otherwise you should use static routing.

10.3.1 How to Enable NAT Using Routing

If you have acquired separate subnet(s) of registered and routable addresses, you must go to every host on the non-secure ring or segment except the firewall, and add a static route that gives the firewall non-secure address as gateway for the NAT subnets as destinations. The non-secure router in particular needs this, so it can advertise the route back through the Internet to the backbone routers.

10.3.2 How to Enable NAT Using ARP

Determine the MAC address of the non-secure adapter with the NAT addresses within its subnet. Do the following to see the MAC addresses and types of adapters:

```
netstat -v | egrep 'STATISTICS|Hardware Address:'
```

Do the following to see the network addresses:

```
netstat -in
```

Create an arp command as follows:

- `arp -s ether HostIPAddr MAC_of_nonsecure_adapter pub`
- `arp -s 802.3 HostIPAddr MAC_of_nonsecure_adapter pub`
- `arp -s fddi HostIPAddr MAC_of_nonsecure_adapter pub`
- `arp -s 802.5 HostIPAddr MAC_of_nonsecure_adapter pub`

using the ether line for Ethernet, the 802.5 line for token-ring, etc.

HostIPAddr is the address you want to move from LAN to the NAT Map or Reserve setup. It must not be in use. Ping it if you are in doubt. It must not be the lowest or the highest in the subnet; they are reserved.

The arp entries will disappear in every reboot, so it is good to add them to the end of /etc/rc.net. This is an example of a token ring entry:

```
arp -s 802.5 1.2.3.66 00:04:ac:b6:0c:6d pub
```

You can see the ARP table by typing: `arp -a`. The ones added say "permanent published".

The following script adds pub ARP entries for the class C network a.b.c.0. Assume that the non-secure interface has a token-ring adapter with address 10.0.5a.c9.3f.63.

```
#!/bin/ksh
i=1
while [ $i -lt 255 ]
do
        arp -s 802.5 a.b.c.$i 10:00:5a:c9:3f:63 pub
        i=`expr $i + 1`
done
```

10.3.3 Maximum Transmission Unit

The Maximum Transmission Unit (MTU) for a host is the maximum datagram size it can handle. When one host sends data to another host it is preferable that the datagrams have the largest size that does not require fragmentation anywhere along the path from source to the destination. This datagram size is referred to as the Path MTU (PMTU).

At the startup of a TCP connection only the MTU values between endpoints are usually considered. So, it is possible that a packet arrives at an intermediate host that has a smaller MTU. This is solved by fragmenting or sending ICMP type 3 code 4 packets when fragmentation is not allowed.

However, NAT does not support ICMP so when using NAT the firewall must have the minimum MTU on the path. Also, all hosts and routers directly connected to the non-secure interface must have this MTU.

The default MTU for the IBM Firewall is 1492. This is the MTU of IEEE 802.3, which is most widely used. Most other LAN media are configured for a larger MTU so they can pass 1492-bytes packets without problems. To still have reasonable throughput with this MTU size, it is recommended to set the default maximum segment size on the firewall to 1440. That is 1492 minus 52, which is the maximum TCP header size. This is can be done with:

```
no -o tcp_mssdflt=1440
```

10.4 Timeout Value

The default timeout value for the NAT connections is 15 minutes. For example, if your telnet NAT connection is idle more than 15 minutes, your NAT address will be released back to the reserved address pool. Effectively it means that your telnet connection is lost. The NAT protocol will send connection reset packets for FTP connections only. All the other TCP connections will not get any reset packets. Fortunately the NAT will log these address releases to the syslog. The log message number is ICA9036. See Appendix A, "Messages" of *IBM Firewall For AIX Reference Version 3.1.1*, SC31-8418-00.

The timeout value is also used to keep your NAT address reserved after you have normally terminated your connection. You can use the same NAT address if you use NAT again within the timeout period. This has more importance with UDP protocol, because it is a connectionless protocol.

You may increase the timeout value if you expect the clients to have longer idle times than 15 minutes. Bear in mind that even when the connection is closed, the address will be reserved for an amount of time equivalent to the timeout value. If you run out of addresses in the reserved address pool, you will get error message ICA9035.

10.5 Example Configuration for NAT Telnet to Internet

Assume we have a secure network which mainly uses official C class 9.24.104.0 addresses. Part of our secure network is using private C class 192.168.36.0 addresses. We have two addresses 9.24.105.60, 9.24.105.61 in the reserved address pool that we want to use for the 192.168.36.0 addresses. Also we want the secure private address 192.168.36.30 to be mapped into 9.24.105.62. The address of the non-secure adapter is 9.24.105.250 and the mask is 255.255.255.0. The MAC address for the non-secure adapter is

```
08:00:5a:fc:3d:ca
```

Figure 184 on page 212 shows the network configuration and IP addresses.

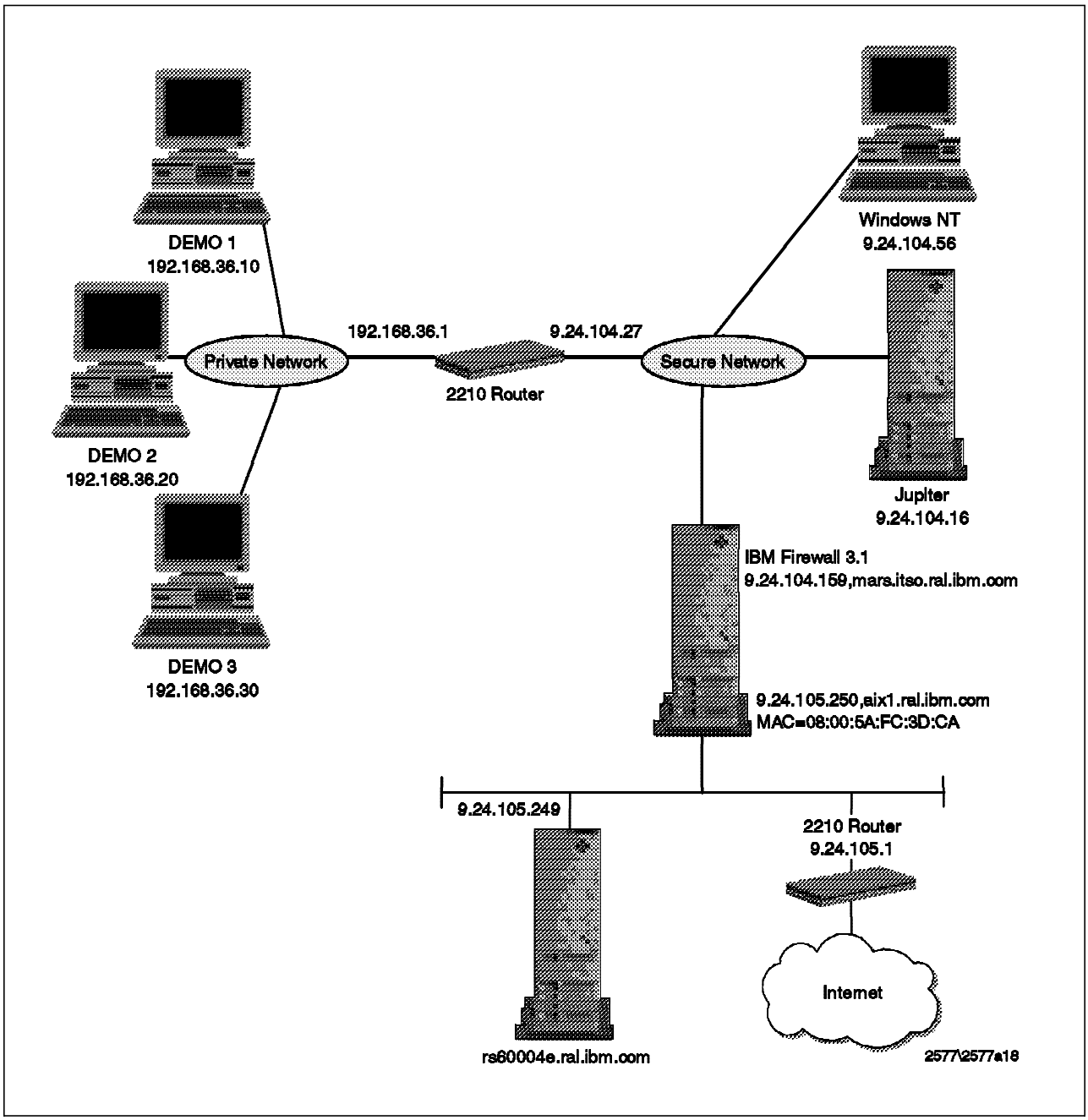


Figure 184. Example Configuration

The following NAT rules are needed for this configuration:

```
RESERVE 9.24.105.60 255.255.255.255 15
RESERVE 9.24.105.61 255.255.255.255 15
TRANSLATE 192.168.36.0 255.255.255.0
MAP 192.168.36.30 9.24.105.62
```

Note that we don't have to exclude the 9.24.104.0 addresses because only the addresses defined by TRANSLATE and MAP are translated.

Our internal routing will route the packets from the 192.168.36.0 network to the firewall. In the firewall we will have a static route for return packets to the 192.168.36.0 network.

```
192.168.36       9.24.104.27       UG        3     1033  tr0
```

The return of NAT packets from the Internet to the IBM Firewall 3.1 must be enabled. We have only two addresses in our reserved address pool and one mapped address. They are within the address and mask of the non-secure adapter, so we have to use the ARP protocol for the NAT return packets. We will have the following ARP entries at the end of our /etc/rc.net file:

```
arp -s ether 9.24.105.60 08:00:5a:fc:3d:ca pub
arp -s ether 9.24.105.61 08:00:5a:fc:3d:ca pub
arp -s ether 9.24.105.62 08:00:5a:fc:3d:ca pub
```

Our filter rules for this connection are the following:

```
permit 192.168.36.0 255.255.255.0 0 0 tcp gt 1023 eq 23 secure route inbound l=n f=y
permit 192.168.36.0 255.255.255.0 0 0 tcp gt 1023 eq 23 non-secure route outbound l=n f=y
permit 0 0 192.168.36.0 255.255.255.0 tcp/ack eq 23 gt 1023 non-secure route inbound l=n f=y
permit 0 0 192.168.36.0 255.255.255.0 tcp/ack eq 23 gt 1023 secure route outbound l=n f=y
```

Here is a small sample of iptrace taken from the firewall's non-secure adapter. It shows the handshaking between client and server. Notice that NAT has changed the 192.168.36.10 address to 9.24.105.60.

```
====( 60 bytes transmitted on interface en0 )==== 11:57:59.126834684
ETHERNET packet : [ 08:00:5a:fc:3d:ca -> 08:00:5a:4d:0f:5c ] type 800(IP)
IP header breakdown:
        < SRC =     9.24.105.60 >
        < DST =    9.24.105.249 >  (rs60004e.ral.ibm.com)
        ip_v=4, ip_hl=20, ip_tos=0, ip_len=44, ip_id=64000, ip_off=0DF
        ip_ttl=126, ip_sum=1d66, ip_p = 6 (TCP)
TCP header breakdown:
        <source port=1043, destination port=23(telnet) >
        th_seq=86ffa, th_ack=0
        th_off=6, flags<SYN>
       th_win=8192, th_sum=149f, th_urp=0
0000000     02040faa                                ]....           ]

===( 60 bytes received on interface en0 )==== 11:57:59.128700209
THERNET packet : [ 08:00:5a:4d:0f:5c -> 08:00:5a:fc:3d:ca ]  type 800
P header breakdown:
       < SRC =    9.24.105.249 >  (rs60004e.ral.ibm.com)
       < DST =     9.24.105.60 >
       ip_v=4, ip_hl=20, ip_tos=0, ip_len=44, ip_id=53139, ip_off=0
       ip_ttl=60, ip_sum=c9d3, ip_p = 6 (TCP)
CP header breakdown:
       <source port=23(telnet), destination port=1043 >
       th_seq=6dd97201, th_ack=86ffb
        th_off=6, flags<SYN ] ACK>
        th_win=16060, th_sum=1fed, th_urp=0
00000000     020405b4                                ]....           ]
====( 60 bytes transmitted on interface en0 )==== 11:57:59.131802457
ETHERNET packet : [ 08:00:5a:fc:3d:ca -> 08:00:5a:4d:0f:5c type 800 (IP)
IP header breakdown:
        < SRC =     9.24.105.60 >
        < DST =    9.24.105.249 >  (rs60004e.ral.ibm.com)
        ip_v=4, ip_hl=20, ip_tos=0, ip_len=40, ip_id=64256, ip_off=0DF
        ip_ttl=126, ip_sum=1c6a, ip_p = 6 (TCP)
TCP header breakdown:
        <source port=1043, destination port=23(telnet) >
        th_seq=86ffb, th_ack=6dd97202
        th_off=5, flags<ACK>
        th_win=8760, th_sum=542e, th_urp=0
```

Chapter 11. Domain Name Service

Before you set up DNS on the firewall, you should understand how it works. We described the operation of the DNS relay on the firewall in 2.1.4, "Domain Name Service" on page 17. We recommend reading that section now so that you have a clear picture of what you are trying to achieve with DNS configuration.

In summary, the objectives of the DNS relay function are:

1. Provide access to nonsecure network domain name/address mappings for users in the secure network
2. Hide the secure network names and addresses from users outside the secure network
3. Provide name/address mapping for resources that you *want* to reveal (usually servers and gateways)

In general, the standard IBM Firewall 3.1 configuration process will set up the name server successfully for the firewall itself. However, for a working system you will also need to configure DNS inside the secure network and, possibly, in the DMZ.

11.1 Configuring Domain Name Server on the Firewall

First we will describe the standard configuration for domain name service on the firewall using GUI. Then we will discuss some of the ways in which you may want to extend that configuration.

Initial configuring is simple, you just have to select the Domain Name Services option inside System Administration. Figure 185 shows the GUI screen. After selecting **OK**, the name server configuration files are created and the named daemon is started automatically.

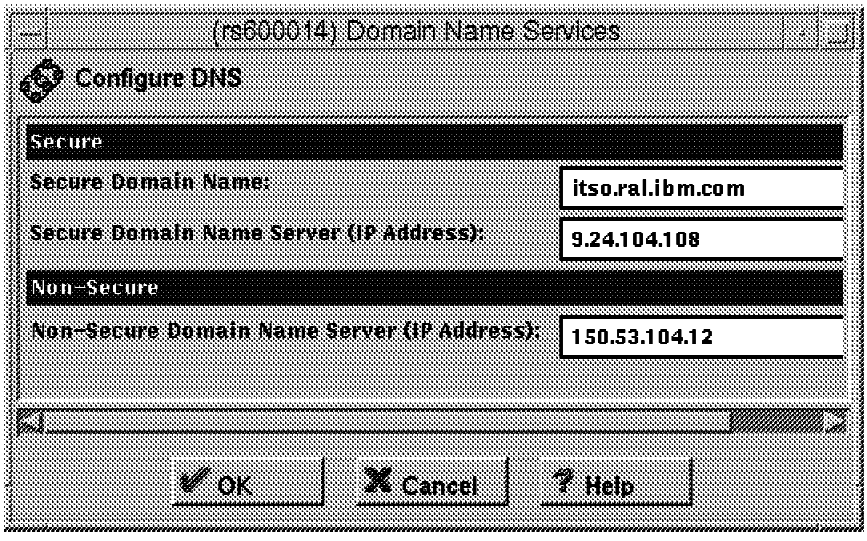


Figure 185. Add Network Services Configuration Entry

The following two name servers have to be defined:

- The secure domain name server is where name requests originating on the firewall itself will go to be resolved. This address will be placed in the file /etc/resolv.conf.
- The nonsecure name server is where the firewall name server goes to resolve all unknown names. If the DNS configuration is properly designed, this will be the name server for the next higher layer in the naming hierarchy. Often it will be an Internet root name server, or a server maintained by the provider of your network connection. Alternatively, if you have a DMZ configuration and maintain a separate name server in the DMZ, then that will be the external name server from the firewall's point of view. Note that this does not strictly follow the rules of the DNS hierarchy, but it is not unusual, so we will show an example of this configuration in 11.3, “DNS Configuration Examples” on page 217.

You also have to provide the domain name of the secure side of the firewall. How do you know what name to use? You may not be able to freely choose the domain name. Assuming that you are connecting an existing IP network to the Internet, the secure network domain name will probably already exist. You don't have to specify the nonsecure domain name here, but you need to consider the following: the nonsecure name must be authorized by the Internet Assigned Numbers Authority and will follow the national and international conventions for IP domain names. This is the name that you will be known by to the rest of the world. If your firewall configuration includes a DMZ, the servers within the DMZ will be in this domain.

The best practice is to strictly follow the hierarchical domain naming standards, which is the way that DNS was designed to work. DNS will work even if you use a non-hierarchical scheme, but we do not recommend it. One thing you should strive for is to have internal and external domain names that are easily differentiated from each other. This means that you can instantly identify whether a resource is inside or outside the firewall, and it makes it much easier to create DNS configurations and mail routing rules.

11.2 Configuring the Internal Network Domain Name Server

The internal network name server also has to be configured to use the firewall to resolve nonsecure names. This involves the following definition:

- A forwarders entry to point to the firewall

A forwarders entry looks like this:

```
forwarders 9.24.104.27 9.24.104.27 9.24.104.27
```

The Internal Network Domain Name Server will query each forwarder only once and then wait for a short period of time. Using the same forwarder entry more than once extends the time that the Internal Network Domain Name Server will wait for the answer.

11.3 DNS Configuration Examples

The best way to understand the possible DNS configurations is to look at some examples. We will consider the following two cases:

1. With a name server located in the DMZ providing resolution for externally visible names
2. A similar configuration, except with the external resolution performed by the IBM Firewall 3.1 machine

11.3.1 Case 1: Internal DNS, Merge DNS, and External DNS

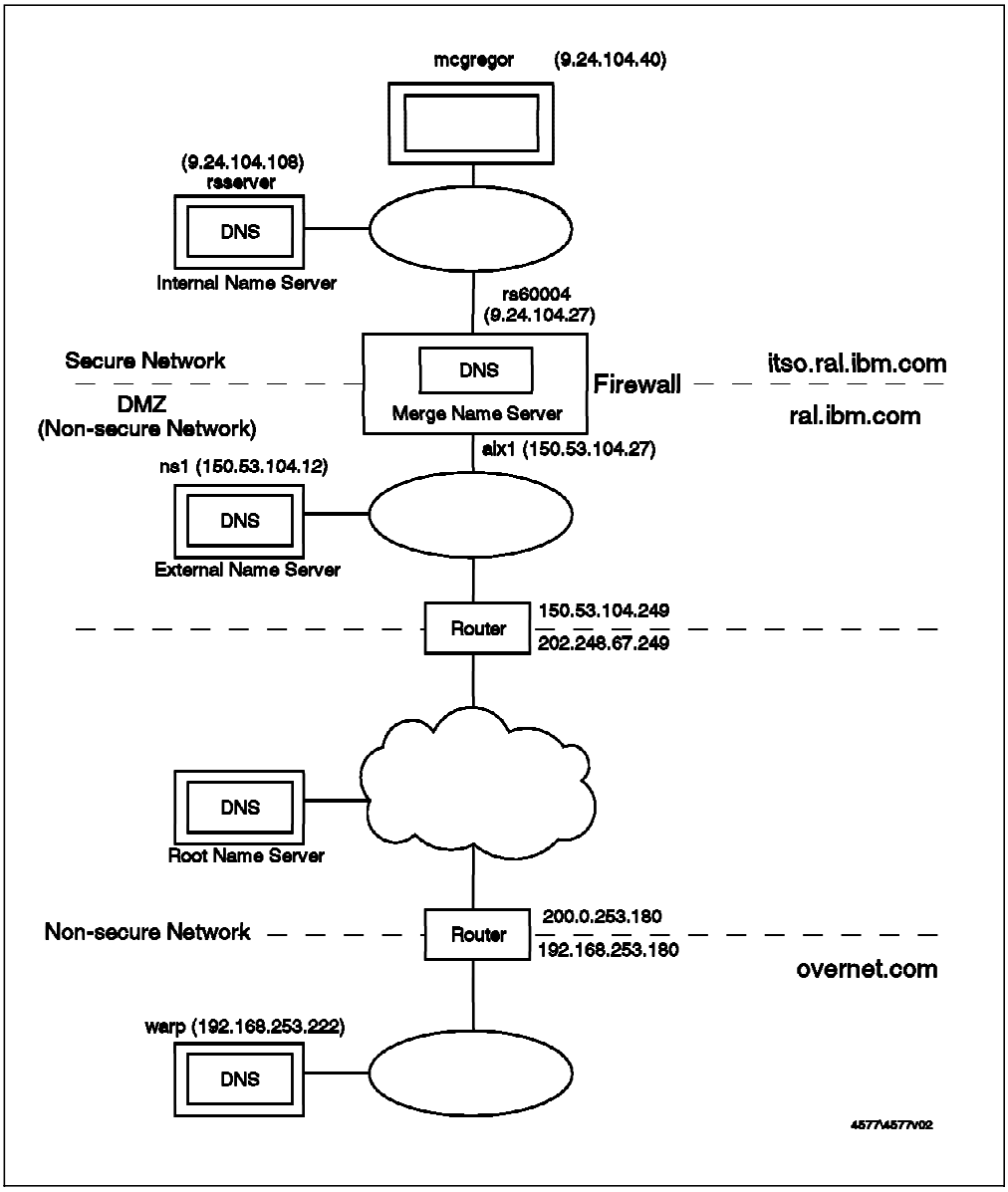


Figure 186. Case 1, DNS Configuration Example

11.3.1.1 Configuration of the Merge DNS on the Firewall

The following files are created by IBM Firewall 3.1 when you add a Network Services Configuration Entry for DNS, as shown in Figure 185 on page 215.

- /etc/resolv.conf
- /etc/fwnamed.boot
- /etc/fwnamed.loc
- /etc/fwnamed.ca

The /etc/resolv.conf file points to the internal name server, which is where name requests originating on the firewall are resolved. In our case this is address 9.24.104.108, so /etc/resolv.conf will contain the following:

```
domain itso.ral.ibm.com
nameserver 9.24.104.108
```

Figure 187. /etc/resolv.conf File on the Firewall

Note that this means that when the firewall machine tries to resolve an IP name, it behaves exactly like a host in the secure network.

The /etc/fwnamed.boot file is the base file from which the DNS configuration is defined. In this case it just specifies the root server hints file /etc/fwnamed.ca, and the loopback/localhost reverse address file.

```
; Created by IBM Firewall 1997080180
cache      .                    /etc/fwnamed.ca
primary    0.0.127.in-addr.arpa /etc/fwnamed.loc
```

Figure 188. Case 1, /etc/fwnamed.boot File on the Firewall

The /etc/fwnamed.loc file just contains the mapping for the loopback/local host address (127.0.0.1).

```
; Created by IBM Firewall 1997080180
@  IN SOA rs60004.itso.ral.ibm.com. root.rs60004.itso.ral.ibm.com.
        (1997085114 3600 600 3600000 86400 )
 IN NS  rs60004.itso.ral.ibm.com.
1 IN PTR localhost.
```

Figure 189. Case 1, /etc/fwnamed.loc File on the Firewall

/etc/fwnamed.ca, the cache hints file, specifies the name server(s) used to request the list of root name servers; in this case the external name server. The Merge DNS on the firewall will ask that name server for the current list of root name servers, and will cache it in memory. It will repeat this process when the cached list time-to-live expires. In our case this is the DNS system inside the DMZ, 150.53.104.12.

Take care if you manually stop and start the firewall DNS server: *stop src -s named* will stop the service, but to restart you must specify the configuration file, *startsrc -s named -a "-b /etc/fwnamed.boot"*, otherwise it will attemp to use the default file /etc/named.boot.

```
; Created by IBM Firewall 1997080180
. IN NS  externaldns.150.53.104.12.
externaldns.150.53.104.12.  3600000 IN A 150.53.104.12
```

Figure 190. Case 1, /etc/fwnamed.ca File on the Firewall

11.3.1.2 Configuration of DNS for the External Name Server

The external name server in this case is actually between the two parts of the firewall in our DMZ. It is a very simple DNS configuration, providing name mappings for DMZ resources. These names must be resolvable from anywhere in the Internet.

Unlike the firewall system, name resolution requests on this machine go to the name server on the same system, so /etc/resolv.conf is conventional.

```
domain ral.ibm.com
nameserver 150.53.104.12
```

Figure 191. Case 1, /etc/resolv.conf File on the External DNS (DMZ)

The DNS name space for the DMZ is called *ral.ibm.com*, so named.boot defines us as the primary server for that domain.

```
directory       /etc
domain          ral.ibm.com
primary         ral.ibm.com             named.zone
primary         104.53.150.in-addr.arpa named.rev
primary         0.0.127.in-addr.arpa    named.loc
cache           .                       named.ca
```

Figure 192. Case 1, /etc/named.boot File on the External DNS (DMZ)

The named.zone file defines all the servers and aliases that are hosted in the DMZ. It also contains the MX record that allows a remote mail server to look up the address where the mail is to be sent. In our case, we want the external mail address for a user inside our secure network to be user@ral.ibm.com, instead of user@host.itso.ral.ibm.com. Therefore the MX record directs this mail ID to the nonsecure interface of the IBM Firewall 3.1 system. In addition to MX record there is an address entry for domain ral.ibm.com. This is for the mailers that do not use MX records.

```
@               IN      SOA     ns1.ral.ibm.com. root.ns1.ral.ibm.com.
                                (
                                1997020701    ;Serial     (yyyymmddvv)
                                3600          ;Refresh    (1 hour)
                                300           ;Retry      (5 min)
                                3600000       ;Expire     (41 days)
                                3600          ;Minimum TTL (1 hour)
                                )
                IN      NS              ns1.ral.ibm.com.
ral.ibm.com.    IN      MX      10      aix1.ral.ibm.com.
ral.ibm.com.    IN      A               150.53.104.27
; For mailers that do not read MX

localhost       IN      A               127.0.0.1
ns1             IN      A               150.53.104.12
aix1            IN      A               150.53.104.27
www             IN      A               150.53.104.31
router          IN      A               150.53.104.249
```

Figure 193. Case 1, /etc/named.zone File on the External DNS (DMZ)

If you want to receive mail with a destination address of user@host.itso.ral.ibm.com, you would add an MX record such as:

```
*.itso.ral.ibm.com.   IN    MX   10     aix1.ral.ibm.com.
```

The reverse name resolution definition file, named.rev, is unremarkable.

```
@               IN      SOA     ns1.ral.ibm.com. root.ns1.ral.ibm.com.
                                (
                                1997020701    ;Serial     (yyyymmddvv)
                                3600          ;Refresh    (1 hour)
                                300           ;Retry      (5 min)
                                3600000       ;Expire     (41 days)
                                3600          ;Minimum TTL (1 hour)
                                )

                IN      NS                    ns1.ral.ibm.com.
12              IN      PTR                   ns1.ral.ibm.com.
27              IN      PTR                   aix1.ral.ibm.com.
31              IN      PTR                   www.ral.ibm.com.
249             IN      PTR                   router.ral.ibm.com.
```

Figure 194. Case 1, /etc/named.rev File on the External DNS (DMZ)

The /etc/named.loc file just contains the mapping for the loopback/local host address (127.0.0.1).

```
@               IN      SOA     ns1.ral.ibm.com. root.ns1.ral.ibm.com.
                                (
                                1997020701    ;Serial     (yyyymmddvv)
                                3600          ;Refresh    (1 hour)
                                300           ;Retry      (5 min)
                                3600000       ;Expire     (41 days)
                                3600          ;Minimum TTL (1 hour)
                                )
                IN      NS                    ns1.ral.ibm.com.
1               IN      PTR                   localhost.
```

Figure 195. Case 1, /etc/named.loc File on the External DNS (DMZ)

The named.ca file contains information on the name servers which are asked for the current list of root name servers. This allows DNS to work even if the named.ca file contains out-of-date information, providing it contains at least one working address. You should periodically obtain the latest version of this file.

```
;       This file holds the information on root name servers needed to
;       initialize cache of Internet domain name servers
;       (e.g. reference this file in the "cache  .  <file>"
;       configuration file of BIND domain name servers).
;       This file is made available by InterNIC registration services
;       under anonymous FTP as
;           file                /domain/named.root
;           on server           FTP.RS.INTERNIC.NET
;       -OR- under Gopher at    RS.INTERNIC.NET
;           under menu          InterNIC Registration Services (NSI)
;              submenu          InterNIC Registration Archives
;           file                named.root
;
;       last update:    Aug 22, 1997
;       related version of root zone:   1997082200
;
;       This file is made available by DOD NIC registration services
;       under anonymous FTP as
;           file                /domain/named.root
;           on server           NIC.DDN.MIL
;
;       last update:    Nov 8, 1995
;       related version of root zone:   199511080
;
; formerly NS.INTERNIC.NET
;
.                       3600000  IN  NS    A.ROOT-SERVERS.NET.
A.ROOT-SERVERS.NET.     3600000      A     198.41.0.4
;
; formerly NS1.ISI.EDU
;
.                       3600000      NS    B.ROOT-SERVERS.NET.
B.ROOT-SERVERS.NET.     3600000      A     128.9.0.107
;
; formerly C.PSI.NET
;
.                       3600000      NS    C.ROOT-SERVERS.NET.
C.ROOT-SERVERS.NET.     3600000      A     192.33.4.12
;
; formerly TERP.UMD.EDU
;
.                       3600000      NS    D.ROOT-SERVERS.NET.
D.ROOT-SERVERS.NET.     3600000      A     128.8.10.90
;
; formerly NS.NASA.GOV
;
.                       3600000      NS    E.ROOT-SERVERS.NET.
E.ROOT-SERVERS.NET.     3600000      A     192.203.230.10
;
; formerly NS.ISC.ORG
;
.                       3600000      NS    F.ROOT-SERVERS.NET.
F.ROOT-SERVERS.NET.     3600000      A     192.5.5.241
;
; formerly NS.NIC.DDN.MIL
;
.                       3600000      NS    G.ROOT-SERVERS.NET.
G.ROOT-SERVERS.NET.     3600000      A     192.112.36.4
;
; formerly AOS.ARL.ARMY.MIL
;
.                       3600000      NS    H.ROOT-SERVERS.NET.
H.ROOT-SERVERS.NET.     3600000      A     128.63.2.53
;
; formerly NIC.NORDU.NET
;
.                       3600000      NS    I.ROOT-SERVERS.NET.
I.ROOT-SERVERS.NET.     3600000      A     192.36.148.17
;
```

Figure 196. Case 1, /etc/named.ca File on the External DNS (DMZ) Part 1 of 2

```
; temporarily housed at NSI (InterNIC)
;
.                       3600000      NS    J.ROOT-SERVERS.NET.
J.ROOT-SERVERS.NET.     3600000      A     198.41.0.10
;
; housed in LINX, operated by RIPE NCC
;
.                       3600000      NS    K.ROOT-SERVERS.NET.
K.ROOT-SERVERS.NET.     3600000      A     193.0.14.129
;
; temporarily housed at ISI (IANA)
;
.                       3600000      NS    L.ROOT-SERVERS.NET.
L.ROOT-SERVERS.NET.     3600000      A     198.32.64.12
;
; housed in Japan, operated by WIDE
;
.                       3600000      NS    M.ROOT-SERVERS.NET.
M.ROOT-SERVERS.NET.     3600000      A     202.12.27.33
; End of File
```

Figure 197. Case 1, /etc/named.ca File on the External DNS (DMZ) Part 2 of 2

11.3.1.3 Configuration of the Internal DNS

The configuration of the internal DNS is standard. As in the case of the DMZ name server, resolution of names from the server itself is conventional.

```
domain itso.ral.ibm.com
name server 9.24.104.108
```

Figure 198. Case 1, /etc/resolv.conf File on the Internal DNS

The named.boot file is also conventional, except that it contains a *forwarders* record pointing to the firewall. This means that any requests for addresses outside its own domain will be passed to DNS on the IBM Firewall 3.1 system. When we defined that DNS, we specified that outside addresses should be cached (see Figure 199), so we will only have to send a request to the root name servers the first time an address needs to be resolved.

```
directory       /etc
forwarders      9.24.104.27 9.24.104.27 9.24.104.27
domain          itso.ral.ibm.com
cache           .                       named.ca
primary         itso.ral.ibm.com        named.zone
primary         104.24.9.in-addr.arpa   named.rev
primary         0.0.127.in-addr.arpa    named.local
slave
```

Figure 199. Case 1, /etc/named.boot File on the Internal DNS

The *slave* directive specifies that the nameserver completely relies on the forwarders servers. Without the slave directive, the internal DNS server will try to directly contact the root name servers, if the request from the *forwarders* servers times out for any reason. These direct requests will always fail, as the firewall filter rules will block them, and in the meantime the internal DNS will appear to "hang". You may use the directive *options forwarders-only* instead of *slave*.

The named.ca file specifies the nameserver used to request the list of root name servers. It must contain the secure interface address of the firewall itself, as only the nameserver running on the firewall has direct access to the root name

servers and the external DNS, as the filter rules block any other DNS attempts through the firewall.

The remaining configuration files for the internal DNS (named.zone, named.rev and named.ca) are conventional, but we have listed them here for completeness.

```
@       IN      SOA         rsserver.itso.ral.ibm.com. dlboone.vnet.ibm.com.
                (
                1996010506  ; Serial number (yyyymmddvv)
                1800        ; Secondary checks primary every 30 min. for refresh
                900         ; Retry interval if connect 2 primary fails -> 15 min.
                172800      ; Secondary expires after 48 hours if still no primary
                1800        ; time-to-live(other servers)-> 30 min.
                )
                IN          NS          rsserver.itso.ral.ibm.com.

rs60004         IN          A           9.24.104.27     ; SHOGREN
mcgregor        IN          A           9.24.104.40     ; MCGREGOR
rsserver        IN          A           9.24.104.108    ; MCGREGOR
rs600020        IN          A           9.24.104.241    ; BARRY
```

Figure 200. Case 1, /etc/named.zone File on the Internal DNS

```
@       IN      SOA         rsserver.itso.ral.ibm.com. dlboone.vnet.ibm.com.
                (
                1996010506  ; Serial number (yyyymmddvv)
                1800        ; Secondary checks primary every 30 min. for refresh
                900         ; Retry interval if connect 2 primary fails -> 15 min.
                172800      ; Secondary expires after 48 hours if still no primary
                1800        ; time-to-live(other servers)-> 30 min.
                )

                IN          NS          rsserver.itso.ral.ibm.com.

27              IN          PTR         rs60004.itso.ral.ibm.com.
40              IN          PTR         mcgregor.itso.ral.ibm.com.
108             IN          PTR         rsserver.itso.ral.ibm.com.
241             IN          PTR         rs600020.itso.ral.ibm.com.
```

Figure 201. Case 1, /etc/named.rev File on the Internal DNS

```
.               IN          NS           rs60004.itso.ral.ibm.com.
rs60004.itso.ral.ibm.com.       IN A   9.24.104.27
```

Figure 202. Case 1, /etc/named.ca File on the Internal DNS

```
@       IN      SOA         rsserver.itso.ral.ibm.com. dlboone.vnet.ibm.com.
                (
                1996010506  ; Serial number (yyyymmddvv)
                1800        ; Secondary checks primary every 30 min. for refresh
                900         ; Retry interval if connect 2 primary fails -> 15 min.
                172800      ; Secondary expires after 48 hours if still no primary
                1800        ; time-to-live(other servers)-> 30 min.
                )

                IN          NS          rsserver.itso.ral.ibm.com.
1               IN          PTR         localhost.
```

Figure 203. Case 1, /etc/named.local File on the Internal DNS

11.3.1.4 Name Resolution from Secure Network

Figure 204 shows a sequence of nslookup requests from a host in the secure network. As you can see, we can resolve addresses wherever they exist, whether they are in the nonsecure network (warp.overnet.com), in the DMZ (ns1.ral.ibm.com, aix1.ral.ibm.com), or in the secure network (mcgregor.itso.ral.ibm.com, rs60004.itso.ral.ibm.com).

```
rs600020:/ > nslookup
Default Server:  rsserver.itso.ral.ibm.com
Address:  9.24.104.108

> warp.overnet.com.
Server:  rsserver.itso.ral.ibm.com
Address:  9.24.104.108

Non-authoritative answer:
Name:    warp.overnet.com
Address:  192.168.253.222

> ns1.ral.ibm.com.
Server:  rsserver.itso.ral.ibm.com
Address:  9.24.104.108

Non-authoritative answer:
Name:    ns1.ral.ibm.com
Address:  150.53.104.12

> mcgregor.itso.ral.ibm.com.
Server:  rsserver.itso.ral.ibm.com
Address:  9.24.104.108

Name:    mcgregor.itso.ral.ibm.com
Address:  9.24.104.40

> rs60004.itso.ral.ibm.com.
Server:  rsserver.itso.ral.ibm.com
Address:  9.24.104.108

Name:    rs60004.itso.ral.ibm.com
Address:  9.24.104.27

> aix1.ral.ibm.com.
Server:  rsserver.itso.ral.ibm.com
Address:  9.24.104.108

Non-authoritative answer:
Name:    aix1.ral.ibm.com
Address:  150.53.104.27
```

Figure 204. Case 1, Name Resolution from Secure Network with nslookup

11.3.1.5 Name Resolution from Nonsecure Network

As shown in Figure 205 on page 225, we cannot resolve secure network addresses (rs60004.itso.ral.ibm.com) from the nonsecure network. However, we can resolve DMZ network addresses (ns1.ral.ibm.com, aix1.ral.ibm.com).

```
C:\mptn\bin>nslookup
Default Server:  warp.overnet.com
Address:  192.168.253.222

> ns1.ral.ibm.com.
Server:  warp.overnet.com
Address:  192.168.253.222

Non-authoritative answer:
Name:    ns1.ral.ibm.com
Address:  150.53.104.12

> aix1.ral.ibm.com.
Server:  warp.overnet.com
Address:  192.168.253.222

Name:    aix1.ral.ibm.com
Address:  150.53.104.27

> rs60004.itso.ral.ibm.com.
Server:  warp.overnet.com
Address:  192.168.253.222

*** warp.overnet.com can't find rs60004.itso.ral.ibm.com.: Server failed
```

Figure 205. Case 1, Resolve Names from Nonsecure Network with nslookup

11.3.2 Case 2: External DNS on the Firewall and Internal DNS

You may not want to set up an individual domain name server within the DMZ. For example, if you only have one machine (such as a World Wide Web server) in the DMZ, or if you do not have a DMZ configuration, it would not be justifiable to buy a new box just for use as an external DNS.

The alternative is to set up DNS on the firewall to do the job.

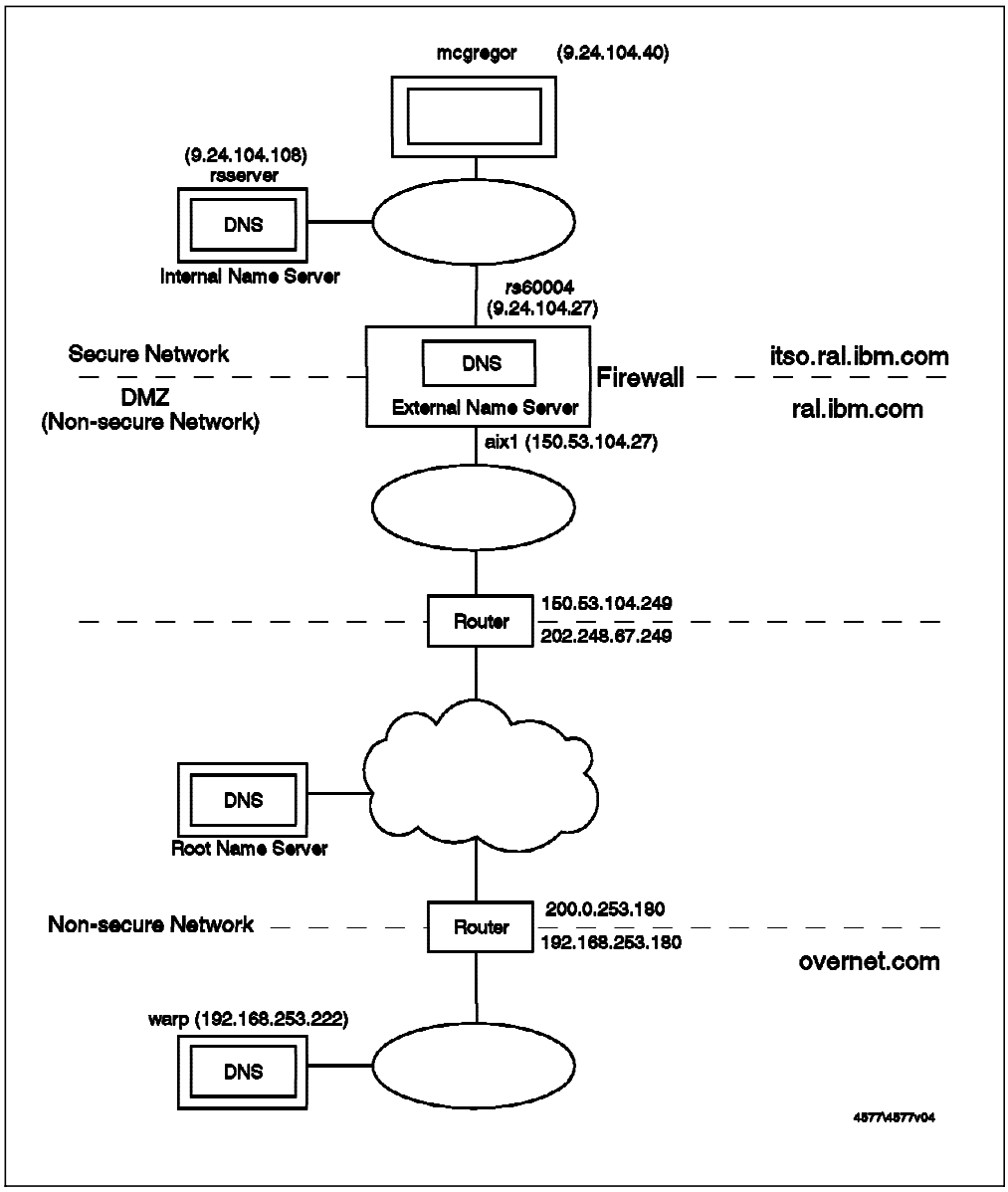


Figure 206. DNS Configuration Example, Case 2

11.3.2.1 Configuration of the External DNS on the Firewall

In this case, we need to modify the DNS configuration after adding the Network Services Configuration Entry. Names served by the external DNS on the firewall must be resolved by all sites in the Internet, in particular its own name. However, as in case 1, name requests originating on the firewall itself are routed to the secure network name server.

```
domain itso.ral.ibm.com
nameserver 9.24.104.108
```

Figure 207. Case 2, /etc/resolv.conf File on the Firewall

The named.boot, named.zone and named.ca files are exactly like the equivalents on the external DNS machine from case 1, except with the external address of the firewall (150.53.104.27, aix1.ral.ibm.com) substituted.

```
directory                       /etc
domain  ral.ibm.com
cache   .                       /etc/named.ca
primary ral.ibm.com             /etc/named.zone
primary 104.53.150.in-addr.arpa /etc/named.rev
primary 0.0.127.in-addr.arpa    /etc/named.loc
```

Figure 208. Case 2, /etc/named.boot File on the Firewall

```
@               IN      SOA     aix1.ral.ibm.com. root.aix1.ral.ibm.com.
                                (
                                1997020702    ;Serial     (yyyymmddvv)
                                3600          ;Refresh    (1 hour)
                                300           ;Retry      (5 min)
                                3600000       ;Expire     (41 days)
                                3600          ;Minimum TTL (1 hour)
                                )
                IN      NS              aix1.ral.ibm.com.
ral.ibm.com.    IN      MX      10      aix1.ral.ibm.com.
ral.ibm.com.    IN      A               150.53.104.27
; For mailers that do not read MX

localhost       IN      A               127.0.0.1
aix1            IN      A               150.53.104.27
router          IN      A               150.53.104.249
www             IN      A               150.53.104.31
```

Figure 209. Case 2, /etc/named.zone File on the Firewall

```
@               IN      SOA     aix1.ral.ibm.com. root.aix1.ral.ibm.com.
                                (
                                1997020702    ;Serial     (yyyymmddvv)
                                3600          ;Refresh    (1 hour)
                                300           ;Retry      (5 min)
                                3600000       ;Expire     (41 days)
                                3600          ;Minimum TTL (1 hour)
                                )
                IN      NS              aix1.ral.ibm.com.
27              IN      PTR             aix1.ral.ibm.com.
31              IN      PTR             www.ral.ibm.com.
249             IN      PTR             router.ral.ibm.com.
```

Figure 210. Case 2, /etc/named.rev File on the Firewall

```
; Created by IBM Firewall 1997080181
@ IN SOA aix1.ral.ibm.com. root.aix1.ral.ibm.com.
 ( 1997020702 3600 600 3600000 86400)
  IN NS aix1.ral.ibm.com.
1 IN PTR localhost.
```

Figure 211. Case 2, /etc/named.loc File on the Firewall

```
;       This file holds the information on root name servers needed to
;       initialize cache of Internet domain name servers
;       (e.g. reference this file in the "cache  .  <file>"
;       configuration file of BIND domain name servers).
;       This file is made available by InterNIC registration services
;       under anonymous FTP as
;           file                /domain/named.root
;           on server           FTP.RS.INTERNIC.NET
;       -OR- under Gopher at    RS.INTERNIC.NET
;           under menu          InterNIC Registration Services (NSI)
;              submenu          InterNIC Registration Archives
;           file                named.root
;
;       last update:    Aug 22, 1997
;       related version of root zone:   1997082200
;
;       This file is made available by DOD NIC registration services
;       under anonymous FTP as
;           file                /domain/named.root
;           on server           NIC.DDN.MIL
;
;       last update:    Nov 8, 1995
;       related version of root zone:   199511080
;
; formerly NS.INTERNIC.NET
;
.                       3600000  IN  NS    A.ROOT-SERVERS.NET.
A.ROOT-SERVERS.NET.     3600000      A     198.41.0.4
;
; formerly NS1.ISI.EDU
;
.                       3600000      NS    B.ROOT-SERVERS.NET.
B.ROOT-SERVERS.NET.     3600000      A     128.9.0.107
;
; formerly C.PSI.NET
;
.                       3600000      NS    C.ROOT-SERVERS.NET.
C.ROOT-SERVERS.NET.     3600000      A     192.33.4.12
;
; formerly TERP.UMD.EDU
;
.                       3600000      NS    D.ROOT-SERVERS.NET.
D.ROOT-SERVERS.NET.     3600000      A     128.8.10.90
;
; formerly NS.NASA.GOV
;
.                       3600000      NS    E.ROOT-SERVERS.NET.
E.ROOT-SERVERS.NET.     3600000      A     192.203.230.10
;
; formerly NS.ISC.ORG
;
.                       3600000      NS    F.ROOT-SERVERS.NET.
F.ROOT-SERVERS.NET.     3600000      A     192.5.5.241
;
; formerly NS.NIC.DDN.MIL
;
.                       3600000      NS    G.ROOT-SERVERS.NET.
G.ROOT-SERVERS.NET.     3600000      A     192.112.36.4
;
; formerly AOS.ARL.ARMY.MIL
;
.                       3600000      NS    H.ROOT-SERVERS.NET.
H.ROOT-SERVERS.NET.     3600000      A     128.63.2.53
;
; formerly NIC.NORDU.NET
;
.                       3600000      NS    I.ROOT-SERVERS.NET.
I.ROOT-SERVERS.NET.     3600000      A     192.36.148.17
;
```

Figure 212. Case 2, /etc/named.ca File on the Firewall (Part 1 of 2)

```
; temporarily housed at NSI (InterNIC)
;
.                       3600000      NS    J.ROOT-SERVERS.NET.
J.ROOT-SERVERS.NET.     3600000      A     198.41.0.10
;
; housed in LINX, operated by RIPE NCC
;
.                       3600000      NS    K.ROOT-SERVERS.NET.
K.ROOT-SERVERS.NET.     3600000      A     193.0.14.129
;
; temporarily housed at ISI (IANA)
;
.                       3600000      NS    L.ROOT-SERVERS.NET.
L.ROOT-SERVERS.NET.     3600000      A     198.32.64.12
;
; housed in Japan, operated by WIDE
;
.                       3600000      NS    M.ROOT-SERVERS.NET.
M.ROOT-SERVERS.NET.     3600000      A     202.12.27.33
; End of File
```

Figure 213. Case 2, /etc/named.ca File on the Firewall (Part 2 of 2)

11.3.2.2 Configuration of the Internal DNS

The configuration of the Internal DNS is exactly the same as in case 1 (see 11.3.1.3, “Configuration of the Internal DNS” on page 222).

11.3.2.3 Name Resolution from the Secure Network

Figure 214 shows the same sequence of nslookup requests that we tried for case 1 from a host in the secure network. As you can see, we can resolve addresses wherever they exist, whether they are in the nonsecure network (warp.overnet.com), in the DMZ (ns1.ral.ibm.com, aix1.ral.ibm.com), or in the secure network (mcgregor.itso.ral.ibm.com, rs60004.itso.ral.ibm.com).

```
rs600020:/ > nslookup
Default Server:  rsserver.itso.ral.ibm.com
Address:  9.24.104.108

> warp.overnet.com.
Server:  rsserver.itso.ral.ibm.com
Address:  9.24.104.108

Non-authoritative answer:
Name:    warp.overnet.com
Address:  192.168.253.222

> www.ral.ibm.com.
Server:  rsserver.itso.ral.ibm.com
Address:  9.24.104.108

Non-authoritative answer:
Name:    www.ral.ibm.com
Address:  150.53.104.31

> aix1.ral.ibm.com.
Server:  rsserver.itso.ral.ibm.com
Address:  9.24.104.108

Non-authoritative answer:
Name:    aix1.ral.ibm.com
Address:  150.53.104.27
```

Figure 214. Case 2, Resolve Names from Secure Network with nslookup

11.3.2.4 Name Resolution from the Nonsecure Network

From the nonsecure network, we can resolve DMZ addresses (www.ral.ibm.com, aix1.ral.ibm.com) but not secure network addresses, as shown in Figure 215.

```
C:\mptn\etc\namedb>nslookup
Default Server:  warp.overnet.com
Address:  192.168.253.222

> aix1.ral.ibm.com.
Server:  warp.overnet.com
Address:  192.168.253.222

Non-authoritative answer:
Name:    aix1.ral.ibm.com
Address:  150.53.104.27

> www.ral.ibm.com.
Server:  warp.overnet.com
Address:  192.168.253.222

Non-authoritative answer:
Name:    www.ral.ibm.com
Address:  150.53.104.31

> rs60004.itso.ral.ibm.com.
Server:  warp.overnet.com
Address:  192.168.253.222

*** warp.overnet.com can't find rs60004.itso.ral.ibm.com.: Server failed

> mcgregor.itso.ral.ibm.com.
Server:  warp.overnet.com
Address:  192.168.253.222

*** warp.overnet.com can't find mcgregor.itso.ral.ibm.com.: Server failed
```

Figure 215. Case 2, Resolve Names from Nonsecure Network with nslookup

Chapter 12. Mail Handling

IBM Firewall 3.1 provides a secure SMTP mail relay capability, as we briefly described in 2.1.5, “Secure Mail Handling” on page 19. This allows you to present a single interface for mail coming into your secure network and to hide the details of the secure network from the outside.

The configuration provided by IBM Firewall 3.1 can be used with no further modification. However, you will have to make changes to adjacent mail servers to let them behave in a more secure way. In this chapter we will first describe the standard setup, and then show examples that address some of the most common additional requirements.

12.1 Configuring Mail Handling Using the Standard Process

Configuring mail is similar to the domain name service configuration. You just invoke the GUI configuration dialog and supply the necessary information. In the GUI select **System Administration** and **Secure Mail Server**. To implement SafeMail you have to open a new mail configuration entry (Figure 216) and provide the name of the Internal Mail Server (see Figure 217 on page 232). You also have to specify the secure domain name and the public domain name.

The secure domain is the name of your secure network. SafeMail will hide the sender address of mail originating from this domain. The public domain name is the name by which you are known to the non-secure network, and for which you receive mail.

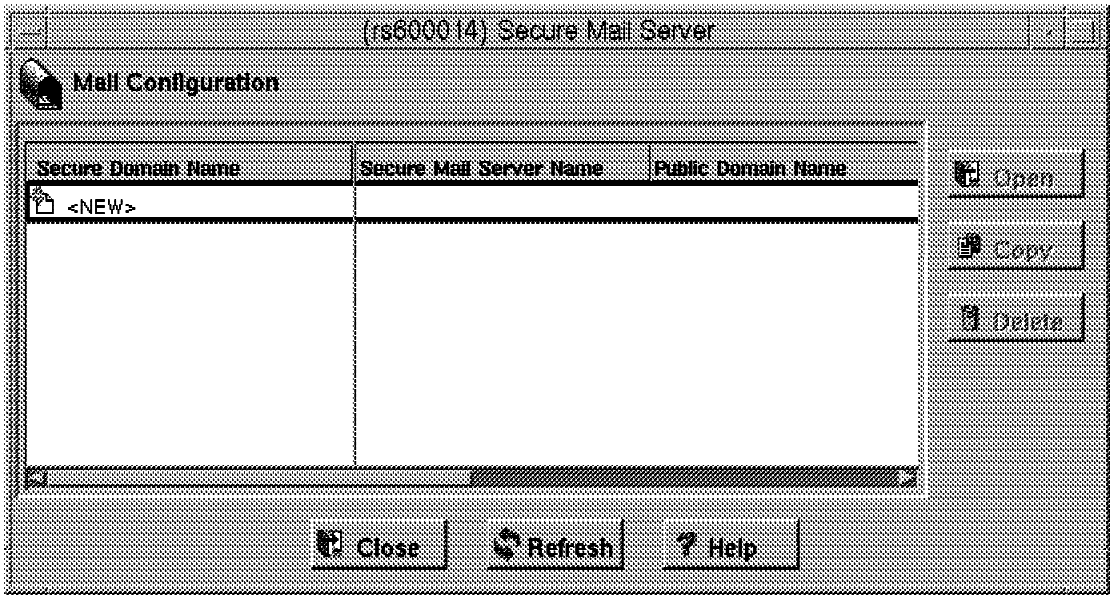


Figure 216. Add New SafeMail Server Entry

In our case, we specified rs600020.itso.ral.ibm.com to be the secure network mail server. It is an RS/6000 machine in the secure network. Our secure domain name is itso.ral.ibm.com. We receive mail with public domain name ral.ibm.com.

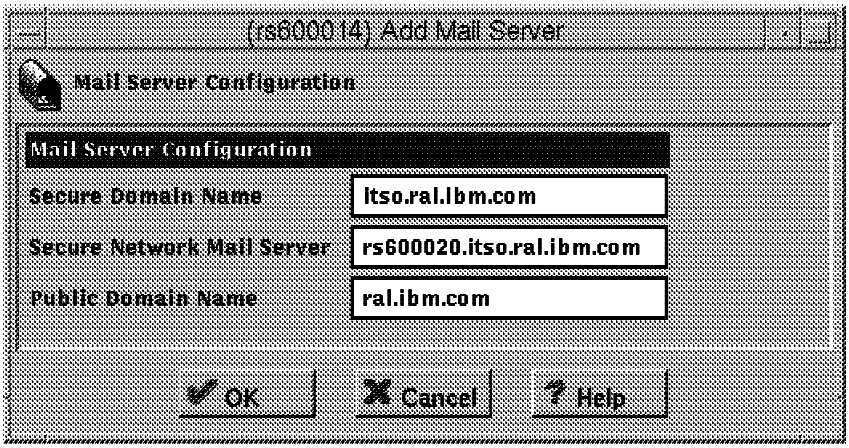


Figure 217. Mail Server Configuration

12.2 Functionality of SafeMail

SafeMail is implemented by the daemon /usr/sbin/fwmaild. This daemon runs under root, but when it accepts a connection it forks a child process that runs under nobody.

SafeMail acts like a mail exchanger, but it does not queue any inbound or outbound mail nor does it store and forward mail. If a destination server accepts the mail, the mail packets are transferred to the destination server like a bidirectional "pipe". If a destination server is not available or does not accept the mail, the mail is rejected.

12.2.1 SMTP Commands

SafeMail can understand all SMTP commands: EHLO, HELO, MAIL, RCPT, DATA, RSET, SEND, SOML, SAML, VRFY, EXPN, HELP, NOOP, QUIT and TURN. Before receiving a RCPT command, SafeMail accepts EHLO, HELO, MAIL, RSET, HELP, NOOP and QUIT, but all others are rejected. After it receives the RCPT command it looks for the destination server according to the path defined by RCPT. If found, SafeMail tries to connect to the destination mail server at TCP port 25. If the destination mail server cannot be found or a connection cannot be made, SafeMail rejects the RCPT command and waits for another RCPT command. If a connection is made, SafeMail transfers subsequent commands (except VRFY, EXPN to a secure mail server) to the destination mail server.

12.2.2 Multiple Secure Mail Servers

You can set up multiple secure mail servers. SafeMail determines the destination secure mail server by looking up the destination specified by the RCPT command. This destination is compared to the domains specified in the "Public Domain Name" field of the secure mail server setup. If a name is found SafeMail tries to connect to the corresponding mail server. If a name is not found, SafeMail attempts to resolve the name as a hostname. If a hostname is not found, Secure Mail attempts to find a mail exchanger that corresponds to the name and if found, it connects to it.

If you have multiple secure-side domains, the mail servers of these domains are required to be able to route mail amongst themselves. This is needed because

SafeMail will route inbound mail to only one of them, so the SMTP server who receives a note to multiple domains must accept all the recipients and resend the note to the other domains as needed.

12.2.3 Host Name and Domain Name Rewriting

SafeMail will replace the secure domain name with the public domain name on all outgoing mail. It will also remove all *Received* headers from the outgoing mail.

SafeMail does not rewrite inbound addressses. It did in IBM Firewall 3.1.0.0, but not in the following updates (3.1.1.1 and later). The secure-side mail domains must be configured so as to accept the public domain name as an alias for their private domain names.

SafeMail also drops the X-LotusFromDomain header as well as the Received: header.

12.3 Recommended Further Configuration

There are items on the internal mail server that we recommend you consider changing:

- Handling incoming mail on the internal mail server
- Setting up a system to understand the MX record

We will describe each of them in turn. The examples in this chapter will meet many typical requirements, but there are sure to be some subtleties in your particular environment that we have not covered. Unfortunately, it seems that whoever devised the format of the sendmail configuration file had a warped sense of humor. It is *not* a simple thing to modify. If you are going to do any serious work with /etc/sendmail.cf, we recommend *sendmail* by Bryan Costales (published by O'Reilly, ISBN: 15620562).

12.3.1 Setting Up a System to Understand the MX Record

Strictly speaking, this is not part of the firewall mail configuration at all, but something that external mail servers have to configure in order to be able to use the gateway. This includes machines inside the DMZ as well as machines that are really external. When the originating mail server wants to send a mail message, it will first try to resolve the destination name into an IP address by looking at the HOST file. If that fails (and if support for the MX record has been enabled), it will use DNS to try to find the address of a mail exchanger that will handle the destination domain. If no MX record satisfies the search, it will look for an DNS A record.

In many mail implementations, such as the one in OS/2 TCP/IP, support for MX is enabled by default. The sendmail in AIX 4.2 and above is MX enabled by default. Other systems require a sendmail configuration update to enable MX support. The following excerpt shows the entry in /etc/sendmail.cf on an AIX system 4.1 that allows the sender to use this support:

```
# Define whether and how to use a name server for resolving recipients.
# Possible values are:
#       MR      use Mail Rename records to resolve recipient users
#       MB      use Mail Box records to resolve recipient users
#       MG      use Mail Group records to resolve recipient users
#       MX      use Mail Exchanger records to resolve recipient hosts
#       ANY     query for ANY records, rather than just CNAMEs, when
#                 canonicalizing the recipient host; NOTE that this
#                 cannot be used when there may be wildcard MX records
#                 for the local domain or any of its parents, since
#                 sendmail will accept this response as the
#                 canonicalized hostname and end up with something like
#                 "host.domain.local.domain"
#       ALL     use all of the above
# You may use any combination of these, although it is recommended that
# you specify MB if MR is specified.  For example, "OK MG MX" would enable
# the use of Mail Group and Mail Exchanger resource records.
# The default is not to use a name server for resolving recipients.
#OK MX
OK MX MR MB MG
```

Figure 218. Setting Up Sender to Use MX Records in AIX 4.1

12.4 Mail Handling Examples

It is easier to understand the interaction of the different mail configuration elements by looking at examples. We will consider two configurations, one using an SMTP mail client inside the secure network and the other using a Post Office Protocol (POP) client. We will track the process by which messages go from sender to receiver and the configuration elements that control each step.

12.4.1 Case 1, Incoming Mail to SMTP Client in the Secure Network

In the first case, we are sending mail to an SMTP client (UltiMail/2 running on OS/2) via the internal mail server. The application protocol is SMTP from end-to-end.

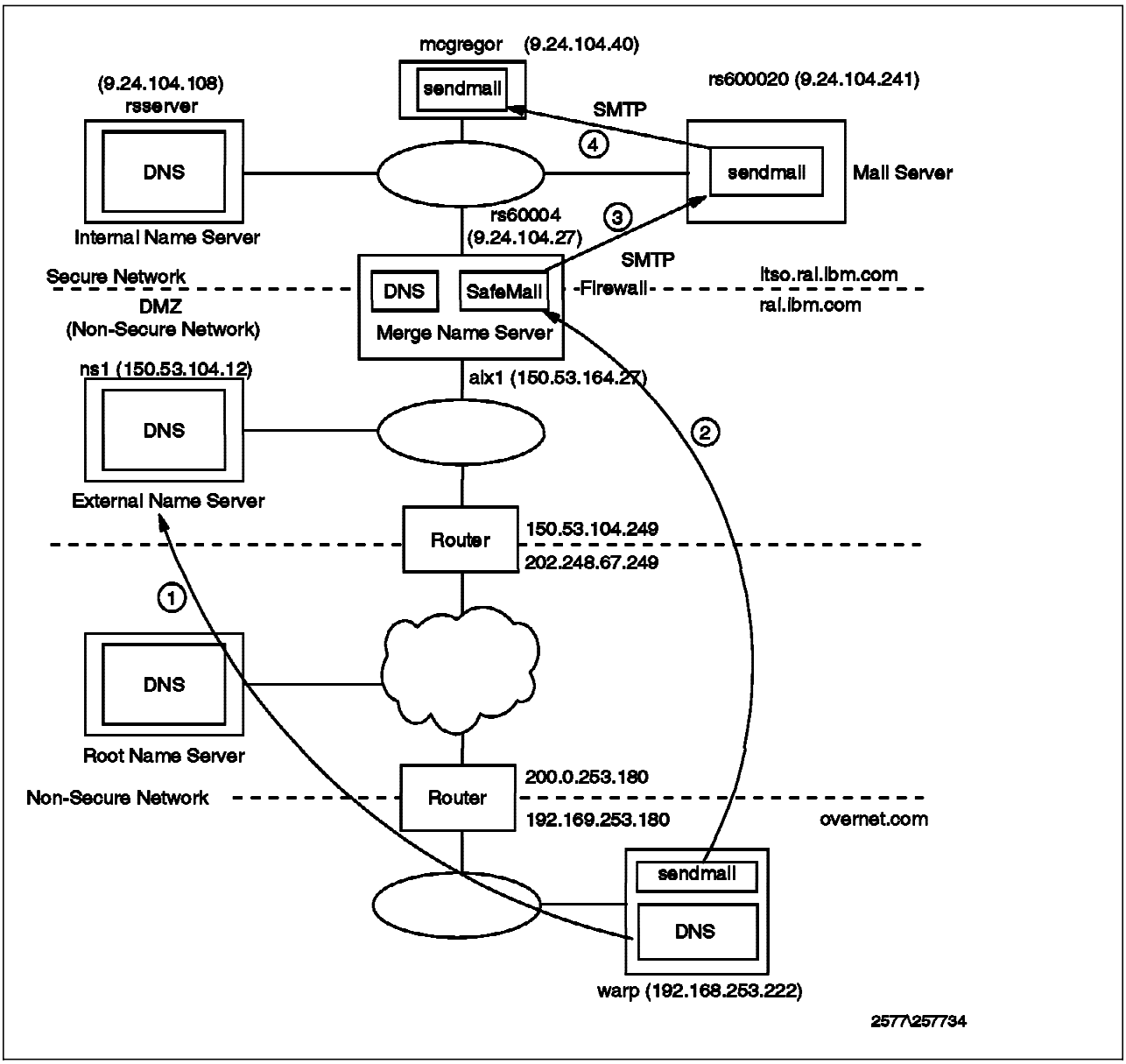


Figure 219. Tracking Incoming Mail to an SMTP Client

The originating user is root@warp.overnet.com and the destination address is rob@ral.ibm.com. However, we want the *real* recipient to be user ID rob on node mcgregor.itso.ral.ibm.com.

Now we will describe how the mail gets through.

Step 1 Before it can send the message, warp.overnet.com must resolve the destination IP address. It uses the following process:

1. The mail client on warp.overnet.com asks DNS on warp.overnet.com to resolve the IP address of ral.ibm.com (that is, the *apparent* IP name of the destination).
2. DNS on warp.overnet.com cannot resolve the IP address of ral.ibm.com.
3. Therefore, DNS on warp.overnet.com sends the same request to the root name server.

4. The root name server knows that ns1.ral.ibm.com is the primary server for the ral.ibm.com domain, so it passes the request on to it.
5. The request will again fail, because ns1.ral.ibm.com has no IP address record for ral.ibm.com.
6. However, warp.overnet.com is tenacious. It may not be able to find a direct IP address mapping for ral.ibm.com, but it is configured to allow the use of a mail exchanger by means of an OK MX record in /etc/sendmail.cf (see Figure 218 on page 234).
7. There *is* an MX record in DNS on ns1.ral.ibm.com.

```
ral.ibm.com.    IN      MX      10      aix1.ral.ibm.com.
```

Figure 220. The /etc/named.zone File on the External Name Server

   The MX records specify a mail exchanger for a domain name. In our case, it indicates that mail to @ral.ibm.com will be forwarded to @aix1.ral.ibm.com.

8. Now the mail client on warp.overnet.com knows to send messages to the IP address of aix1.ral.ibm.com.

Step 2 The mail client on warp.overnet.com opens a connection to the SMTP port (TCP/25) on aix1.ral.ibm.com instead of ral.ibm.com. Note that this is the firewall system, so the rule base has to be configured to permit this session.

The mail message is received by SafeMail on the firewall.

Step 3 SafeMail on the firewall does not store the mail message, but just receives it and reroutes it on a session to the SMTP port on the internal mail server.

The internal mail server (rs600020.itso.ral.ibm.com) receives the mail message, which is addressed to rob@ral.ibm.com, since SafeMail does not rewrite inbound RCPTs. Since rs600020.itso.ral.ibm.com has an alias configured, it accepts the mail anyway.

Step 4 The internal mail server does not store the mail message. Instead, it will reroute it to rob@mcgregor.itso.ral.ibm.com. This is controlled by an alias definition. The alias definition is the following entry in the /etc/aliases file:

```
rob:rob@mcgregor.itso.ral.ibm.com
```

Figure 221. The /etc/aliases File on the Internal Mail Server

This says that mail to user rob will be delivered to rob@mcgregor.itso.ral.ibm.com.

Finally, sendmail on mcgregor.itso.ral.ibm.com can receive mail for user rob. Figure 222 on page 237 shows the result.

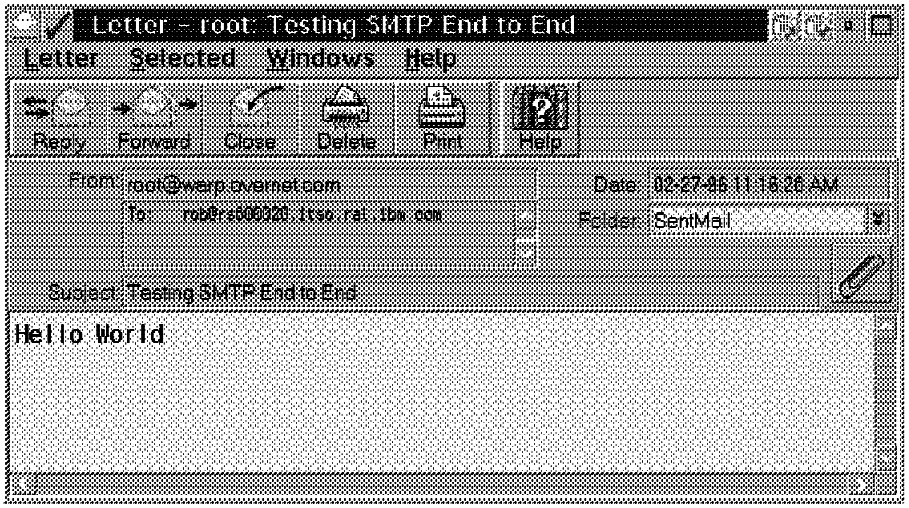


Figure 222. Incoming Mail via SMTP

12.4.2 Case 2, Outgoing Mail from SMTP Mail Client

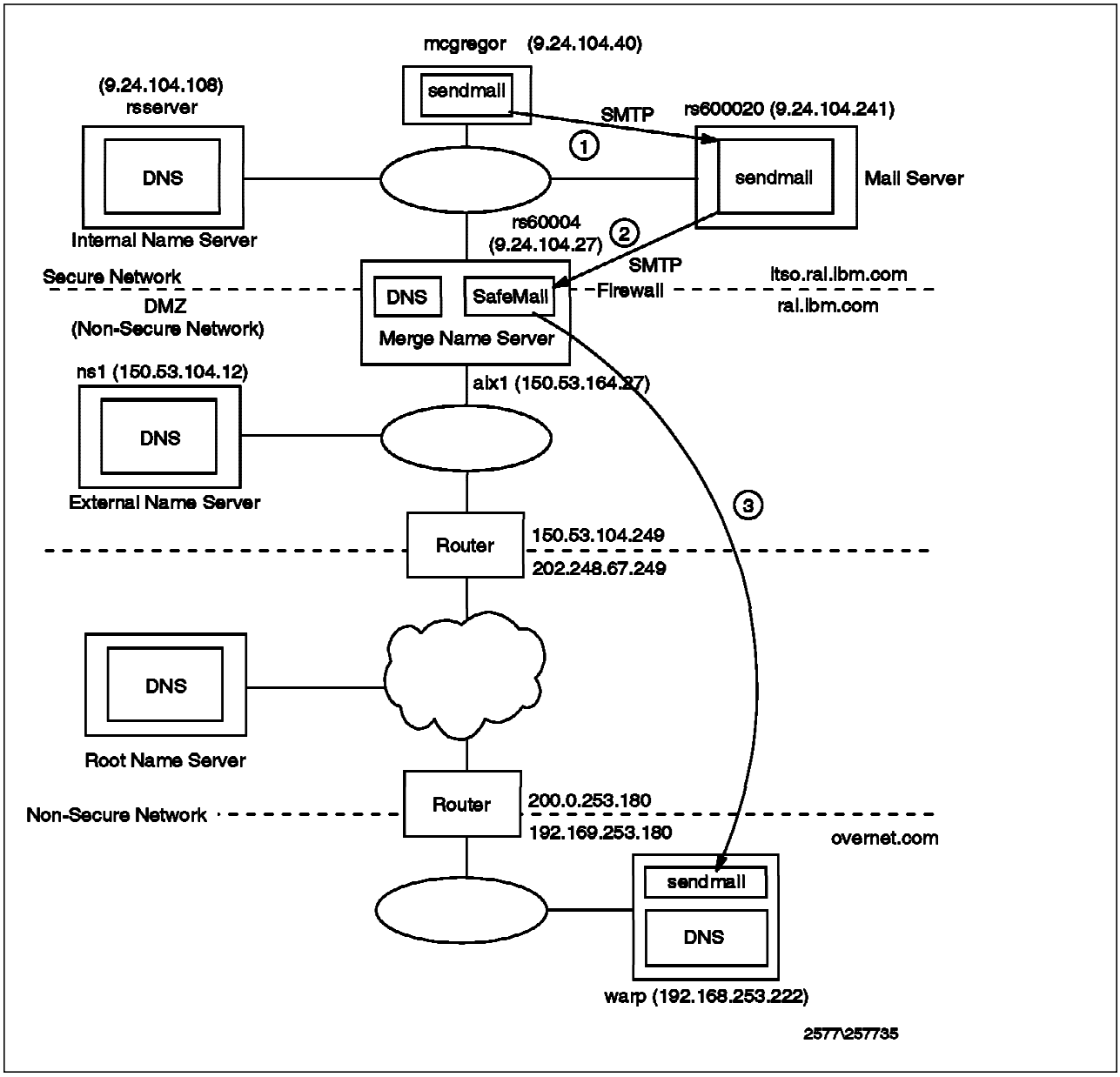


Figure 223. Tracking Outgoing Mail from an SMTP Client

This is the reverse of the previous case. We will send mail from rob@mcgregor.itso.ral.ibm.com to root@warp.overnet.com. As before, the first step is for the originating host to resolve the IP address of the destination. In this case there is no mail exchanger involved, so all that is necessary is a simple IP name lookup. It will first try to solve by looking at the HOST file, and if this fails it will use DNS. At the time of testing we noticed that the target machine or domain of the outgoing mail has to have an MX record. It will not look for an A record. Otherwise the SafeMail will reset the connection. Figure 11 on page 18 describes the process by which an internal node resolves external addresses.

Once the target address has been determined, the following process takes place:

Step 1 The Ultimail/2 client on mcgregor.itso.ral.ibm.com sends mail to the local mail server (rs600020.itso.ral.ibm.com). This is defined by the DV macro in the %ETC%\sendmail.uml file.

```
# DVYour.External.Gateway
DVrs600020
```

Figure 224. DV Macro in the Sendmail.uml File on the Mail Client

AIX Users

If your client is an AIX 4.1 machine, you would specify the DR macro in /etc/sendmail.cf instead of the DV macro shown above. You would specify:

```
DRrs600020
```

in this case.

If your client is an AIX 4.2 machine, you would specify the DS macro in /etc/sendmail.cf instead of the DV macro shown above. You would specify:

```
DSrs600020
```

in this case.

Step 2 The sendmail daemon on the internal mail server receives the mail message and then reroutes it to the firewall (rs60004.itso.ral.ibm.com). This is controlled by the DR macro in the /etc/sendmail.cf file.

```
#DRRelayHostName (AIX version 4.1)
DRrs60004
#DSSmartRelayHostName (AIX version 4.2)
DSrs60004
```

Figure 225. RelayHostName Macro in /etc/sendmail.cf on the Internal Mail Server

Step 3 Next, SafeMail on the firewall (rs60004.itso.ral.ibm.com) receives the mail message. SafeMail will examine the mail headers and it removes all *Received* lines from the headers. SafeMail converts internal domain name inside *From* headers to the public domain name. SafeMail does not store the mail message, but instead reroutes it to the destination address.

Finally, the sendmail daemon on warp.overnet.com receives and stores the mail message.

12.4.3 Case 3, Incoming Mail to POP Client

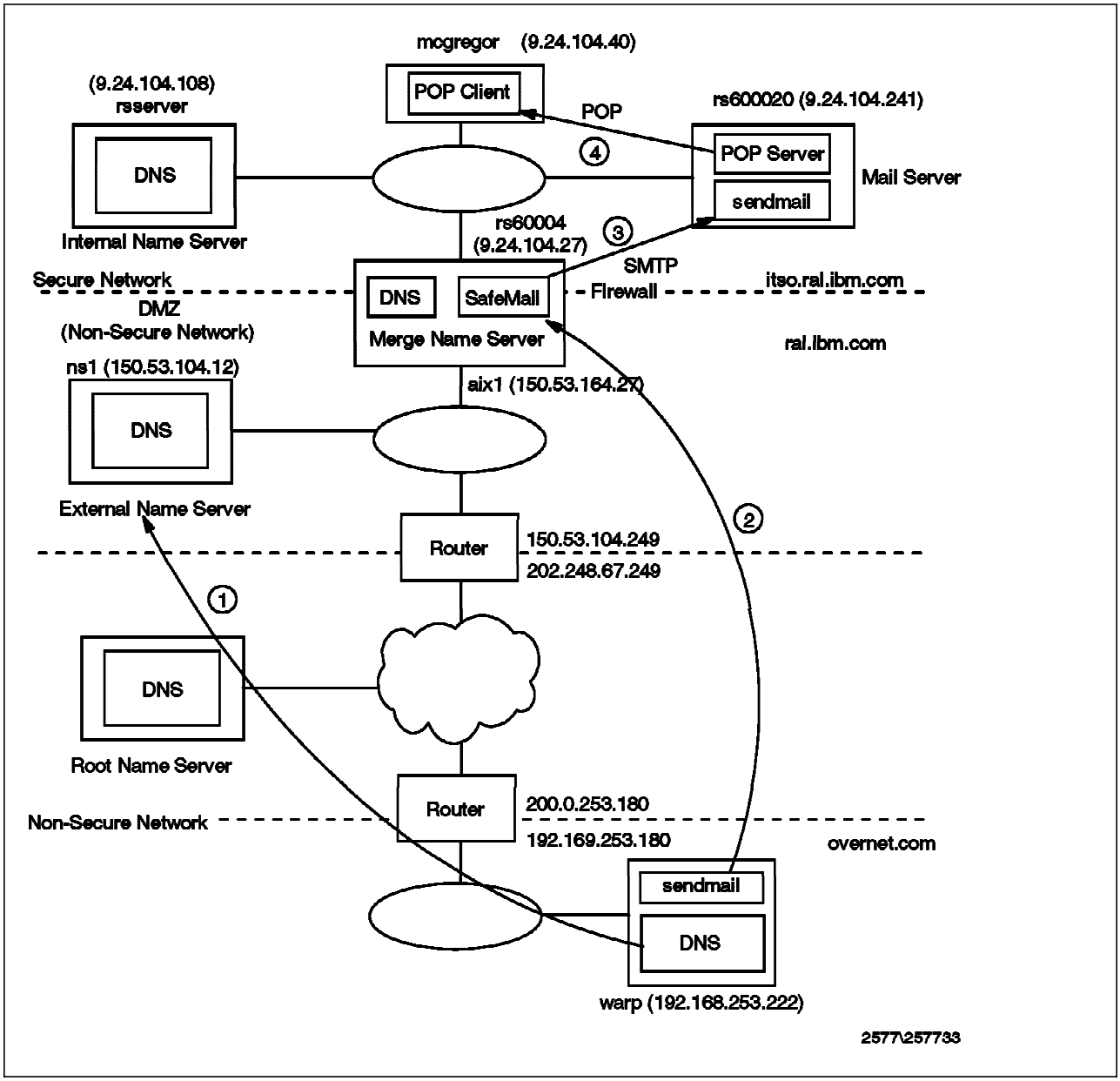


Figure 226. Tracking Incoming Mail to a POP Client

This is very similar to the first case we dealt with, except that, in this case, we use Post Office Protocol 3 (POP3) to receive the mail. POP3 is defined by RFC1225 and it specifies a mail client that requires fewer resources from the host machine than a full-fledged SMTP client. In fact, we used the same product as before, Ultimail/2, configured as a POP client instead of an SMTP client.

Figure 227 on page 241 shows how a POP client accesses mail messages through a POP server. Incoming mail to POP client users is spooled in the mailbox on the server. The users access the POP server indirectly to obtain their incoming mail. The protocol used between client and server is POP3. However, when users send mail, they are sending it using SMTP directly. In fact, it is not necessary to send mail to the mail server. It could be sent directly to the destination host. The principal advantage of the POP approach is that only

one machine, the mail server, has to be permanently available to process mail messages. It also places the disk space to store mail messages on the server instead of distributed among the mail clients. This is a more controlled and economical use of hardware resources.

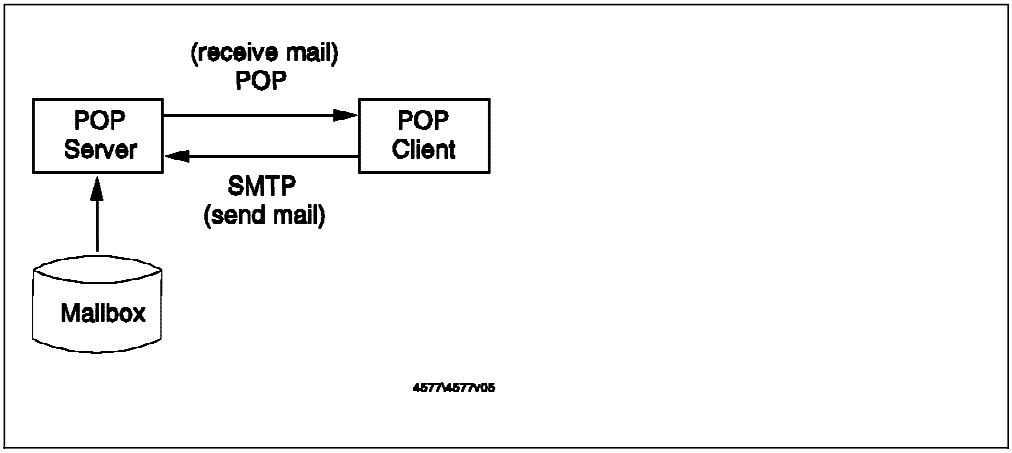


Figure 227. POP Server and Client

Now, we will describe how the incoming mail is processed step-by-step. As before, we will send mail from root@warp.overnet.com to rob@ral.ibm.com, but we expect that user rob on mcgregor.itso.ral.ibm.com will be the real recipient.

Steps 1-3 These 3 steps are exactly the same as case 1 (see 12.4.1, "Case 1, Incoming Mail to SMTP Client in the Secure Network" on page 234), but with one exception. The sendmail daemon on the internal mail server (rs600020.itso.ral.ibm.com) receives and stores the mail message, instead of passing it on to mcgregor.itso.ral.ibm.com.

Step 4 The POP client accesses the POP server on the internal mail server in order to get mail from the mailbox. It then retrieves the mail message from the mailbox.

12.4.4 Case 4, Outgoing Mail from POP Mail Client

The POP client uses SMTP protocol in order to send mail, as shown in Figure 227. In other words, this case is exactly the same as case 2. In POP Mail Client you sometimes can specify your *Reply-To* address. Remember to put there your address that is visible to the Internet. In our case it would be rob@ral.ibm.com.

12.5 What if SafeMail is not for You?

If you find that SafeMail does not provide the functions that you expect from a mail server, you may want to replace it with sendmail. You should change the rc.tcpip file to start the sendmail deamon instead of fwmaild. You will have to also configure sendmail.cf, which is not a trivial task.

Chapter 13. Testing Your Configuration

In this section, we analyze some of the tools that are available for testing your firewall configuration. We use Network Security Auditor (the utility provided by IBM Firewall), strobe and SATAN (some useful tools you can get from the Internet).

13.1 Introduction

Let's face it. Sooner or later somebody will test your configuration. It might be someone from your organization checking up on you, or a cracker from the outer reaches of the Internet. You should test the configuration yourself before somebody else does it for you.

13.2 Network Security Auditor

Network Security Auditor is a tool which scans target machines and tries to exploit detected services on known weaknesses. For example, some obvious passwords are tried on services which ask for a password, like Telnet and FTP, or commands considered dangerous or risky on sendmail, like DEBUG, VRFY and EXPN.

Network Security Auditor scans TCP, UDP and RPC services, for which you can select a predefined scan configuration or define your own scan configuration. Some examples of predefined scan configurations are:

- Default
  - tcp ports: 21,23,25,111,139,512-514,6000
  - udp ports: 69,111,137,161
  - rpc service: nfsmount
- Fulltcp
  - tcp ports: 1-65535
  - udp ports: 69,111,137,161
  - rpc services: nfs,nfsmount,ypserv
  - options: tcp-seq-num ftp-walk-tree
- Firewall
  - tcp ports: 1-65535
  - udp ports: 1-65535
  - options: ip-source-route, tcp-seq-num, ip-options, ftp-walk-tree
  - rpc services: all

Other scan types are: baseline, medium, standard and complete.

When you run Network Security Auditor a Web browser is started from which you can configure setup, start scans and generate reports, as shown in Figure 228 on page 244.

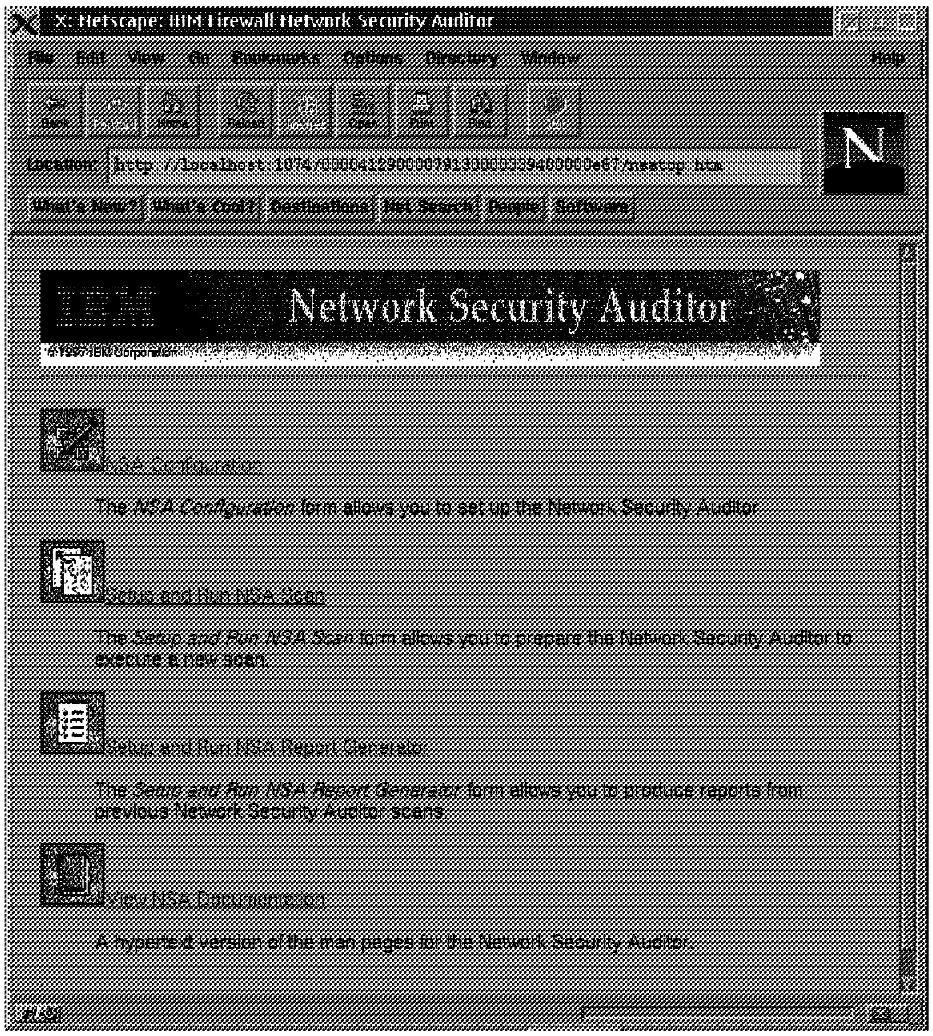


Figure 228. Network Security Auditor Start Window

However, if you want, you can run Network Security Auditor on the command line.

We started a scan on the nonsecure interface (192.168.10.14) of the firewall. The type of the scan was firewall and the output was sent to the data file report.fw, as shown in Figure 229 on page 245.

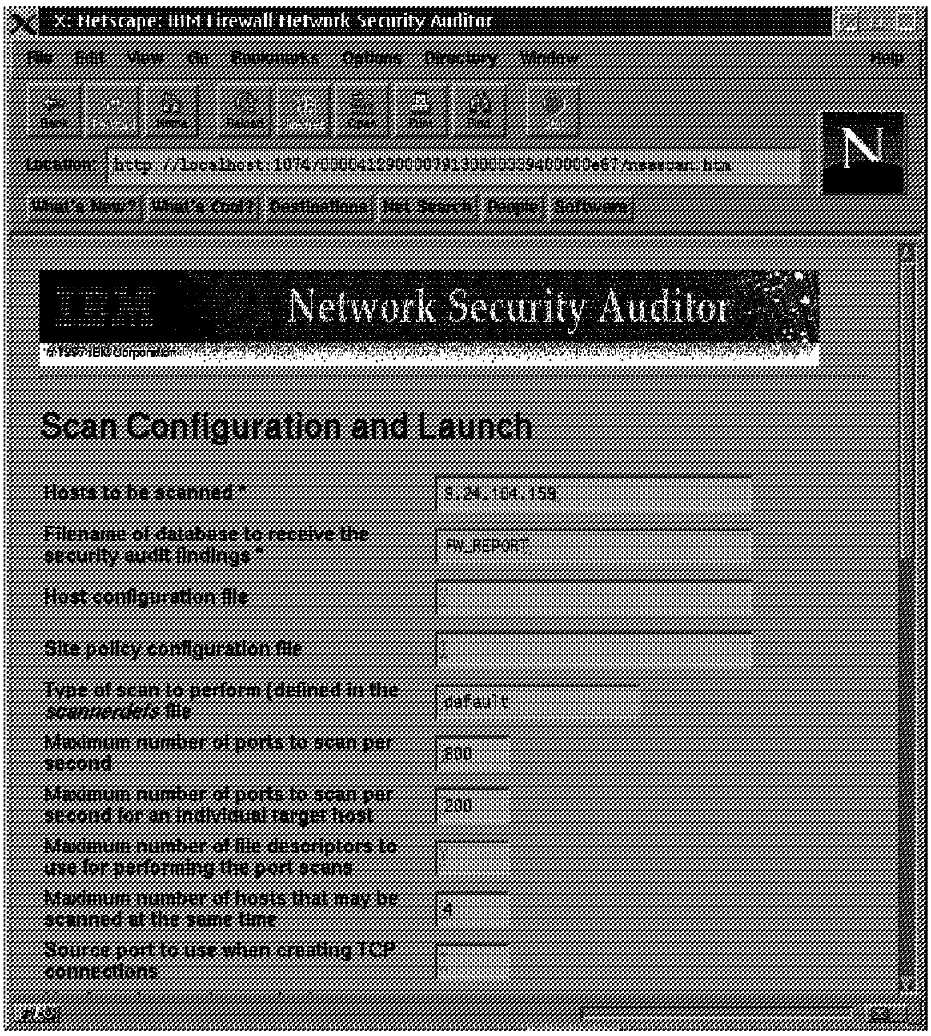


Figure 229. Set up a Scan for Nonsecure Firewall Interface

After you make a scan, you can generate different kinds of reports, such as a standard, summary, vulnerability or policy report. The generated report can then be printed or stored depending on the browser you are using. For example, with Netscape you can store a report as a text, HTML or postscript file. In Figure 230 on page 246 the output of our scan is shown in HTML format. As you can see only the Telnet, FTP and ping responded.

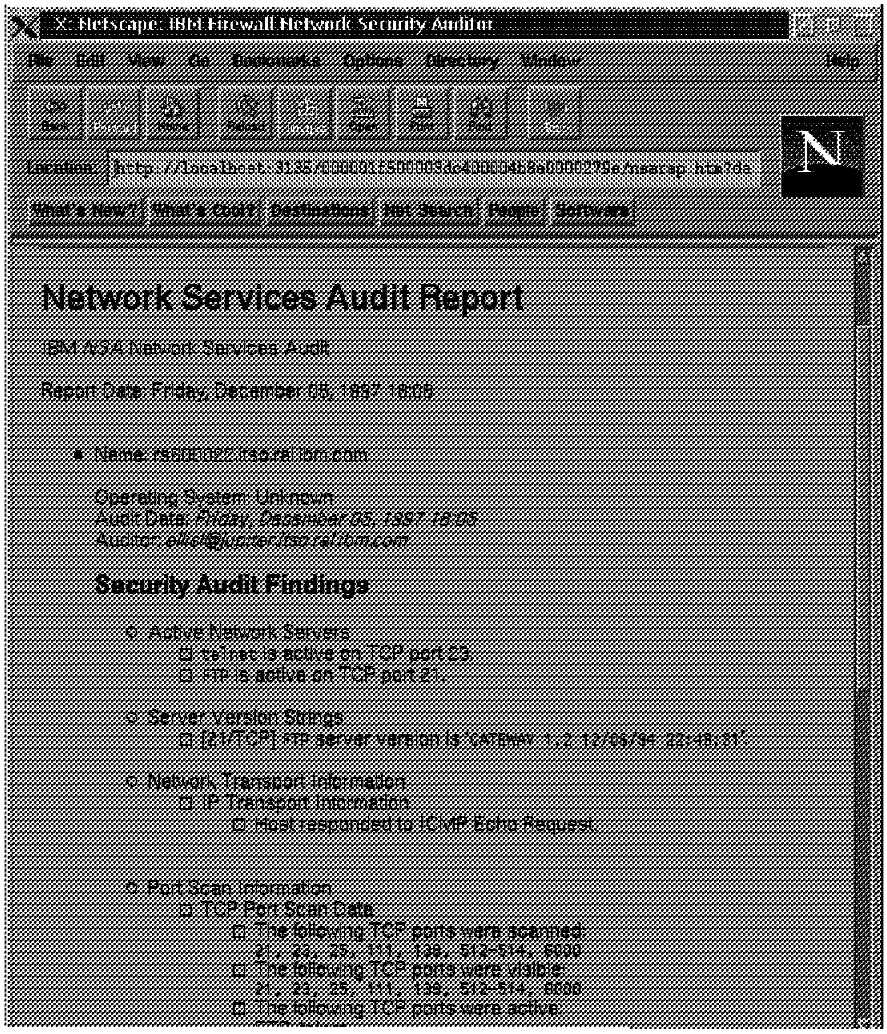


Figure 230. Report of the Firewall Scan

13.3 Strobe

Strobe is a port scanning tool written (and copyrighted) by Julian Assange. It may be used freely and you can get it from ftp://suburbia.net:/pub/strobe.tgz.

It's an extremly fast TCP port scanner, and it allows the user to specify the source port of the connections (in order to emulate a source porting attack). It does not do stealth scanning (see 3.1.4.2, “Stealth Scanning” on page 35) so you will see packets with the SYN bit on if you trace it.

13.3.1 Strobe Using Plain IP Routing

As before, we first scan our target machines with the firewall filters disabled, so it should be able to reach all available ports.

```
# ./strobe -P 20 -t 2 9.24.104.215 9.24.104.241
strobe 1.03 (c) 1995 Julian Assange (proff@suburbia.net).
9.24.104.215   echo           7/tcp Echo [95,JBP]
9.24.104.241   echo           7/tcp Echo [95,JBP]
9.24.104.215   discard        9/tcp Discard [94,JBP]
9.24.104.241   discard        9/tcp Discard [94,JBP]
9.24.104.215   daytime       13/tcp Daytime [93,JBP]
9.24.104.215   chargen       19/tcp ttytst source Character Generator
9.24.104.215   ftp           21/tcp File Transfer [Control] [96,JBP]
9.24.104.215   telnet        23/tcp Telnet [112,JBP]
9.24.104.215   smtp          25/tcp Simple Mail Transfer [102,JBP]
9.24.104.215   time          37/tcp Time [108,JBP]
9.24.104.241   daytime       13/tcp Daytime [93,JBP]
9.24.104.241   ftp           21/tcp File Transfer [Control] [96,JBP]
9.24.104.241   telnet        23/tcp Telnet [112,JBP]
9.24.104.241   smtp          25/tcp Simple Mail Transfer [102,JBP]
9.24.104.241   time          37/tcp Time [108,JBP]
9.24.104.241   pop2         109/tcp postoffice Post Office Protocol - Version 2
9.24.104.241   pop3         110/tcp Post Office Protocol - Version 3 [122,MTR]
9.24.104.241   sunrpc       111/tcp rpcbind SUN Remote Procedure Call
9.24.104.241   loc-srv      135/tcp Location Service [JXP]
9.24.104.241   snmptrap     162/tcp SNMPTRAP [15,MTR]
9.24.104.241   cmip-man     163/tcp CMIP/TCP Manager [4,AXB1]
9.24.104.241   cmip-agent   164/tcp CMIP/TCP Agent
9.24.104.241   smux         199/tcp SMUX [MTR]
9.24.104.241   exec         512/tcp remote process execution;
 .
 .
 .
```

Figure 231. Strobe Scanning with No Firewall Filters

13.3.2 Strobe

Now we activate the firewall, and we verify that the attacker cannot gain any information, because the packets are blocked by the filters (with IBM Firewall configured as a dual-homed firewall).

```
# ./strobe -P 20 -t 2 9.24.104.215 9.24.104.241
strobe 1.03 (c) 1995 Julian Assange (proff@suburbia.net).
#
```

Figure 232. Strobe Scanning with Firewall Active

It is interesting to look at the log files, to see what marks a port scanner leaves behind. You can see a great number of connections coming from the same IP address (150.53.104.12) to different destinations, where a destination is fully defined by an IP address (9.24.104.215) a protocol (tcp) and a port (1-XXXX).

One interesting attack that we can do with strobe is to attempt source-port scanning, asking strobe to use as a source port the ftp-data port (port 20). This emphasizes the point that you cannot trust that a well-known port will always be used by the service that it is assigned to. Figure 233 on page 248 shows the result of running this scan with firewall filters set to allow secure network users to have direct FTP access to nonsecure servers.

```
Feb 11 15:35:09 rs60004 : ICA1036i: #:79 R:d  i:150.53.104.27 s:150.53.104.12 d:9.24.104.215 p:tcp sp:20 dp:1
Feb 11 15:35:09 rs60004 : ICA1036i: #:79 R:d  i:150.53.104.27 s:150.53.104.12 d:9.24.104.241 p:tcp sp:20 dp:1
Feb 11 15:35:09 rs60004 : ICA1036i: #:79 R:d  i:150.53.104.27 s:150.53.104.12 d:9.24.104.215 p:tcp sp:20 dp:2
Feb 11 15:35:09 rs60004 : ICA1036i: #:79 R:d  i:150.53.104.27 s:150.53.104.12 d:9.24.104.241 p:tcp sp:20 dp:2
Feb 11 15:35:09 rs60004 : ICA1036i: #:79 R:d  i:150.53.104.27 s:150.53.104.12 d:9.24.104.215 p:tcp sp:20 dp:3
Feb 11 15:35:09 rs60004 : ICA1036i: #:79 R:d  i:150.53.104.27 s:150.53.104.12 d:9.24.104.241 p:tcp sp:20 dp:3

Feb 11 15:35:09 rs60004 : ICA1036i: #:79 R:d  i:150.53.104.27 s:150.53.104.12 d:9.24.104.215 p:tcp sp:20 dp:28
Feb 11 15:35:09 rs60004 : ICA1036i: #:79 R:d  i:150.53.104.27 s:150.53.104.12 d:9.24.104.241 p:tcp sp:20 dp:28
Feb 11 15:35:09 rs60004 : ICA1036i: #:79 R:d  i:150.53.104.27 s:150.53.104.12 d:9.24.104.215 p:tcp sp:20 dp:29
Feb 11 15:35:09 rs60004 : ICA1036i: #:79 R:d  i:150.53.104.27 s:150.53.104.12 d:9.24.104.241 p:tcp sp:20 dp:29
Feb 11 15:35:09 rs60004 : ICA1036i: #:79 R:d  i:150.53.104.27 s:150.53.104.12 d:9.24.104.215 p:tcp sp:20 dp:30
Feb 11 15:35:09 rs60004 : ICA1036i: #:79 R:d  i:150.53.104.27 s:150.53.104.12 d:9.24.104.241 p:tcp sp:20 dp:30
```

Figure 233. Strobe Scanning Using Source-Porting Technique

You can see how strobe scans 60 ports in just one second (ports 1 to 30 for machines 9.24.104.215 and 9.24.104.241).

Figure 234 shows the output from the tcpdump command for the same scan. Here you can see the SYN bit on, so this shows that it is not stealth scanning (that is, it is initiating the conventional TCP session handshake, instead of emulating the server end of an existing IP connection).

```
15:35:08.293612032 150.53.104.12.20 > 9.24.104.215.1: S 1737212929:17372 12929(0) win 16384 <mss 512>
15:35:08.344706304 150.53.104.12.20 > 9.24.104.241.1: S 1737276929:1737276929(0) win 16384 <mss 512>
15:35:08.345345920 150.53.104.12.20 > 9.24.104.215.2: S 1737340929:1737340929(0) win 16384 <mss 512>
15:35:08.345980288 150.53.104.12.20 > 9.24.104.241.2: S 1737404929:1737404929(0) win 16384 <mss 512>
15:35:08.346621824 150.53.104.12.20 > 9.24.104.215.3: S 1737468929:1737468929(0) win 16384 <mss 512>
15:35:08.347233152 150.53.104.12.20 > 9.24.104.241.3: S 1737532929:1737532929(0) win 16384 <mss 512>
15:35:08.377754624 150.53.104.12.20 > 9.24.104.215.28: S 1740668929:1740668929(0) win 16384 <mss 512>
15:35:08.378351232 150.53.104.12.20 > 9.24.104.241.28: S 1740732929:1740732929(0) win 16384 <mss 512>
15:35:08.378944768 150.53.104.12.20 > 9.24.104.215.29: S 1740796929:1740796929(0) win 16384 <mss 512>
15:35:08.379545856 150.53.104.12.20 > 9.24.104.241.29: S 1740860929:1740860929(0) win 16384 <mss 512>
15:35:08.380142208 150.53.104.12.20 > 9.24.104.215.30: S 1740924929:1740924929(0) win 16384 <mss 512>
15:35:08.380741248 150.53.104.12.20 > 9.24.104.241.30: S 1740988929:1740988929(0) win 16384 <mss 512>
```

Figure 234. Tcpdump Trace of Strobe Source-Porting Scan

13.4 SATAN

SATAN is a tool designed by Dan Farmer and Wietse Venema to scan networks and pinpoint security problems. SATAN has been the subject of some criticism, because it is freely available. Some people feel that it is dangerous to make such a tool so widely available. The authors contend, however, that the dangerous crackers know all of SATAN's tricks already, so it is better to give the knowledge to network administrators, thereby letting them test their own defenses. Unlike other tools, it probes for security holes from outside and searches for a number of different security exposures, in addition to basic port scanning.

SATAN uses a Web browser for its graphical user interface and it generates the reports in HTML form so they can be viewed online within the Web browser. In order to run it you need PERL Version 5 or later.

To configure SATAN, we selected the configuration screen and specified the Level=Heavy option. We limited our probing to network 9.24.104. We disabled the use of PING (as the IBM Firewall blocking of ICMP echo requests could prevent SATAN from probing other services) and also disabled the use of DNS (because we are hiding our network structure in the firewall DNS).

13.4.1 SATAN Using Plain IP Routing

Once again, we first ran SATAN with the firewall filters disabled, so it acted as a normal IP router. As you can see from Figure 235, it was able to discover several security holes.

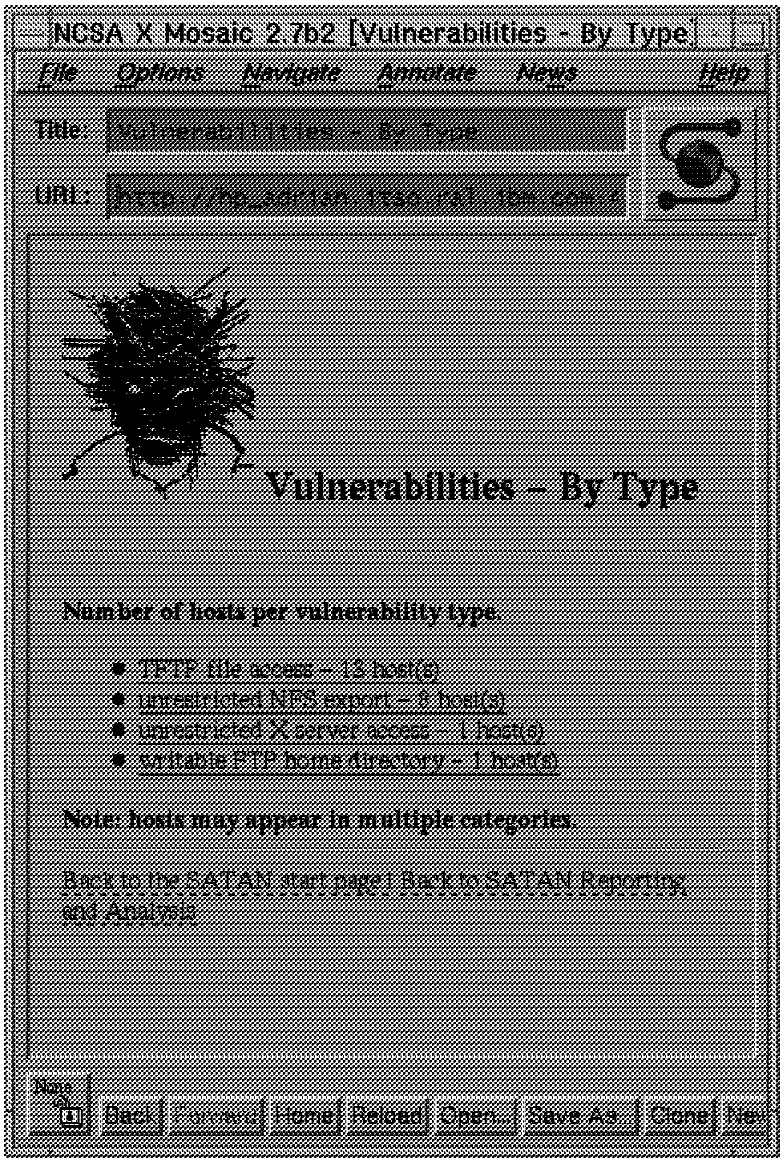


Figure 235. SATAN Using Plain IP Routing

13.4.2 SATAN

Next we turned on the firewall filtering. Once again it was configured as a dual-homed gateway, with no IP routing allowed, so SATAN was not able to reach the secure network hosts.

Figure 236. SATAN Using IBM Firewall

We are not providing any service to the outside network, when we tried to scan the firewall, it was not able to gain any information from the firewall itself.

Chapter 14. Logging

Logging is essential to the day-to-day operation of IBM Firewall 3.1. Unless you log the activity on your firewall and generate alerts for suspicious activity, you could be under attack without even realizing it. Worse, in the event of an attack, you would be seriously hampered in your attempts to determine the origin and target of the attack.

For more information about logging, see "Chapter 16, Managing Log and Archive Files" *IBM Firewall For AIX Version 3.1.1 User's Guide* GC31-8419-00.

14.1 Configure Logging

When IBM Firewall 3.1 is initially installed, the logging facilities need to be configured before they become active.

From the main IBM Firewall 3.1 Netscape panel (see Figure 24 on page 45) select **System Administration** and then **System Logs**, now select **Log Facilities**.

You should now see Figure 237.

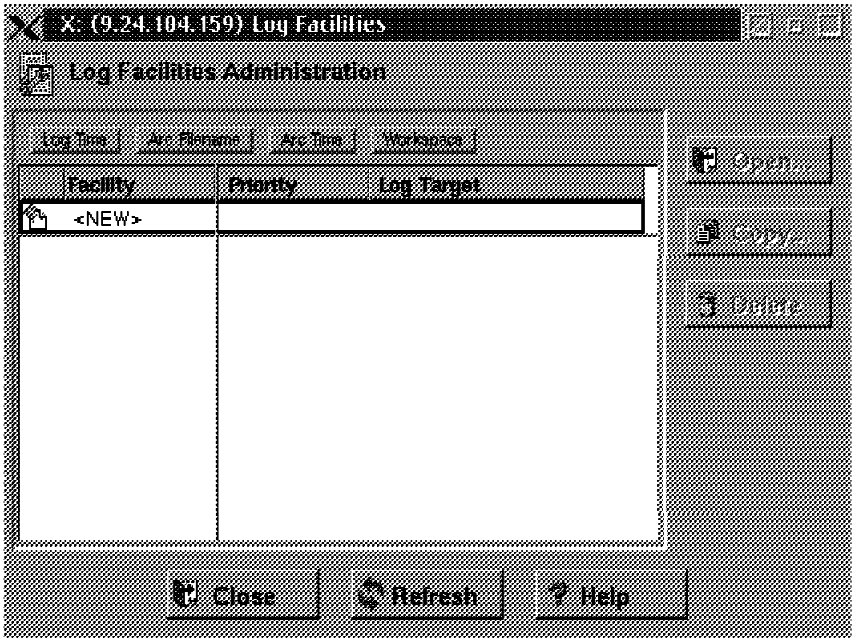


Figure 237. Log Facilities Administration

First we will create a log facility to monitor all activity on the firewall, select **<NEW>** in Figure 237.

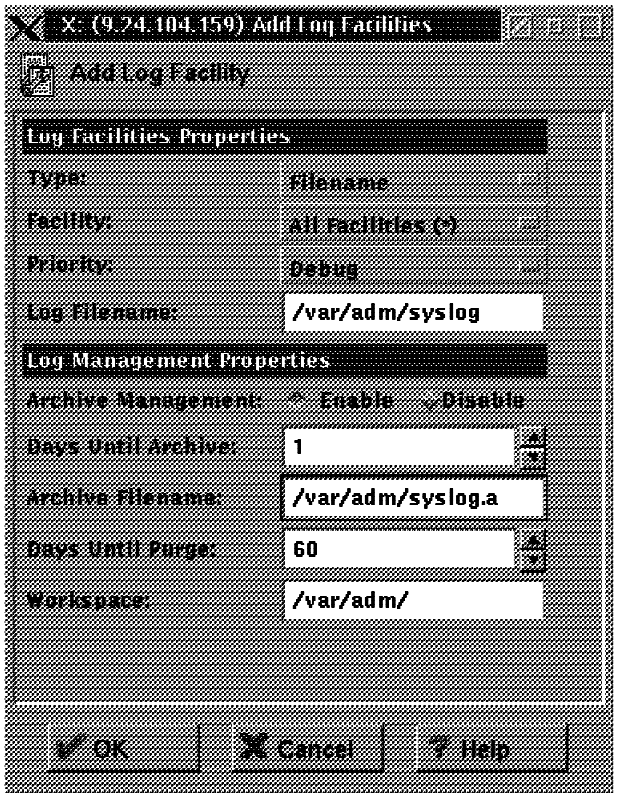


Figure 238. Add Log Facility

We decided to use All Facilities debug logging for the following reasons:

- We have plenty of disk space
- We will archive regularly to efficiently use the space available
- We need to capture everything we can while we test the firewall

See Figure 238 for the settings we chose for All Facilities debug logging.

Where possible we advise you to log as much as you can or risk missing vital information.

Note: You may not be able to use * debug logging if you are using one or more Windows 95 IPSec Client tunnels, as the output from their dynamic filter rules can be enormous. It's also worth mentioning that the debug output from these dynamic filter rules goes to Local2.

We enabled archive management (see also 14.2, “Activate Archiving” on page 254) and set our log to archive once a day, clearing after 60 days. If you find disk space a problem, daily archiving can help.

We repeated the above steps and created several log facilities detailed in Figure 239 on page 253.

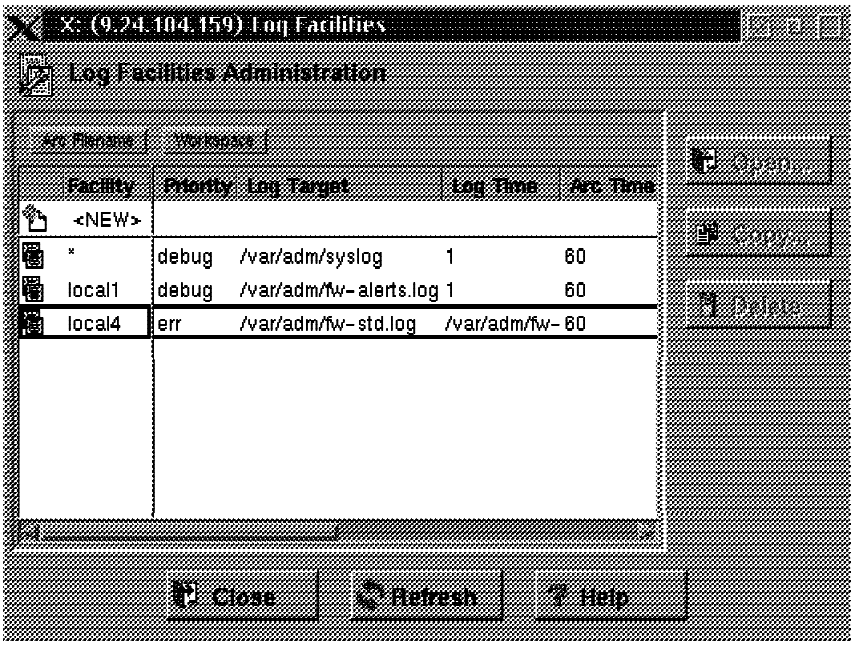


Figure 239. Completed Log Facilities

The All Facilities log will capture all activity, making the Local4 and Local1 logging seem redundant. However the firewall generates alerts to Local1, an essential service. Local4 logging is used by the reporting utilities in combination with the su log.

14.1.1 A Quick Look at Syslog

Syslog is a built-in AIX logging facility. IBM Firewall 3.1 takes advantage of the Local4, and Local1 services provided by syslog to generate logs.

The parameter file for syslog is /etc/syslog.conf. You should however use the Remote Configuration Client to configure log facilities as it will take archive settings into account and refresh the syslogd process for you. There are eight priority levels you can specify to reduce the amount of logging activity for a specific service, they are:

- debug
- information
- notice
- warning
- error
- critical
- alert
- emergency

At debug level all activity for that facility will be logged. At emergency level, very little activity would be logged at all.

IBM Firewall 3.1 only uses three of the available levels of priority, they are:

- debug

- notice
- error

14.2 Activate Archiving

While configuring various log facilities you have probably seen a panel similar to Figure 240.

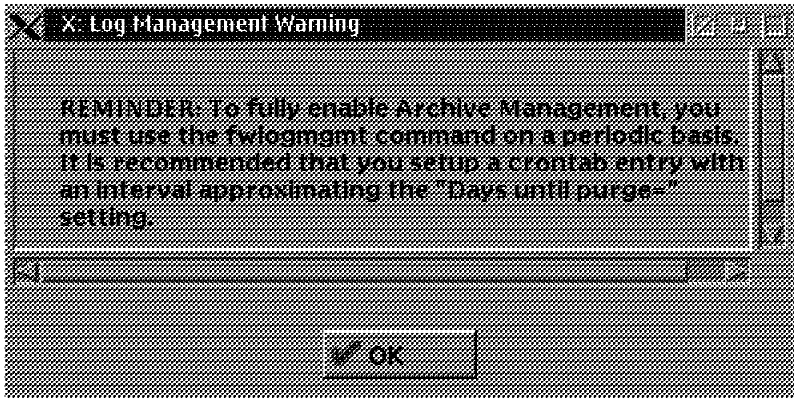


Figure 240. Archiving Warning Message

To complete configuration of archiving you need to modify the AIX crontab file for the root user. Unfortunately it won't stop this message from appearing, but you will be able to safely ignore it.

Logon as the root user in aix. Now edit the crontab file with the command:

```
crontab -e
```

Add the following lines:

```
# FW update archive from logs
0 0 * * * /usr/bin/fwlogmgmt -l > /dev/null 2>&1
# FW purge log entries
0 1 * * * /usr/bin/fwlogmgmt -a > /dev/null 2>&1
```

This will run fwlogmgmt twice.

The first command will archive your logs, the second will purge entries over the age you specified earlier. If your system is particularly busy around midnight you may want to archive early in the morning instead.

The parameter file for fwlogmgmt is /etc/security/logmgmt.cfg. This file is filled out by the Remote Configuration Client, you can check here to make sure your archiving is set ok.

14.3 Managing Alerts

This is the best way to begin extracting information from your logs. Once you become proficient with this facility you will be able to customize a variety of alerts to notify you of firewall misuse.

The Log Monitor facility checks the output to the Local4 log based on user defined thresholds. If a threshold is exceeded then an alert is generated and reported to the Local1 log facility. You can also opt to be alerted by mail or pager, or run an executable.

This section will activate the Alerts Display area of the Main IBM Firewall 3.1 panel (see Figure 24 on page 45).

For more information on Log Thresholds see "Chapter 17. Monitoring the Firewall Logging", *IBM Firewall 3.1 User's Guide*, GC31-8419-00.

14.3.1 Log Monitor Thresholds

We set our base thresholds to alert at high values due to the amount of testing we are performing on the firewall.

You will have to experiment with the threshold settings in your environment so you don't get swamped with too many alerts.

You can configure monitor thresholds:

- `User` - User related authentication failures
- `Host` - Host related authentication failures
- `Total authentication failure` - All authentication failures
- `Message` - ICA Message tags

You can also set up the mail or exec threshold here, which triggers if another threshold is exceeded.

While this list may seem limited, the ability to set a message threshold gives you a lot of options.

A quick overview of messages is at 14.3.1.2, “Message Tag Reference” on page 257; for a complete breakdown see "Appendix A. Messages", *IBM Firewall 3.1 Reference*, SC31-8418-00.

14.3.1.1 Configuring a Monitor Threshold

We wanted an alert to be generated whenever a user failed authentication at the firewall.

Here are the steps we followed to configure the threshold.

1. From the Netscape IBM Firewall 3.1 main panel (see Figure 24 on page 45) select **System Administration**
2. Select **System Logs**
3. Select **Log Monitor Thresholds**

   You see the Log Administration panel in Figure 241 on page 256

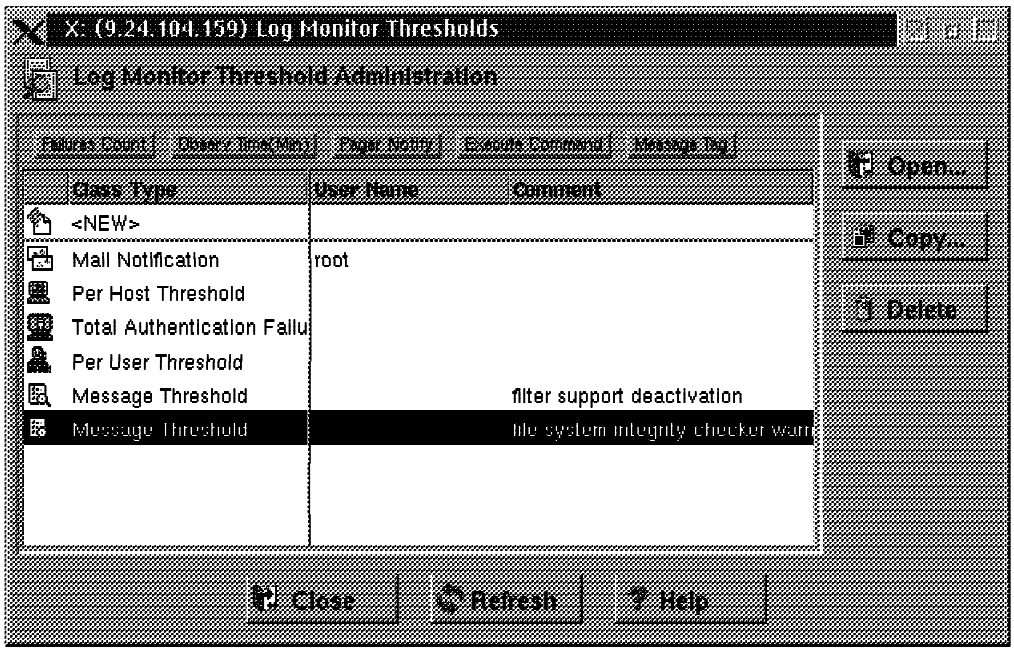


Figure 241. Log Monitor Threshold Administration

4. Select **<NEW>**
5. Fill in the panel as shown in Figure 242

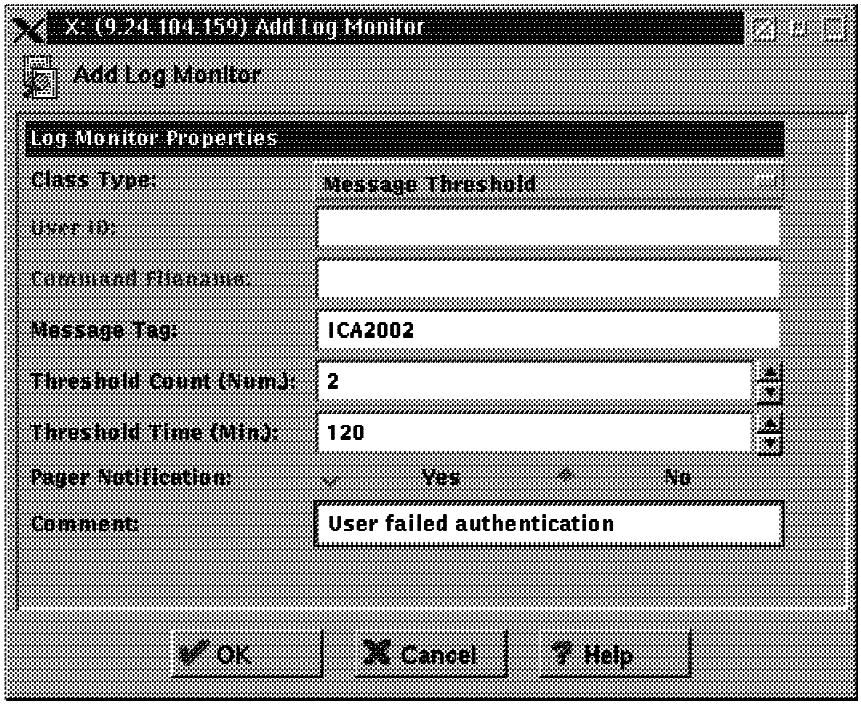


Figure 242. Add Log Monitor

6. Select **OK** to save your monitor

The description for ICA2002 is:

Firewall is unable to authenticate the indicated user name using the specified authentication method.

It's important to put a good comment in here as it is the best way to figure out which message threshold is which.

We attempted to log in with an invalid user name three times to trigger an alert; see Figure 243.

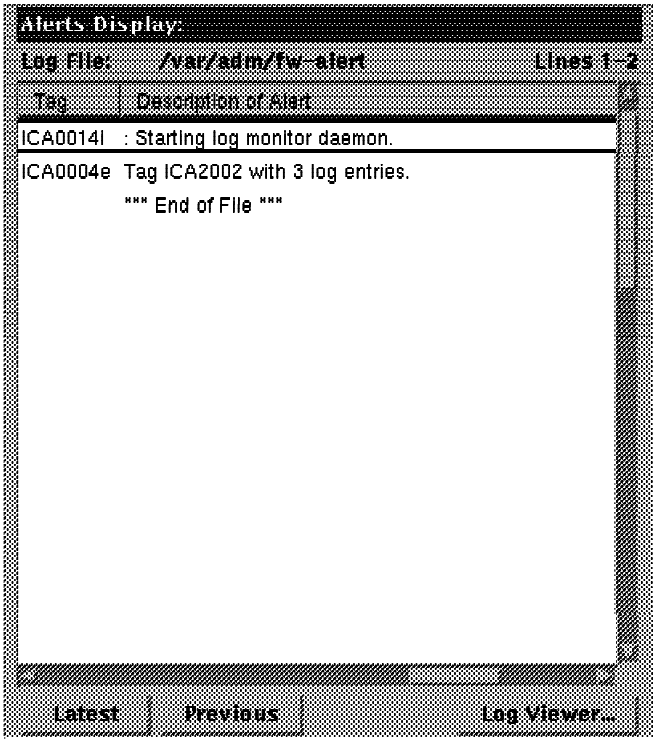


Figure 243. Alert Monitor Panel

14.3.1.2 Message Tag Reference

Here is the message tag breakdown as it is a valuable reference:

ICA	Fixed 3 byte tag identifier
xxxx	A number from 0000 - 9999
a	Message severity indicator • i - info • w - warning • e - error • s - severe

The 0000-9999 is broken into the following categories:

- 0000 - 0999 Intrusion Alarm
- 1000 - 1999 Filters
- 2000 - 2999 Proxy
- 3000 - 3999 Socks
- 4000 - 4999 Pager

- 5000 - 5999 Available
- 6000 - 6999 Dynamic Rules
- 7000 - 8999 Available
- 9000 - 9999 General/Other

Chapter 15. Building Logging Reports

In this chapter we describe how to build useful reports from firewall logging. The reports may provide information about attack rates, type of attacks, but also lots of information of resource usage from the secure network. For example, information about the amount and duration of internet traffic. Or total number of sessions per given period, regardless what session type or total number of bytes transferred by FTP per given period and so on.

Firewall log files contain this information but it is not easily derived from them. Here we describe a method to convert the log files into tables which can be handled by advanced database managers.

15.1 New Log Format

With the new version 3 of the firewall also the format of the logging entries has changed. Now, the logging entries are built of variables, which contain the essential information. The following lines are an example of the new log entries:

```
Mar 19 16:48:36 rs600014 : 1997;7474: 4073;ICA2036i;858790116;
root;9.24.104.37;9.24.104.191;
Mar 19 16:48:38 rs600014 : 1997;9526: 4049;ICA2024i;root;password;
secure network;9.24.104.37 ;
Mar 19 16:50:17 rs600014 : 1997;4230: 2073;ICA1036i;#:;8;R:d;i:9.24.104.191;
s:9.24.104.123;d:;9.24.104.191;p:;udp;sp:;4466;dp:;161;r:;l;a:;s;f:;n;T:;0:;101;
```

The new logging format can still be shown in the old format, that is, the long text format. This can be done by piping the new log through fwlogtxt, for example by:

```
cat syslog | fwlogtxt
```

In this way the previously shown new log lines will translate to:

```
Mar 19 16:48:36  1997: ICA2036i: Telnet Session 858790116 started for user root
(9.24.104.37 :9.24.104.191).
Mar 19 16:48:38  1997: ICA2024i: User root successfully authenticated using pass
word authentication from secure network:9.24.104.37 .
Mar 19 16:50:17  1997: ICA1036i: #:8 R:d  i:9.24.104.191 s:9.24.104.123 d:9.24.1
04.191 p:udp sp:4466 dp:161 r:l a:s f:n T:0 e:n l:101
```

So, if you have developed your own log reporting tools for the old log format you can still use them on the new log files after they are translated.

15.2 Convert Logging to Tables

With the Firewall Reporting Utilities it is possible to convert the firewall logging into tables. These tables can then be imported into a database system such as DB2/6000, DB/2, or Oracle. In this way you can use all the power of the Structured Query Language (SQL), or other tools like IBM's Visualizer or Query Management Facility to query the data and generate reports.

The tables are in delimited ASCII (DEL) file format, with no character string delimiters, and using semicolon (;) as the column delimiters.

The tables are stored in files with a .tbl extension. The Firewall Reporting Utilities can be installed separately, and should be installed on another machine

than the firewall. This may be the Sysloghost or a machine which contains some database facilities, such as DB/2.

There are two ways to generate the tables, by SMIT or via the command line. By using SMIT, it is assumed you supply the results of the firewall log management facility. These are archived compressed log files. The tables are generated as follows:

```
Report Utilities - IBM Firewall V3R1 for AIX
  Create Tabulated Message Files
    Enter Log Archive File Name                          [logarch1.a]
```

After this you can select the log file and determine the directory in which you want the tables. You can also choose to append to existing tables or overwrite the tables. The following SMIT entries appear:

```
* Log File Name                                        [970319syslog.Z]
  Log File Type                                         Firewall log
  Append to existing files                              yes
  Directory for Output Files                           [/var/data]
  Log Archive File Name                                 /tmp/arch.a
```

The tables can also be generated directly from a log file by the following command:

```
fwlogtbl -a -d /var/data syslog
```

This will generate tables from the file syslog in the directory /var/data.

15.2.1 Using DB/2 to Query the Tables

Here we describe how to use DB/2 to process the tables, but the tables are in a format that can be handled by other database managers as well, and the SQL queries can be handled by other database managers.

You need the following files to handle the tables: fwschema.ddl, fwimport.dat, fwqrysmp.dat, these can be found in the /usr/lpp/fw/sample directory. We assume that you have DB/2 installed on a machine and have an instance defined.

15.2.1.1 Create the Initial Database

First you have to create an empty database. This is done by the following command:

```
db2 create database fwlog
```

Next, you must connect to fwlog, by:

```
db2 connect to fwlog
```

Similarly, if you stop using the database, disconnect with:

```
dbs disconnect fwlog
```

To create the tables and all indexes needed, the following command must be issued once:

```
db2 -vf fwschema.ddl > schema.out
```

Check schema.out for possible errors.

15.2.1.2 Import the Tables into the Database

Go to the directory where you put all the tables that were generated from the log files. Run the following command to load the data from the tables into the DB/2 database:

```
db2 -vf fwimport.dat > import.out
```

If the user of the import command is not the creator of the tables, the table names in the fwimport.dat file may need to be prefixed with the name of the table creator and a dot. Each import command in the fwimport.dat file produces info in a <tablename>.msg file. Furthermore, there is information on the standard out, so in import.out.

15.2.1.3 Queries

The file fwqrysmp.dml, provided with the Report Utilities, contains some SQL sample queries you can try on your database. The following command starts the query:

```
db2 -vf fwqrysmp.dml > report.out
```

The results can be found in report.out.

Which queries you need depends on the kind of reports you want to make. We suggest to study the example queries and the file fwschema.dll which contains the structure of the tables. After that it will be relatively easy to build your own queries.

Examples: We conclude with some sample queries and their output:

```
SELECT USERID, APPLICATION, DATE_TIME from SESSION where SESSION_EVENT = 'begin'

USERID   APPLICATION DATE_TIME
-------- ----------- --------------------------
root     telnet      1997-03-06-17.45.12.000000
root     telnet      1997-03-06-20.08.28.000000
root     telnet      1997-03-14-19.34.22.000000
root     telnet      1997-03-14-19.34.22.000000
root     telnet      1997-03-17-15.17.09.000000


SELECT INTERFACE, DIRECTION, SRC_IP, DST_IP, PROTOCOL, SRC_PORT, DST_PORT from
FILTER_MATCH where ACTION='deny' and DATE_TIME between '1997-02-03-00.00.00' and
'1997-08-27-00.00.00'

INTERFACE  DIRECTION SRC_IP          DST_IP          PROTOCOL SRC_PORT DST_PORT
---------- --------- --------------- --------------- -------- -------- --------
secure     inbound   9.24.104.32     9.19.169.87     icmp            8        0
secure     inbound   9.24.104.205    9.24.104.191    udp          2220       53
secure     inbound   9.24.104.205    9.24.104.191    udp          2220       53
nonsecure  inbound   205.201.12.4    192.168.10.14   tcp          1025       23
nonsecure  inbound   193.58.42.1     192.168.10.14   tcp          1450       21
```

15.3 Stalker

Stalker is a software product from Trusted Information Systems. It examines data from the AIX auditing subsystem. Based on that audit data Stalker can trace and browse security violations and other interesting events. Stalker has a rich set of options to produce security violation reports. The user interface is based on X Windows.

Stalker has also a misuse detector. It is based on a misuse signature database. This database has misuse signatures of known attacks, attempts to exploit known system vulnerabilities and typical outcomes of system attacks. Stalker analyzes the audit data and searches for evidence of misuse. Stalker Manager designates a storage hierarchy for audit trail files and conditions for moving files to these locations.

Stalker has basically two components: Stalker Manager and Stalker Agent. The recommended configuration with the firewall is to install both Stalker Manager and Agent in the firewall. There is a special version of Stalker that is suitable for IBM Firewall 3.1. Stalker for Firewalls is modified so that Stalker Manager and Agent can communicate locally inside the firewall. Normally they would communicate using NFS and remote commands.

If you want to run Stalker Manager and Agent in different machines, you can secure the connection between them with IPSec tunnel, Secure Shell (ssh) or Kerberos. The installation process will modify AIX audit subsystem and add more audit events to it.

Stalker Manager has to be able to get access to a number of IBM Firewall 3.1 configuration files and program binaries. It has to be able to access the data produced by the AIX audit subsystem. Stalker does not use the syslog data produced by IBM Firewall 3.1 at all.

You can get more information about Stalker from http://www.haystack.com.

Chapter 16. How to Use the IBM Firewall to Counter Security Holes

Much thought has gone into finding weaknesses in computer security and responses to defend them. An excellent summary of 42 such potential weaknesses may be found in *Firewalls and Internet Security: Repelling the Wily Hacker* (Cheswick and Bellovin, Addison-Wesley, 1994). In this chapter, reproduced with the permission of the authors and publisher, we list those points (from the 42) that relate directly to firewall configurations and include notes on how the IBM Firewall 3.1 can help you to counter them.

16.1.1.1 1. Password Failures Are the Biggest Single Problem

The most common problem is that people pick trivial passwords that are easy to guess. Guessing in this respect does not necessarily mean repeatedly trying to log in. If a cracker can get a copy of the /etc/passwd file, or its shadow /etc/security/passwd in the case of AIX, it can be used for a *dictionary attack* using a tool such as crack. The main way a firewall helps is by preventing access to machines in the secure network, thus reducing the number of places that crackers can do their guessing. You can use the facilities of AIX to impose password rules. If you want to avoid this problems and sniffing problems, you should use one-time passwords as a solution.

Of course, there are some password weaknesses for which the only solution is user education, such as the *social engineering* attack: "Hi, this is Fred. Joe asked me to check out a problem on the machine; can you give me a guest ID?".

16.1.1.2 2. Sequence Number Attacks Can Subvert Address-Based Authentication

Some TCP/IP servers use a simple authentication scheme based solely on the IP address of the client. Examples are rsh, rexec and rcp. An attacker can subvert these by predicting IP sequence numbers and inserting bogus messages into the session. The firewall can only protect against this if you use it to prevent such sessions from passing across it. You should try to avoid the need to route such services outside your secure network, thereby allowing use of a blocking IP filter.

You should also beware of crackers using secure network IP addresses from the nonsecure side of the firewall. We discussed a filter that will block such attempts in 6.2.1, “Rules to Block Attempts at IP Address Spoofing” on page 73.

16.1.1.3 3. It Is Easy to Spoof UDP Packets

A UDP packet has none of the session context (synchronization, sequence numbers) of TCP, so each packet stands alone. The only totally secure answer to this is not to allow UDP-based protocols across your firewall, or at least to choose very carefully which protocols you do allow. It is again important to block nonsecure nodes spoofing as secure network addresses.

16.1.1.4 4. ICMP Packets Can Tear Down Connections between a Pair of Hosts

This is really only a problem with older TCP/IP implementations that do not pay attention to session-specific information in ICMP messages such as destination unreachable or redirect. The AIX TCP/IP implementation on the SNG firewall will *not* stop all sessions when it receives such a message.

16.1.1.5 5. ICMP Redirect Messages Can Subvert Routing Tables

The ICMP redirect message tells a host that a more optimum route exists to a destination. Abuse of this can allow a cracker to either intercept messages or masquerade as a session partner host. As we described in 3.1.3, “An Introduction to TCP Packets” on page 33, you can use SNG filtering rules to block incoming redirect messages and log the outgoing redirect messages in order to notify the responsible host later.

16.1.1.6 6. IP Source Routing Can Bypass Routing Tables

The source routing field in an IP packet tells the receiver to override the normal route to the originator of a request. As with the ICMP redirect message, this is a powerful way for an attacker to eavesdrop or masquerade. By default, AIX will not obey source routing requests. This prevents the route tables on the firewall itself from becoming corrupted. The IBM Firewall 3.1 automatically prevents IP source routing fields from being transmitted, so sessions passing through the firewall are also protected from this attack.

16.1.1.7 7. It Is Easy to Generate Bogus RIP Messages

The Routing Information Protocol (RIP) is a technique used by IP routers to communicate routes to each other. You can block RIP on TCP port 520. Unless you are a major network provider, you should use static routing with proper routes to your own networks and a default route to your network provider. There is no way to know whether behind one of the gateways that you route to lies subverted RIP dynamic routing.

16.1.1.8 8. The Inverse DNS Tree Can Be Used for Name-Spoofing

In this attack, the attacker gains control of inverse name resolution and corrupts it so that his IP address resolves to a node name that your machine trusts. This is only effective against applications that rely on host names for authentication (rsh, rcp, X-windows). Recent versions of these servers (for example, in AIX) do a double-check to make sure that the name → address and address → name mappings are consistent. There have also been exposures in Web browsers with Java support which allow information to be sent to another machine if the inverse name resolution is corrupted. This problem is also fixed by means of a double-check on the name resolution.

If you are not sure that all your hosts have this kind of checking, the safest thing to do is to prevent these applications from crossing the firewall by using filters. If you must have them routed, you may want to manually define the name/address mappings for trusted nodes in your secure network name server.

16.1.1.9 9. The DNS Cache Can Be Contaminated to Foil Cross-Checks

In the previous attack, the firewall DNS server was fetching address mapping information from a corrupted remote DNS server. This, more subtle, version relies on the attacker sending phony DNS responses to populate the firewall DNS server's cache, *before* requesting the session. In this case the cross-check will succeed. We believe that the SNG name server is not susceptible to this attack.

The net result of this kind of DNS attack is that name-based authentication is fundamentally insecure, so you should block such services unless absolutely necessary.

16.1.1.10 10. Return Addresses in Mail Aren't Reliable

The return address in a mail message is placed there by the mailer, and so it can be set to anything. There is nothing you can do to stop people doing this to you, but you should educate staff to be suspicious about unusual return addresses. Sensitive information should not be put in a mail message without some secondary validation that the message is going to the correct person.

16.1.1.11 11. sendmail Is a Security Risk

The sendmail program has been the entry point for many hacking attacks over the past few years. All of the known weaknesses have been addressed in the current AIX version, but it is likely that there are others as yet undiscovered. On the SNG firewall, sendmail is restricted to a simple relay function. This has the dual advantage of providing a less tempting target for crackers and hiding the real secure network mail gateway behind it. You could make the SNG sendmail relay more secure by preventing it from doing local mail delivery which would mean you could avoid running it under the root ID. We did not have time to explore this approach during this project. If you want to disable sendmail, you can do it with the command:

```
/usr/sbin/chrctcp -S -d sendmail
```

16.1.1.12 12. Don't Blindly Execute MIME Messages

Multipurpose Internet Mail Extensions (MIME) is an encoding standard that allows references to other files to be embedded in mail-like messages. MIME-encoded messages are not only handled as mail, it is also the standard used to encapsulate non-text media in WWW documents. The idea with MIME is that it includes instructions on where to find resources (files for example) and how to handle them. However, the crackers could plant their own commands in an otherwise innocent message. Client applications that handle MIME messages should guard against this, but it is not easy to do without restricting the functionality of the product.

This problem is at the application level, so there is nothing the firewall can directly do about it. However, refer also to 16.1.1.24, “25. Be Careful about Interpreting WWW Format Information” on page 267.

16.1.1.13 13. It Is Easy to Wiretap Telnet Sessions

You should think very carefully before you allow regular telnet access from machines in nonsecure networks. Apart from conventional eavesdropping (traces and network analyzers) it is quite common for crackers to replace the telnet command with a version that collects passwords, etc. Even if you use one-time passwords, we recommend also using encryption for such sessions, for example by means of the SNG secure IP tunnel.

16.1.1.14 14. You Can Subvert NTP to Attack Authentication Protocols

This is a rather exotic form of attack, but certainly one that is theoretically possible. Many authentication mechanisms (the Kerberos security system and some smart cards for example) rely on a degree of clock synchronization to ensure key freshness. If crackers could set back the clock of the authenticating system, they could replay an old authentication string as a way to break in. The solution to this is to avoid passing the network time protocol (NTP) across the firewall gateway, but instead to use some other source for clock synchronization (dialed line or radio).

16.1.1.15 15. Finger Discloses Too Much Information about Users

Finger is a protocol that is used to find out information about a machine and the users logged on to it. This is a good way for a cracker to find potential targets. The easy answer is to disable finger by blocking it with IP filters. However, finger can be a very useful facility, so you may use the proposed modification for giving contact information in a safe way.

16.1.1.16 16. Don't Trust RPC's Machine Name Field

This refers to the authentication area within Sun RPC messages. The information in here includes the calling machine name and user ID. Application code should not trust this information, since it can easily be overwritten by corrupted client code.

This is an application-level problem. Your only defence with SNG is to use IP filters to block RPC-based sessions.

16.1.1.17 17. Portmapper Can Call RPC Services for Its Caller

An RPC-based client can request portmapper to pass an RPC call directly to the target server, instead of just returning the port number of that server. This means that the server sees the request as having come from the local host. Your normal response to this would be to use filters to block the application at the firewall. However, with RPC-based services things are not so simple because you cannot predict the port numbers that portmapper will allocate. This means that you are led either to block *all* Sun RPC-based applications, or to use address filtering (which we have already said is not 100% effective. See 16.1.1.2, “2. Sequence Number Attacks Can Subvert Address-Based Authentication” on page 263).

16.1.1.18 18. NIS Can Often Be Persuaded to Give Out Password Files

NIS ("Yellow Pages") provides a mechanism for distributing user information, including /etc/passwd, amongst your systems. Needless to say, it is very dangerous to let NIS get out of your secure network. Unfortunately this is not so easy, since NIS is an RPC application and does not use predefined ports. At the very least, you should not use NIS on the firewall machine which is the most exposed machine in your network.

16.1.1.19 19. It Is Sometimes Possible to Direct Machines to Phony NIS Servers

This is a variation on the previous attack. It involves inserting a machine into the network that claims to be a backup for a real NIS server. As before, the response is not to run NIS on any exposed machines, and block traffic with an IP filter if possible.

16.1.1.20 20. It Is Hard to Revoke NFS Access

Access control in NFS works by means of the server providing the client with a file handle at mount time. It does this based on a list of authorized mounts (in file /etc/exports on an AIX system). However, NFS is a stateless protocol (the server retains no context about current mounts). This means that once a client has obtained a file handle, there is no need to go through the mount process again. In other words, anyone who can obtain the file handle can access the NFS-mounted disk using a hacked NFS client. The message from this is don't let NFS through your firewall! Fortunately it normally uses a well-known UDP port, 2049, which you can block with IP filters.

16.1.1.21 21. If Misconfigured, TFTP Will Hand Out /etc/passwd

Trivial FTP uses a configuration file to restrict the directories to which a client has access. Frequently, however, it is installed in an unconfigured way, allowing access to much more than you would like. You can stop it from routing through the firewall by blocking UDP port 69. This can pose a problem for supporting some network devices that use TFTP to load their operating code and configuration. Some routers keep the password used for online control, in clear, in the configuration file. So if you need to load, say, the gateway router to the Internet using TFTP, how do you prevent a cracker stealing the configuration file, logging in to the router, planting erroneous route definitions and then using them for a masquerade attack? The following two courses of action are possible answers to this:

1. Normally block TFTP and only let it through when you are loading the router (but you must not forget to lock the door when you finish!)
2. Prohibit online update for that particular router

16.1.1.22 22. Don't Make ftp's Home Directory Writable by ftp

When the client user ID is anonymous, the FTP daemon, ftpd, uses chroot so that it runs with a root directory of /u/ftp. This is a good security feature, since it means that no one can gain access to the real operating system files. However, if /u/ftp is writable by user ID ftp, any anonymous user will be able to create files there, such as .rhosts, which could permit a masquerade attack. When you set up anonymous FTP in AIX this directory restriction is automatically configured.

16.1.1.23 24. FSP Is Often Abused to Give Out Files to Those Who Should Not Have Them

FSP is a file transfer protocol that nobody uses legitimately. You should block it at your firewall by a deny filter for UDP port 21.

16.1.1.24 25. Be Careful about Interpreting WWW Format Information

Surfing the World Wide Web involves following a sequence of embedded pointers in documents. These pointers themselves can be dangerous, since they can specify a program to execute, host to retrieve data from, commands to issue, as well as the data itself. Someone could set up an apparently legitimate server with documents containing bogus commands that could hack into, or plant bombs in, the client's machine. Ideally a Web browser will prohibit pointers to commands that are obvious abuses of this sort. There are more subtle ways to do the same thing, though. For example some legitimate file formats may also include embedded commands (postscript for example). A browser cannot easily intercept attacks of this sort.

How can the SNG firewall help with this? One possibility is to use the telnet proxy service. This may operate with a very restricted shell, so the scope for damaging embedded commands is much reduced. Unfortunately this answer has some serious problems. The main one is performance; running a large number of X-windows servers will be a big drain on the firewall machine. There is also the extra delay introduced by running the X-protocol, and anyway we want to avoid running X on our firewall if possible. Finally there is the question of convenience; most people prefer to run applications like this on a personal computer. You need to assess the risk involved in order to come to a judgment about where and to what extent you allow WWW access in your secure network.

The particular security aspects of the Web are discussed in *Safe Surfing: How to Create a Secure World Wide Web Connection*, SG24-4564.

16.1.1.25 26. WWW Servers Should Be Careful about File Pointers

File pointers can also be used to subvert a WWW server, so crackers could use a legitimate user to move files on their behalf, without the legitimate user being aware of it. There is nothing the firewall can do about this, since it happens at the application level. When setting up a WWW server, however, you should be careful about the type of data it could potentially access if an attacker compromised it. This is another argument for insulating such servers by placing them outside the firewall.

16.1.1.26 27. Attackers Can Use FTP to Create gopher Control Information

This is a variation on the previous attack. The idea is that an attacker places file on the system via anonymous FTP, and then executes them remotely.

16.1.1.27 28. Poorly Written Query Scripts Pose a Danger to WWW Servers

As people add more nifty functions to their WWW servers, they become more and more complex. Often the applications are held together by shell scripts, written using the CGI interface, which may never have been rigorously designed and are therefore likely to provide security holes. As before, there is not much the firewall can do about this, but it is another argument for isolating the WWW server outside the secure network.

16.1.1.28 29. The MBone Can Be Used to Route through Some Firewalls

The MBone is the Internet's network of multi-cast routers. Multi-cast messages are encapsulated for transmission between MBone routers, and then forwarded to the correct UDP ports on the receiving nodes. The SNG firewall cannot, currently, provide any filtering for this, since it does not understand the encapsulation process.

16.1.1.29 30. An Attacker Anywhere on the Internet Can Probe for X11 Servers

X-Windows is often described as a back-to-front protocol. The terminal (X-station or workstation code) is the server, and the application is the client. Normally this is all well controlled; users log in to their application machine and start the session to their own X server. But in fact there is usually nothing to prevent anyone starting such a session and then having full authority to read the keyboard or screen, generate key presses, etc.

You can see from this that X-windows is a dangerous protocol to allow out through your firewall. Blocking it is not so easy; X11 normally uses a TCP port around 6000-6005 but it is dynamic, so there is a danger of blocking out other services (although you may want to block them anyway).

16.1.1.30 31. Don't Believe Port Numbers Supplied by Outside Machine

What this means is although the *normal* use for a port may be for a well-known server, a cracker can easily generate messages using that port (that is, a client using the normal server port). We have seen how in filters for TCP services we can prohibit this by using the tcp/ack filter feature to distinguish between the session initiator and responder (see 3.1.3, “An Introduction to TCP Packets” on page 33).

16.1.1.31 32. It Is All but Impossible to Permit All UDP Traffic through a Packet Filter Safely

Using tcp/ack (16.1.1.30, “31. Don't Believe Port Numbers Supplied by Outside Machine”) is effective for TCP services, but for UDP there is no equivalent indicator of which direction a session started from. When configuring filters for UDP traffic you should always be as specific as possible, both in terms of allowable ports and allowable addresses. In this way you are limiting the damage that can be done. Look at 6.16, “Network Management Sessions” on page 122 as an example.

16.1.1.32 33. A Tunnel Can Be Built on Top of almost Any Transport Mechanism

A *tunnel* is the technique of passing one protocol wrapped in another protocol, for example using a Telnet connection and having a program running at client and server ends that passes another protocol, say NFS, across it. Tunnelling can compromise a firewall because a banned protocol can be tunnelled within a permitted protocol and it may well be undetectable by IP filters. The saving grace is that constructing such a tunnel requires collusion between the cracker and someone inside your secure network, and it is therefore less likely to happen. The only way you are likely to detect a tunnel of this sort is if you do traffic analysis and see some unusual usage characteristics.

16.1.1.33 34. Firewalls Can't Block Attacks at Higher Levels of the Protocol Stack

This generalizes something that has arisen in several other modes of attack. A firewall can be used to deny a service, but if you permit a service to pass that is fundamentally insecure, there is nothing the firewall can do about it.

16.1.1.34 35. X11 Is Very Dangerous, Even When Passed through a Gateway

This does not apply to SNG, since it does not provide an authenticating X11 relay.

16.1.1.35 36. Network Monitoring Tools Can Be Dangerous on an Exposed Machine

Trace tools (such as the AIX `iptrace` command) can be used to collect passwords and IDs. If a cracker breaks into an exposed machine such as your firewall, he can use the routines (or the kernel routines they use) for such eavesdropping. There is no way currently to disable these functions in AIX

16.1.1.36 37. Be Careful about Pointing a Finger at a Subverted Machine

If you think you are being attacked, one thing you can do is use the `finger` command to try to find something out about the originator. Some crackers have corrupted finger servers, which will flood you with data or control codes. This can fill your file systems.

Appendix A. How to Get the Samples in This Book

There are two code packages described in this book, both of which are available using anonymous FTP. The code packages are:

tcp_relay. This is a TCP/IP relay program that can be used to provide a proxy service for the Network News Transfer Protocol (NNTP). It is described in 6.7, "NNTP: Network News Transfer Protocol" on page 101 and the source code is listed in Appendix D, "Sample NNTP Relay Program" on page 279.

LAID Logging, Alerting and Intruder Detection is a package of installation scripts and configuration files that:

1. Configure a logging environment, using syslog services.
2. Configure the AIX audit facility.
3. Configure Systems Monitor for AIX SLM and SIA agents to monitor the SNG system to check for problems that may indicate an attack or security exposure.

You can get these packages using anonymous ftp, as follows:

For Users Outside the IBM TCP/IP Network

- Connect to ftp.almaden.ibm.com via FTP
- Specify a user ID of anonymous
- Enter your E-mail ID as a password

You will find the samples in directory /redbooks/SG242577. You will also find a file named sg242577.README in the same directory, which includes instructions on how to download and unpack the samples. There are further README files within the samples giving detailed installation instructions.

For Users Within the IBM TCP/IP Network

- Connect to rsserver.itso.ral.ibm.com via FTP
- Specify a user ID of anonymous
- Enter your E-mail ID as a password

You will find the samples in directory /pub/SG242577. You will also find a file named sg242577.README in the same directory, which includes instructions on how to download and unpack the samples. There are further README files within the samples giving detailed installation instructions.

Appendix B. Summary of ICMP Messages Types

The following is a list of all the ICMP message types and the RFC in which they are defined, taken from the latest RFC of Assigned Numbers (RFC 1700).

Table 8. ICMP Message Types

Type	Name	Reference
0	Echo Reply	RFC792
1	Unassigned	JBP
2	Unassigned	JBP
3	Destination Unreachable	RFC792
4	Source Quench	RFC792
5	Redirect	RFC792
6	Alternate Host Address	JBP
7	Unassigned	JBP
8	Echo	RFC792
9	Router Advertisement	RFC1256
10	Router Selection	RFC1256
11	Time Exceeded	RFC792
12	Parameter Problem	RFC792
13	Timestamp	RFC792
14	Timestamp Reply	RFC792
15	Information Request	RFC792
16	Information Reply	RFC792
17	Address Mask Request	RFC950
18	Address Mask Reply	RFC950
19	Reserved (for Security)	Solo
20-29	Reserved (for Robustness Experiment)	ZSu
30	Traceroute	RFC1393
31	Datagram Conversion Error	RFC1475
32	Mobile Host Redirect	David Johnson
33	IPv6 Where-Are-You	Bill Simpson
34	IPv6 I-Am-Here	Bill Simpson
35	Mobile Registration Request	Bill Simpson
36	Mobile Registration Reply	Bill Simpson
37	Domain Name Request	RFC1788
38	Domain Name Reply	RFC1788
39-255	Reserved	JBP

B.1 Summary of ICMP Message Codes

The following is the list of the code numbers, taked also from the same RFC.

Table 9 (Page 1 of 3). ICMP Message Codes

Type	Name	Reference
0	Echo Reply Codes 0 No Code	RFC792
1	Unassigned	JBP
2	Unassigned	JBP
3	Destination Unreachable Codes 0 Net Unreachable 1 Host Unreachable 2 Protocol Unreachable	RFC792

Table 9 (Page 2 of 3). ICMP Message Codes

Type	Name	Reference
	3 Port Unreachable 4 Fragmentation Needed and Don't Fragment was Set 5 Source Route Failed 6 Destination Network Unknown 7 Destination Host Unknown 8 Source Host Isolated 9 Communication with Destination Network is Administratively Prohibited 10 Communication with Destination Host is Administratively Prohibited 11 Destination Network Unreachable for Type of Service 12 Destination Host Unreachable for Type of Service 13 Communication Administratively Prohibited by Filtering 14 Host Precedence Violation 15 Precedence Cutoff in Effect	
4	Source Quench Codes 0 No Code	RFC792
5	Redirect Codes 0 Redirect Datagram for the Network (or subnet) 1 Redirect Datagram for the Host 2 Redirect Datagram for the Type of Service and Network 3 Redirect Datagram for the Type of Service and Host	RFC792
6	Alternate Host Address Codes 0 Alternate Address for Host	JBP
7	Unassigned	JBP
8	Echo Codes 0 No Code	RFC792
9	Router Advertisement Codes 0 No Code	RFC1256
10	Router Selection Codes 0 No Code	RFC1256
11	Time Exceeded Codes 0 Time to Live exceeded in Transit 1 Fragment Reassembly Time Exceeded	RFC792
12	Parameter Problem Codes 0 Pointer indicates the error 1 Missing a Required Option 2 Bad Length	RFC792 RFC1108
13	Timestamp Codes 0 No Code	RFC792
14	Timestamp Reply Codes 0 No Code	RFC792
15	Information Request Codes 0 No Code	RFC792
16	Information Reply Codes 0 No Code	RFC792

Table 9 (Page 3 of 3). ICMP Message Codes

Type	Name	Reference
17	Address Mask Request Codes 0 No Code	RFC950
18	Address Mask Reply Codes 0 No Code	RFC950
19	Reserved (for Security)	Solo
20-29	Reserved (for Robustness Experiment)	ZSu
30	Traceroute	RFC1393
31	Datagram Conversion Error	RFC1475
32	Mobile Host Redirect	David Johnson
33	IPv6 Where-Are-You	Bill Simpson
34	IPv6 I-Am-Here	Bill Simpson
35	Mobile Registration Request	Bill Simpson
36	Mobile Registration Reply	Bill Simpson
37	Domain Name Request	RFC1788
38	Domain Name Reply Codes 0 No Code	RFC1788

Appendix C. Port Number Table

The following table shows some of the more common IP well-known port numbers. The official list of all assigned numbers is maintained in an RFC. The latest version is RFC1700, which obsoletes the previous list, RFC1340. RFC text is available on the World Wide Web at http://www.internic.net.

Table 10 (Page 1 of 2). Internet Port Numbers

Keyword	Decimal/protcol	Description
echo	7/tcp, 7/udp	
discard	9/tcp, 9/udp	Sink null
systat	11/tcp	Active users information
daytime	13/tcp, 13/udp	
qotd	17/tcp	Quote of the day
chargen	19/tcp, 19/udp	Character generator
ftp-data	20/tcp	File transfer protocol (data)
ftp	21/tcp	File transfer protocol (control)
telnet	23/tcp	Telnet
smtp	25/tcp	Simple mail transfer protocol
time	37/tcp, 37/udp	Time server
rlp	39/tcp, 39/udp	Resource location protocol
whois	43/tcp	Who is
domain	53/tcp, 53/udp	Domain nameserver
sql*net	66/tcp, 66/udp	Oracle SQL*NET
bootps	67/udp	Bootstrap protocol server
bootpc	68/udp	Bootstrap protocol client
tftp	69/udp	Trivial file transfer protocol
gopher	70/tcp	Gopher
finger	79/tcp	Finger information system
www-http	80/tcp	World Wide Web HTTP
kerberos	88/tcp	Kerberos security system
npp	92/tcp	Network printing protocol
hostname	101/tcp	NIC host name server
pop	109/tcp	Post office protocol
sunrpc	111/tcp, 111/udp	Sun remote procedure call
auth	113/tcp	Authentication service (ident service)
sftp	115/tcp	Simple file transfer protocol
uucp-path	117/tcp	UUCP path service
nntp	119/tcp	Network news transfer protocol
ntp	123/tcp, 123/udp	Network time protocol
cisco-xxx	130-132	Various Cisco-specific protocols
ingres-net	134/tcp	Ingres-net service
snmp	161/udp	SNMP gets and sets
snmp-trap	162/udp	SNMP traps
xdmcp	177/tcp	X display manager control protocol
irc	194/tcp	Internet relay chat
netware-ip	396/udp	Novell Netware over IP
exec	512/tcp	Remote command execution (rexec)
biff	512/udp	Inform users of new mail received
login	513/tcp	Remote login (rlogin)
who	513/udp	Who is logged on
shell	514/tcp	Remote command execution (rsh)
syslog	514/udp	UNIX logging port
printer	515/tcp	Print spooler

Table 10 (Page 2 of 2). Internet Port Numbers

Keyword	Decimal/protcol	Description
talk	517/udp	Interactive messaging
timed	525/udp	timeserver
uucp	540/tcp	UNIX-to-UNIX copy program
netviewdm1	729/tcp	IBM NetView Distribution Manager server/client
netviewdm1	730/tcp	IBM NetView Distribution Manager send
netviewdm1	731/tcp	IBM NetView Distribution Manager receive
SOCKS	1080/tcp	SOCKS application-level gateway
lotusnote	1352/tcp	Lotus Notes
x11	6000-6063/tcp	X Windows system

Appendix D. Sample NNTP Relay Program

This program is based on a sample in *Actually Useful Internet Techniques* by Larry Hughes (published by New Riders, ISBN 1-56205-508-9).

```
/* ====================================================================
 *
 * Program: tcp_relay.c
 *
 * Author : Adrian Fabio Setton <setton@vnet.ibm.com>
 *          IBM Argentina
 *
 * Tweaked By: Andreas Siegert <afx@ibm.de>
 *             IBM Germany
 *
 * Derived from: passtru.c, with permission.
 *        Author : Larry J. Hughes, Jr. <larry@bodhisoft.com>
 *                 "Actually Useful Internet Security Techniques"
 *                 New Riders Publishing, ISBN 1-56205-508-9
 * Thanks Larry !!!
 *
 * Purpose :
 *     Relays a connection thru the SNG avoiding IP forwarding
 *     It uses a configuration file in order to define which connections to
 *     relay.
 *     The format of the /etc/tcp_relay.cfg file is:
 *        inAddress inPort outAddress outPort.
 *        inAddress is the IP address that establishes the connection
 *        inPort    in the Port that receives the connection on the SNG.
 *     Lines starting with a # and empty lines are ignored
 *
 * Example of the Config File for News
 *        150.53.104.12 119  9.24.104.241 119
 *        9.24.104.241  119 150.53.104.12 119
 * Usage  :  configure the following line in /etc/inetd.conf for NNTP (news):
 *           nntp stream tcp nowait nobody /usr/local/bin/tcp_relay tcp_relay
 *
 * Compile with
 *        make tcp_relay
 * ==================================================================== */

/* ====================================================================
 * Includes
 * ==================================================================== */
#include <sys/types.h>
#include <sys/time.h>
#include <stdio.h>
#include <stdlib.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <sys/socket.h>
#include <sys/select.h>
#include <sys/syslog.h>

typedef struct TABLE {
        ulong   inaddress;
        int     inport;
        ulong   outaddress;
        int     outport;
} TABLE ;
```

Figure 244 (Part 1 of 5). Relay Program for NNTP

```
/* ============================================================================
 * Defines
 * ============================================================================ */
#define TRUE    1
#define FALSE   0
#define MAX_ENTRYS 1024
#define CFGFILE "/etc/tcp_relay.cfg"

/* ============================================================================
 * Global Variables      (My apologies ...)
 * ============================================================================ */
int     nTableEntrys;
TABLE   Table[MAX_ENTRYS];
int  serverSocket=0;

/* ============================================================================
 * Prototypes
 * ============================================================================ */
int     main(int argc, char *argv[]);
int     RelayConnection(int client, int server);
int     NetWrite(int socket, char *buffer, int length);
void    ReadTable(char *szFileName);
void    FatalError(int n, char *s);

/* ============================================================================
 * Main
 * ============================================================================ */
main(int argc, char *argv[])
{
        struct sockaddr_in clientAddress;
        struct sockaddr_in firewallPort;
        struct sockaddr_in serverAddress;
        int clientLength=sizeof(clientAddress);
        int firewallLength=sizeof(firewallPort);
        int i, nIndex;
        int one = 1;
        int Found=FALSE;
        char connbuff[1024], errbuff[1024];
        char a1[24],a2[24];

        /* set up syslog */
        openlog ("tcp_relay",LOG_PID | LOG_ODELAY | LOG_CONS,LOG_AUTH);

        ReadTable(CFGFILE);

        if(getpeername(0,(struct sockaddr*)&clientAddress, &clie ntLength)<0) {
                FatalError(2, "getpeername error");
        }

        if(getsockname(0,(struct sockaddr*)&firewallPort, &firewallLength)<0) {
                FatalError(3, "getsockname error");
        }

        for (i=0; i<nTableEntrys && !Found; i++) {
                if ( (clientAddress.sin_addr.s_addr==Table[i].inaddress) &&
                        (firewallPort.sin_port==Table[i].inport)
                ) {
                        Found=TRUE;
                        nIndex=i;
                }
        }

        strcpy(a1,inet_ntoa(clientAddress.sin_addr.s_addr));
        if (!Found) {
                sprintf (errbuff,"%s:%i not authorized",
                         a1,firewallPort.sin_port);

                FatalError(4, errbuff);
        }
```

Figure 244 (Part 2 of 5). Relay Program for NNTP

```
        /* Create socket for connection to remote server */
        if ((serverSocket = socket(AF_INET, SOCK_STREAM, 0)) == -1) {
                FatalError(5, "socket error");
        }


        /* Set keepalive on client and server sockets so we can detect
           half-open client connections */
        if (setsockopt(0, SOL_SOCKET, SO_KEEPALIVE,
                (char *)&one, sizeof(one)) == -1) {
                FatalError(6, "keepalive error");
        }
        if (setsockopt(serverSocket, SOL_SOCKET, SO_KEEPALIVE,
                       (char *)&one, sizeof(one)) == -1) {
                FatalError(7, "keepalive error");
        }


        /* Build the server address structure */
        memset((char *)&serverAddress, '\0', sizeof(serverAddress));
        memcpy( (char *)&serverAddress.sin_addr,
                (char *)&(Table[nIndex].outaddress),
                sizeof(Table[nIndex].outaddress));
        serverAddress.sin_port   = htons(Table[nIndex].outport);
        serverAddress.sin_family = AF_INET;


        /* Connect to the remote server */
        strcpy(a2,inet_ntoa(Table[nIndex].outaddress));
        if (connect(serverSocket, (struct sockaddr *)&serverAddress,
                sizeof(struct sockaddr)) == -1) {
                sprintf (errbuff,"%s:%i connection error",
                         a2,Table[nIndex].outport);
                FatalError(8, errbuff);
        }


        /* Relay the connection */

        syslog(LOG_DEBUG, "relaying %s:%i to %s:%i",
           a1,Table[nIndex].inport,
           a2,Table[nIndex].outport);
        RelayConnection(0, serverSocket);
        close(0);
        close(serverSocket);
        syslog(LOG_DEBUG, "terminating %s:%i to %s:%i",
           a1,Table[nIndex].inport,
           a2,Table[nIndex].outport);
        closelog();
        exit(0);
}

/* ====================================================================
 * Relay Connection
 * ==================================================================== */
int RelayConnection(int client, int server)
{
  int    numSelected;
  int    rBytes, wBytes;
  char   buffer[1024];
  fd_set ibits;
```

Figure 244 (Part 3 of 5). Relay Program for NNTP

```
  /*
   * Read bytes from client and send to server, and vice versa.
   * Do this until one side goes away or an error is detected.
   */
  FD_ZERO(&ibits);
  while (TRUE)
  {
    FD_SET(client, &ibits);
    FD_SET(server, &ibits);

    numSelected =
      select(16, &ibits, (fd_set *)0, (fd_set *)0, (struct timeval *)0);

    if (numSelected == -1)
    {
          FatalError(9, "select error");
          break;
    }
    /* client -> server */
    else if (FD_ISSET(client, &ibits))
    {
      rBytes = read(client, buffer, sizeof(buffer));
      if (rBytes <= 0) break;
      wBytes = NetWrite(server, buffer, rBytes);
      if (wBytes != rBytes) break;
    }

    /* server -> client */
    else if (FD_ISSET(server, &ibits))
    {
      rBytes = read(server, buffer, sizeof(buffer));
      if (rBytes <= 0) break;
      wBytes = NetWrite(client, buffer, rBytes);
      if (wBytes != rBytes) break;
    }

  }
}

/* ====================================================================
 * Network Write
 * ==================================================================== */
int NetWrite(int socket, char *buffer, int length)
{
  int numToWrite, numWritten;

  /*
   * Write the entire buffer or die trying.  Might take several attempts.
   */
  numToWrite = length;
  do
  {
    numWritten = write(socket, buffer, numToWrite);
    if (numWritten == -1)
    {
          FatalError(10, "write error");
    }
    buffer += numWritten;
    numToWrite -= numWritten;
  } while (numToWrite > 0);
  return(length);
}
```

Figure 244 (Part 4 of 5). Relay Program for NNTP

```
/* ========================================================================
 * Fatal Error
 * ======================================================================== */
void    FatalError(int n, char *s)
{
        syslog(LOG_ERR, "[%d] %s", n, s);
        close(0);
        if (serverSocket!=0) {
             close(serverSocket);
        }
        closelog();
        exit(n);
}

/* ========================================================================
 * Read Table
 * ======================================================================== */
void    ReadTable(char *szFileName)
{
        FILE *fp;
        char szBuffer[255];
        char szInAddress[255],  szInPort[255];
        char szOutAddress[255], szOutPort[255];
        char errbuff[1024];
        int i, nRet, lineno;

        fp=fopen(szFileName, "r");
        if (!fp) {
                sprintf(errbuff,"Error opening %s",szFileName);
                FatalError(11, errbuff);
        }
        i=0;
        lineno=0;
        while (fgets((szBuffer), sizeof(szBuffer)-1, fp) && i<MAX_ENTRYS) {
                lineno++;

                if ((szBuffer[0]=='#') || (strlen(szBuffer)<=1)) {
                        /* empty or comment line */
                        continue ;
                }
                nRet=sscanf(szBuffer, "%s%s%s%s", szInAddress, szInPort,
                                                    szOutAddress, szOutPort);
                if (nRet!=4) {
                        fclose(fp);
                        sprintf(errbuff,"Error reading line %i of %s",lineno,szFileName);
                        FatalError(12, errbuff);
                }

                Table[i].inaddress=inet_addr(szInAddress);
                Table[i].inport=atoi(szInPort);
                Table[i].outaddress=inet_addr(szOutAddress);
                Table[i].outport=atoi(szOutPort);

                if (Table[i].inaddress==-1 || Table[i].outaddress==-1) {
                        fclose(fp);
                        sprintf (errbuff,"Config file format error at line %i: %s",lineno,szBuffer);
                        FatalError(13, errbuff);
                }
                i++;
        }
        nTableEntrys=i;
        fclose(fp);
}
```

Figure 244 (Part 5 of 5). Relay Program for NNTP

Appendix E. IBM Firewall Related Products

There are several non-IBM products that complement the functions of the IBM Firewall. These products have been tested and work with the IBM Firewall.

For a current list of IBM Firewall related products, you should check:

`http://www.software.ibm.com/enetwork/firewall/`

. Click on Products.

We will briefly describe the main features of the products. Most of the information has been taken from the vendors' Web sites. The following is the list of IBM Firewall related products:

1. MIMEsweeper from Integralis (*www.integralis.com*)

   MIMEsweeper does content security checks of mail and Web access. It has two main components: MAILsweeper and WEBsweeper. It can:

   - Manage junk e-mail
   - Block URLs
   - Protect confidential information
   - Block Java applets and JavaScript
   - Sanitize cookies
   - Add legal disclaimers to mail
   - Quarantine dubious files

   MIMEsweeper runs under Windows NT.

2. S/KEY from Bellcore (*www.bellcore.com/security/skey.html*)

   S/KEY is a software authentication package that you can use with the IBM Firewall user-supplied authentication API.

   S/KEY uses a client/server model of authentication. The server generates a challenge response to the client's logical request. The client then generates the appropriate one-time password. The client calculates this password by using the server's challenge and the user's secret pass phrase as inputs to the S/Key system's one-time password algorithm. The client sends the generated one-time password back to the server, which verifies it and logs the user into the system.

   This system has the advantage over similar ones that does not require any special hardware.

   IBM Firewall provides a sample of user-supplied authentication code. It is supplied as three files (*makefile.ex, fwuserpt.c, fwuserau.c*) in the /usr/llp/FW/sample directory. See *Firewall for AIX Reference*, SC31-8418 for details.

3. SecureID card from Security Dynamics (*http://www.securitydynamics.com/*)

   The SecurID token provides a one step process to positively identify network and system users and prevent unauthorized access. It can be used, for example, to allow/deny access to the proxy server.

   When the user wants to access the system, he/she logs in, and then is prompted to enter the one-time password read off the SecureID token. This

response is then authenticated by a SecureID server running on the secure network. This password is updated every 60 seconds, so that even if a hacker can catch it, it will not do him/her any good.

The ACE/Server client code must be installed in the Firewall and the ACE/Server server must be installed in some machine inside the secure network.

4. SecureNet Key products from Axent (*http://www.axent.com/*)

   The SecureNet Key (SNK) is a one time password solution for strong authentication. It is similar to the SecureID card from Security Dynamics. Although throughout this redbook and the IBM Firewall publications this card is mentioned as SecureNet, Axent has lately changed the name to Defender.

   The SecureNet Key (SNK) is a device like a small calculator. It implements a challenge/response authentication scheme, using secret (shared) key cryptography.

   Before the SNK key can be used, it must be loaded with the secret key, by the security administrator. A copy of this secret key is held in the firewall's key database. The SNK is also protected by a PIN code, which may be set by the user or by the administrator.

   When the user wishes to access the system, he/she will log in. The firewall will then generate a random challenge number. The user enters their PIN and the challenge number into the SNK card, which encrypts the challenge number with a varient of DES encryption, and displays the result. The user then enters this response. The firewall also performs the same encryption (without a PIN), and compares the results. If they match, the computer allows access.

   SNK can be used, for example, to allow/deny access to the proxy server.

   There is a new version of Defender that does not require the device; it is a software only version.

5. Stalker from Haystack Labs

   Stalker is software designed to detect and respond to UNIX system misuse. By comparing logs of system activities against its database of ways to break into UNIX systems, Stalker is able to detect security breaches.

   Stalker can manage multiple and heterogeneous UNIX stations from a single server, running 24 hours a day, in automated unattended mode. When tampering occurs, alarms are sent via e-mail, or to printed reports detailing who did what, when, where, and how.

   Stalker for IBM Firewall is a monitoring package for the firewall that watches the firewall for network and operating system attacks and malicious manipulation of security files and databases.

   See 15.3, “Stalker” on page 261 for details.

6. AutoSOCKS client code from Aventail Corporation (*www.aventail.com*)

   AutoSOCKS is a Windows program that socksifies the applications running on the client. The programs can access the IBM Firewall SOCKS server without any modification.

   We did some tests with this product. See E.1, “Aventail AutoSOCKS” on page 288 for details.

7. Telemate.net from Telemate Software (*http://www.telemate.net/*)

Telemate is a reporting tool. It will read the IBM Firewall log and produce reports. It runs in Windows 95 or NT.

Telemate can answer questions as:

- How is my bandwidth being used?
- Who is hitting questionable sites?
- What sites are individual users hitting?
- Who is generating the most traffic?
- How much should I bill for use?

We should also mention two IBM products that also complement the IBM Firewall functions:

- HACMP (High-Availability Cluster Multi-Processing) (*http://www.rs6000.ibm.com/software/Apps/hacmp/*)

  HACMP for AIX is a control application that can link up to eight RS/6000 servers or SP nodes into highly available clusters. With the enhanced scalability feature, up to 16 SP nodes can be linked. Clustering servers or nodes enables parallel access to their data, which can help provide the redundancy and fault resilience required for business critical applications.

  Using HACMP with two or more IBM Firewalls will enhance the reliability of the systems.

  IBMers will find more information on this subject in this intranet Web Site: *http://hawww.ak.munich.ibm.com/HACMP/HA-FW/HA-FW-2.html*

- Interactive Network Dispatcher (*http://www.ics.raleigh.ibm.com/netdispatch/*)

  Interactive Network Dispatcher (IND) is a software product that performs load-balancing and IP traffic management across multiple IBM Firewalls.

  It enables multiple IBM Firewalls to efficiently function as a single system, allowing customers to distribute traffic to servers located anywhere in the world, greatly increasing system availability and responsiveness.

  IND can co-exist in the same machine with IBM Firewall.

  In order to be used with IBM Firewall 3.1, the Network Dispatcher needs access to port 10003 on the firewall. To set up the firewall correctly, do the following:

  1. Define two network objects on the firewall with the same IP address. The names must be different. This will allow the firewall to talk to itself, which would not be allowed if the network object names were the same.
  2. Define a connection with the two network objects just created. Use one as the source and the other as the destination.
  3. Within the connection, define a service with the following rules:
  4. Rule 1 is a permit rule:

     ```
     source port = 10003, Destination port >1024, route = local, direction = both, interface = non-secure.
     ```

  5. Rule 2 is a permit rule:

     ```
     source port > 1024, Destination port =10003, route = local, direction = both, interface = non-secure.
     ```

  With the connection defined, the two rules will be built, which will allow Network Dispatcher to talk to itself on port 10003.

E.1 Aventail AutoSOCKS

Aventail AutoSOCKS is a 16 and 32-bit Windows application for firewall traversal. It enables any TCP/IP application to traverse existing SOCKS V4 or V5-based firewalls.

AutoSOCKS makes Windows-based TCP/IP applications SOCKS-compliant. By implementing AutoSOCKS with a corporation's existing TCP/IP applications, clients can communicate through the SOCKS-enabled firewalls.

A diagram of the flow for TCP/IP services in a Windows client is shown in Figure 245.

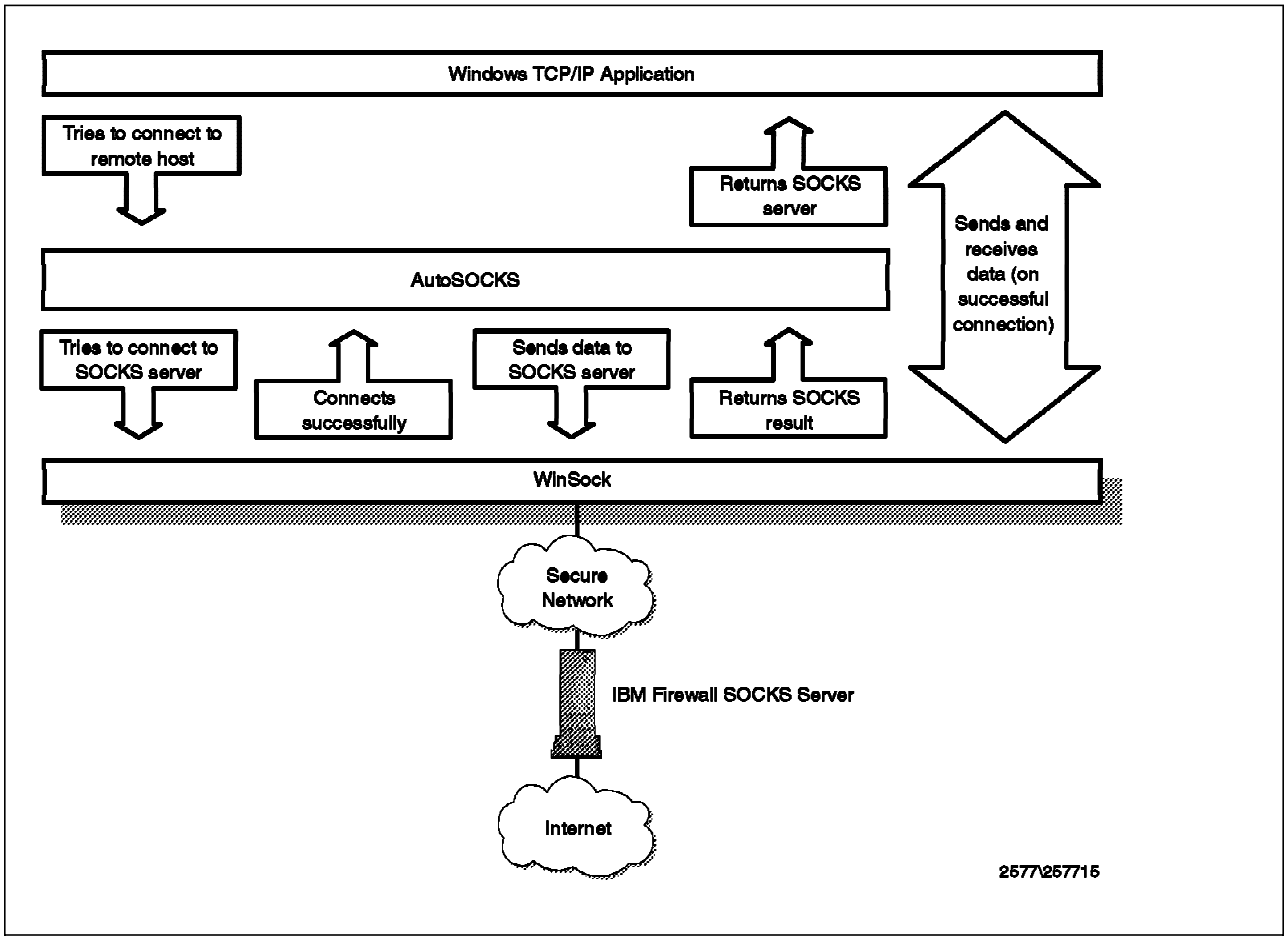


Figure 245. Aventail AutoSOCKS Flow

AutoSOCKS works by intercepting WinSock communication requests issued by applications and then processes the requests based on a set of rules. These rules govern whether or not a WinSock request is redirected through a SOCKS server. While communications within the local network may proceed unchanged, all communication with external networks can be redirected through one or many SOCKS servers.

E.1.1 Major Features

The following is a list of AutoSOCKS features provided by the vendor:

- Supports all Windows platforms
  - Windows 3.x, Windows for Workgroups 3.11, Windows 95 and Windows NT
- Supports multiple authentication and encryption standards
  - Simple username/password
  - Challenge-Handshake Authentication Protocol (CHAP)
  - Kerberos and SSL (optional)
- End-user transparency
  - Integrates with existing desktop applications and TCP/IP stacks
  - Seamlessly routes connections from Windows applications to external networks through a SOCKS server
  - Transparently negotiates authentication and encryption with SOCKS V5 server
  - Support for multiple SOCKS servers
  - No modification to Windows system components and environment
- Standards support
  - Interoperable with 16- and 32-bit WinSock 1.1 applications
  - Supports all Windows, Windows 95 and Windows NT TCP/IP stacks
  - Supports publicly available SOCKS V4 and V5 standards
- Network administration
  - User interface provides a single configuration point for all WinSock client applications
  - Online documentation and help
  - Logging tool for troubleshooting
  - Ping and traceroute for monitoring network connections

E.1.2 Our Tests

Our installion of AutoSOCKS went pretty smoothly. We used the evaluation copy that can be downloaded from:

`www.aventail.com`

There is an Installation Wizard that comes with the product, which guides you during the product setup. There is also a configuration tool. We had to only specify a few parameters, for example, the firewall IP address (or name) and the version of socks.

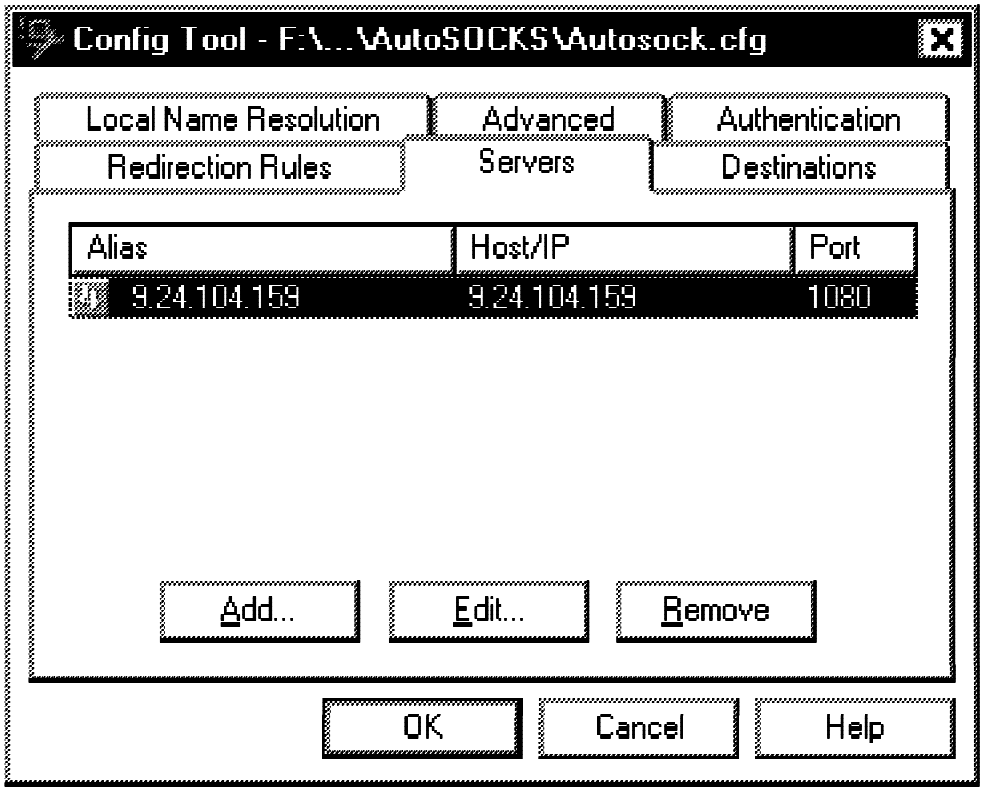


Figure 246. Defining the SOCKS Server to Aventail AutoSOCKS

AutoSOCKS can check which is the SOCKS version of the firewall.

Next we had to specify the IP address of our subnet; we were in the secure network 9.24.104.0.

We also had to specify the domain name of our secure network. AutoSOCKS will redirect any traffic that is not intended for the network to the SOCKS server in the firewall.

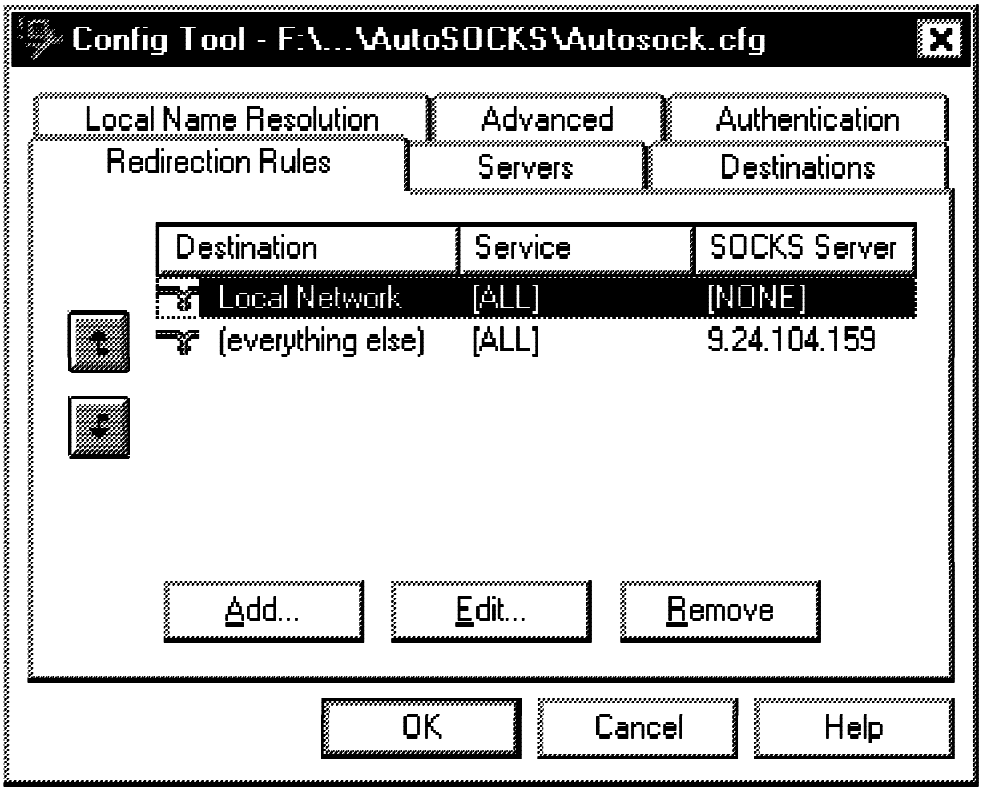


Figure 247. Defining the Subnet to Aventail AutoSOCKS

We only tested Netscape, telnet, FTP, RealAudio and PointCast.

AutoSOCKS keeps a log of all the applications that use its service. A sample log is shown in Figure 248.

```
11/11/97 10:23:26: (32 bit) Info:  Cannot provide DNS proxying.  One or more servers is version 4 (9.24.104.159)
11/11/97 10:24:46: (32 bit) Info:  Socket 38: Matched against a redirection rule, destination (everything else)
11/11/97 10:24:46: (32 bit) Info:  Socket 38: Redirecting to SOCKS server 9.24.104.159
11/11/97 10:24:46: (32 bit) Info:  Socket 38: Redirecting connection!
11/11/97 10:25:10: (32 bit) Info:  Socket 43: Matched against a redirection rule, destination (everything else)
11/11/97 10:25:11: (32 bit) Info:  Socket 43: Redirecting to SOCKS server 9.24.104.159
11/11/97 10:25:11: (32 bit) Info:  Socket 43: Redirecting connection!
11/11/97 10:25:11: (32 bit) Info:  Socket 44: Matched against a redirection rule, destination (everything else)
11/11/97 10:25:11: (32 bit) Info:  Socket 44: Redirecting to SOCKS server 9.24.104.159
11/11/97 10:25:11: (32 bit) Info:  Socket 44: Redirecting connection!
```

Figure 248. AutoSOCKS log

There are three options on what type of records to log:

- Errors
- Warnings
- Information

We turned on *information* in order to get the records shown in Figure 248. These records represent our connections (telnet or FTP) to 9.24.105.249 or using Netscape to the non-secure network from subnet 9.24.104.0 (see Figure 249 on page 292). They were redirected to the firewall SOCKS server (9.24.104.159).

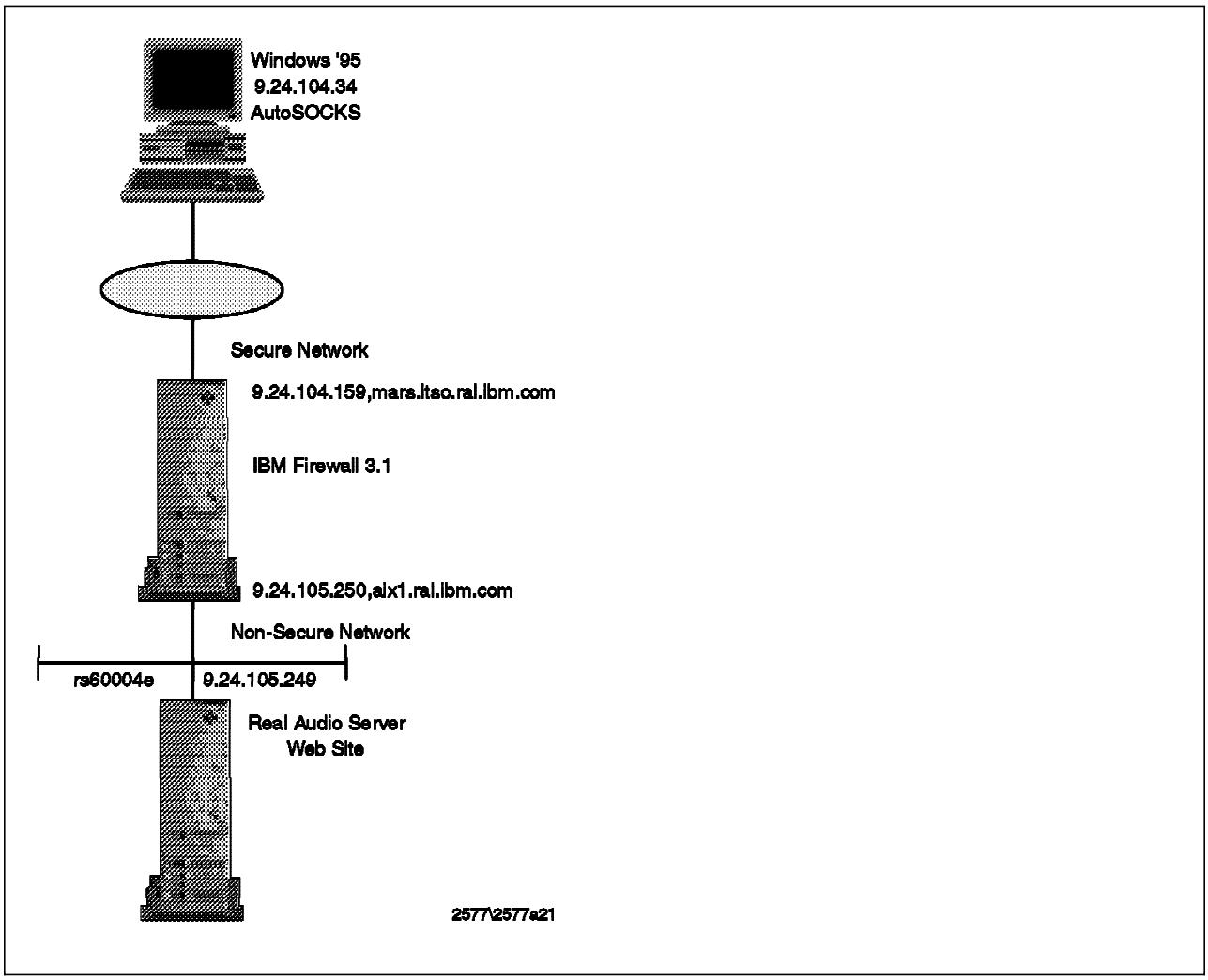


Figure 249. Test Scenario for AutoSOCKS

Appendix F. Listing the IBM Firewall Rules

The following awk program was developed to list the rules defined in a IBM Firewall:

```
# Prints out Services lines
awk -F \] '
BEGIN { rulefile="/etc/security/fwrules.cfg";}
 {printf "%-32s %-s\n",$2,$3; # $3 up to 69 chars
 nr=split($8,rr," ");
 for (i=1; i<=nr; i++)
  {
    split(rr[i],rn,";");
    do
   {
     getline < rulefile;
   }
    while ($1 != rn[1]);
    printf "  >%c<\t%-40s %-s\n", rn[2] , $2, $3;  # $3 up to 53 chars
    if ($6 == "any") {sp=$6 " " $7}
    if ($6 == "eq")  {sp="="    $7}
    if ($6 == "gt")  {sp=">"    $7}
    if ($8 == "any") {dp=$8 " " $9}
    if ($8 == "eq")  {dp="="    $9}
    if ($8 == "gt")  {dp=">"    $9}
    printf "\t%-7s%-8s%-6s%-6s%-11s%-6s%-9s%-4s%-3s\n",$4,$5,sp,dp,$10,$11,$12,$13,$14;
    close(rulefile);
  }
 print "";
 }
' /etc/security/fwservices.cfg
```

This awk program reads each line of /etc/security/fwservices.cfg (that's the standard input to the awk program), which holds the service definitions, and then, for each of the rule numbers in field 8, it opens rule file /etc/security/fwrules.cfg, and searches for that rule.

A sample of the output follows:

```
All non-secure                 Deny all traffic across non-secure interface
  >o< All - deny non-secure                    deny All non-secure
      deny   all     any 0 any 0 non-secure both  both     l=y f=y

All open locally               all open locally
  >i< All open locally                         all open locally
      permit all     any 0 any 0 both       local both     l=n f=y
  >o< All open locally                         all open locally
      permit all     any 0 any 0 both       local both     l=n f=y

All permit, in one direction   Permit all (for debugging purposes only)
  >i< All - permit                             permit All
      permit all     any 0 any 0 both       both  both     l=y f=y

All permit                     Permit all (for debugging purposes only)
  >i< All - permit                             permit All
      permit all     any 0 any 0 both       both  both     l=y f=y
  >o< All - permit                             permit All
      permit all     any 0 any 0 both       both  both     l=y f=y

All secure                     Deny all traffic across secure interface (in case of security breach)
  >i< All - deny secure                        deny All secure
      deny   all     any 0 any 0 secure     both  both     l=y f=y

All shutdown                   Deny all packets (shutdown or debug)
  >i< All - deny any                           deny All
      deny   all     any 0 any 0 both       both  both     l=y f=y
  >o< All - deny any                           deny All
```

```
      deny   all     any 0 any 0 both       both  both     l=y f=y

Anti Spoofing                   Deny inbound non-secure packets with secure source addresses
  >i< Anti Spoofing                             deny in any port non-secure
      deny   all     any 0 any 0 non-secure both  inbound  l=y f=h

Broadcasts                      Deny broadcast messages to non-secure interface
  >i< Broadcast                                 deny UDP All non-secure
      deny   udp     any 0 any 0 non-secure both  both     l=n f=h

Config Client non-secure        Permit use of config client from non-secure network
  >i< Remote Config in non-secure               TCP in port 1014 non-secure
      permit tcp     >1023 =1014 non-secure local inbound  l=n f=y
  >o< Remote Config Ack out non-secure          TCP/ACK out port 1014 non-secure
      permit tcp/ack =1014 >1023 non-secure local outbound l=n f=y

Config Client secure            Permit use of config client from secure network
  >i< Remote Config in secure                   TCP in port 1014 secure
      permit tcp     >1023 =1014 secure     local inbound  l=n f=y
  >o< Remote Config Ack out secure              TCP/ACK out port 1014 secure
      permit tcp/ack =1014 >1023 secure     local outbound l=n f=y

CU-SeeMe                        CU-SeeMe Video on default ports 7649 and 7648
  >i< Permit CU-SeeMe TCP Session               Permit CU-SeeMe from 7649 to 7648
      permit tcp     =7649 =7648 both       route both     l=n f=y
  >i< Permit CU-SeeMe TCP/ACK                   Permit CU-SeeMe TCP/ACK from 7648 to 7649
      permit tcp/ack =7648 =7649 both       route both     l=n f=y
  >i< Permit CU-SeeMe UDP                       Permit CU-SeeMe UDP Video Transfer
      permit udp     =7648 =7648 both       route both     l=n f=y
  >o< Permit CU-SeeMe TCP Session               Permit CU-SeeMe from 7649 to 7648
      permit tcp     =7649 =7648 both       route both     l=n f=y
  >o< Permit CU-SeeMe TCP/ACK                   Permit CU-SeeMe TCP/ACK from 7648 to 7649
      permit tcp/ack =7648 =7649 both       route both     l=n f=y
  >o< Permit CU-SeeMe UDP                       Permit CU-SeeMe UDP Video Transfer
      permit udp     =7648 =7648 both       route both     l=n f=y

Deny log entries                ospf,udp 137,138,520
  >i< Deny OSPF                                 Deny OSPF
      deny   ospf    any 0 any 0 both       both  both     l=n f=y
  >i< Deny udp 137                              Deny udp 137
      deny   udp     =137  =137  both       both  both     l=n f=y
  >i< Deny udp 138                              Deny udp 138
      deny   udp     =138  =138  both       both  both     l=n f=y
  >i< Deny udp 520                              Deny udp 520
      deny   udp     =520  =520  both       both  both     l=n f=y

DNS queries                     Permit DNS queries
  >i< DNS Server queries                        UDP port 53
      permit udp     =53   =53   both       local both     l=n f=y
  >i< DNS Client queries                        UDP port 53
      permit udp     >1023 =53   both       local both     l=n f=y
  >i< DNS Replies                               UDP port 53
      permit udp     =53   >1023 both       local both     l=n f=y

DNS transfers                   Permit DNS zone transfers (for secondary name servers)
  >i< DNS - Long Server Queries                 TCP port 53
      permit tcp     =53   =53   both       local both     l=n f=y
  >i< DNS - Long Client Queries                 TCP port 53
      permit tcp     >1023 =53   both       local both     l=n f=y
  >i< DNS - Long Replies                        TCP/ACK port 53
      permit tcp/ack =53   >1023 both       local both     l=n f=y

EFM config non-secure           Permit use of EFM out non-secure
  >i< EFM config out non-secure                 TCP out to port 1024 non-secure
      permit tcp     >1025 =1024 non-secure local outbound l=n f=y
  >o< EFM config ACK in non-secure              TCP/ACK in from port 1024 non-secure
      permit tcp/ack =1024 >1025 non-secure local inbound  l=n f=y

FTP direct out                  Permit FTP outbound from secure network to non-secure network
  >i< FTP Control 1/2                           TCP in port 21 secure
      permit tcp     >1023 =21   secure     route inbound  l=n f=y
  >i< FTP Control 2/2                           TCP out port 21 non-secure
      permit tcp     >1023 =21   non-secure route outbound l=n f=y
  >o< FTP Control Ack 1/2                       TCP/ACK in port 21 non-secure
      permit tcp/ack =21   >1023 non-secure route inbound  l=n f=y
```

```
  >o< FTP Control Ack 2/2                    TCP/ACK out port 21 secure
      permit tcp/ack =21   >1023 secure     route outbound l=n f=y
  >o< FTP Data in 20                         TCP in port 20 non-secure
      permit tcp     =20   >1023 non-secure route inbound  l=n f=y
  >o< FTP Data out 20                        TCP out port 20 secure
      permit tcp     =20   >1023 secure     route outbound l=n f=y
  >i< FTP Data Ack in 20                     TCP/ACK in port 20 secure
      permit tcp/ack >1023 =20   secure     route inbound  l=n f=y
  >i< FTP Data Ack out 20                    TCP/ACK out port 20 non-secure
      permit tcp/ack >1023 =20   non-secure route outbound l=n f=y
  >i< FTP Data in 1023+                      TCP in port 1023+ secure
      permit tcp     >1023 >1023 secure     route inbound  l=n f=y
  >i< FTP Data out 1023+                     TCP out port 1023+ non-secure
      permit tcp     >1023 >1023 non-secure route outbound l=n f=y
  >o< Proxy FTP Data Ack in non-secure       TCP/ACK port 1023+
      permit tcp/ack >1023 >1023 non-secure route inbound  l=n f=y
  >o< FTP Data Ack out 1023+                 TCP/ACK out port 1023+ secure
      permit tcp/ack >1023 >1023 secure     route outbound l=n f=y

FTP proxy in 1/2               Permit FTP inbound from non-secure network to firewall
  >i< Proxy FTP Control in non-secure 1/2    TCP in port 21 non-secure
      permit tcp     >1023 =21   non-secure local inbound  l=n f=y
  >o< Proxy FTP Control Ack out non-secure 1/2 TCP/ACK port 21
      permit tcp/ack =21   >1023 non-secure local outbound l=n f=y
  >o< Proxy FTP Data out non-secure 1/2      TCP port 20
      permit tcp     =20   >1023 non-secure local outbound l=n f=y
  >i< Proxy FTP Data Ack in non-secure 1/2   TCP/ACK port 20
      permit tcp/ack >1023 =20   non-secure local inbound  l=n f=y
  >i< Proxy FTP Data in non-secure 1/2       TCP port 1023+
      permit tcp     >1023 >1023 non-secure local inbound  l=n f=y
  >o< Proxy FTP Data Ack out non-secure 1/2  TCP/ACK port 1023+
      permit tcp/ack >1023 >1023 non-secure local outbound l=n f=y

FTP proxy in 2/2               Permit FTP inbound from firewall to secure network
  >i< Proxy FTP Control out secure 1/2       TCP port 21
      permit tcp     >1023 =21   secure     local outbound l=n f=y
  >o< Proxy FTP Control Ack in secure 2/2    TCP/ACK port 21
      permit tcp/ack =21   >1023 secure     local inbound  l=n f=y
  >o< Proxy FTP Data in secure 2/2           TCP port 20
      permit tcp     =20   >1023 secure     local inbound  l=n f=y
  >i< Proxy FTP Data Ack out secure 2/2      TCP/ACK port 20
      permit tcp/ack >1023 =20   secure     local outbound l=n f=y

FTP proxy out 1/2              Permit FTP outbound from secure network to firewall
  >i< Proxy FTP Control in secure 1/2        TCP port 21
      permit tcp     >1023 =21   secure     local inbound  l=n f=y
  >o< Proxy FTP Control Ack out secure 1/2   TCP/ACK port 21
      permit tcp/ack =21   >1023 secure     local outbound l=n f=y
  >o< Proxy FTP Data out secure 1/2          TCP port 20
      permit tcp     =20   >1023 secure     local outbound l=n f=y
  >i< Proxy FTP Data Ack in secure 1/2       TCP/ACK port 20
      permit tcp/ack >1023 =20   secure     local inbound  l=n f=y
  >i< Proxy FTP Data in secure 1/2           TCP in port 1023+
      permit tcp     >1023 >1023 secure     local inbound  l=n f=y
  >o< Proxy FTP Data Ack out secure 1/2      TCP/ACK port 1023+
      permit tcp/ack >1023 >1023 secure     local outbound l=n f=y

FTP proxy out 2/2              Permit FTP outbound from firewall to non-secure network
  >i< Proxy FTP Control out secure 2/2       TCP port 21
      permit tcp     >1023 =21   non-secure local outbound l=n f=y
  >o< Proxy FTP Control Ack in non-secure 2/2  TCP/ACK port 21
      permit tcp/ack =21   >1023 non-secure local inbound  l=n f=y
  >o< Proxy FTP Data in non-secure 2/2       TCP port 20
      permit tcp     =20   >1023 non-secure local inbound  l=n f=y
  >i< Proxy FTP Data Ack out non-secure 2/2  TCP/ACK port 20
      permit tcp/ack >1023 =20   non-secure local outbound l=n f=y
  >i< Proxy FTP Data out non-secure 2/2      TCP port 1023+
      permit tcp     >1023 >1023 non-secure local outbound l=n f=y
  >o< Proxy FTP Data Ack in non-secure 2/2   TCP/ACK port 1023+
      permit tcp/ack >1023 >1023 non-secure local inbound  l=n f=y

Gopher proxy out 2/2           Permit gopher from firewall to non-secure network
  >i< Proxy gopher out 2/2                   TCP out port 70 non-secure
      permit tcp     >1023 =70   non-secure local outbound l=n f=y
  >o< Proxy gopher Ack out 2/2               TCP/ACK in port 70 non-secure
```

```
      permit tcp/ack =70   >1023 non-secure local inbound  l=n f=y

HTTP deny non-secure            Deny HTTP to non-secure interfaces
  >i< HTTP deny                                deny All in port 8080 non-secure
      deny   all     any 0 =8080 non-secure both  inbound  l=y f=h

HTTP direct out                 Permit HTTP from secure network directly to non-secure network
  >i< HTTP 1/2                                 TCP in port 80 secure
      permit tcp     >1023 =80   secure     route inbound  l=n f=y
  >i< HTTP 2/2                                 TCP out port 80 non-secure
      permit tcp     >1023 =80   non-secure route outbound l=n f=y
  >o< HTTP Ack 1/2                             TCP/ACK in port 80 non-secure
      permit tcp/ack =80   >1023 non-secure route inbound  l=n f=y
  >o< HTTP Ack 2/2                             TCP/ACK out port 80 secure
      permit tcp/ack =80   >1023 secure     route outbound l=n f=y

HTTP proxy out 1/2              Permit HTTP (port 8080) from secure network to the firewall
  >i< HTTP in 8080                             TCP in port 8080 secure
      permit tcp     >1023 =8080 secure     local inbound  l=n f=y
  >o< HTTP response                            TCP/ACK out port 8080 secure
      permit tcp/ack =8080 >1023 secure     local outbound l=n f=y

HTTP proxy out 2/2              Permit HTTP from firewall to non-secure network
  >i< Proxy HTTP out 2/2                       TCP out port 80 non-secure
      permit tcp     >1023 =80   non-secure local outbound l=n f=y
  >o< Proxy HTTP Ack out 2/2                   TCP/ACK in port 80 non-secure
      permit tcp/ack =80   >1023 non-secure local inbound  l=n f=y

HTTP proxy out secure (2/2)     Permit HTTP from firewall to secure network
  >o< Proxy HTTP out secure (2/2)              TCP out port 80 secure
      permit tcp     >1023 =80   secure     local outbound l=n f=y
  >i< Proxy HTTP ACK out secure (2/2)          TCP/ACK in port 80 secure
      permit tcp/ack =80   >1023 secure     local inbound  l=n f=y

HTTPS direct out                Permit HTTPS (SSL) from secure network to non-secure network
  >i< HTTPS 1/2                                TCP in port 443 secure
      permit tcp     >1023 =443  secure     route inbound  l=n f=y
  >i< HTTPS 2/2                                TCP out port 443 non-secure
      permit tcp     >1023 =443  non-secure route outbound l=n f=y
  >o< HTTPS Ack 1/2                            TCP/ACK in port 443 non-secure
      permit tcp/ack =443  >1023 non-secure route inbound  l=n f=y
  >o< HTTPS Ack 2/2                            TCP/ACK out port 443 secure
      permit tcp/ack =443  >1023 secure     route outbound l=n f=y

HTTPS proxy out 2/2             Permit HTTPS (SSL tunnel) from firewall to non-secure network
  >i< Proxy HTTPS out 2/2                      TCP out port 443 non-secure
      permit tcp     >1023 =443  non-secure local outbound l=n f=y
  >o< Proxy HTTPS Ack out 2/2                  TCP/ACK in port 443 non-secure
      permit tcp/ack =443  >1023 non-secure local inbound  l=n f=y

IDENTD                          Permit user identification with Socks protocols
  >i< IDENTD                                   TCP out port 113 secure
      permit tcp     >1023 =113  secure     local outbound l=n f=y
  >o< IDENTD Ack                               TCP/ACK in on port 113 secure
      permit tcp/ack =113  >1023 secure     local inbound  l=n f=y

Mail locally in and out         Mail locally in and out
  >i< Mail                                     TCP port 25
      permit tcp     >1023 =25   both       local both     l=n f=y
  >o< Mail Ack                                 TCP/ACK port 25
      permit tcp/ack =25   >1023 both       local both     l=n f=y
  >o< Mail                                     TCP port 25
      permit tcp     >1023 =25   both       local both     l=n f=y
  >i< Mail Ack                                 TCP/ACK port 25
      permit tcp/ack =25   >1023 both       local both     l=n f=y

Mail                            Permit Mail traffic through firewall
  >i< Mail                                     TCP port 25
      permit tcp     >1023 =25   both       local both     l=n f=y
  >o< Mail Ack                                 TCP/ACK port 25
      permit tcp/ack =25   >1023 both       local both     l=n f=y

MF Config from EFM on non-secure Permit config from EFM on non-secure
  >i< EFM config in non-secure                 TCP in port 1024 non-secure
      permit tcp     >1025 =1024 non-secure local inbound  l=n f=y
```

```
  >o< EFM config ACK out non-secure           TCP/ACK out port 1024 non-secure
      permit tcp/ack =1024 >1025 non-secure local outbound l=n f=y

Permit ALL with Logging          Permit all (for debugging purposes only)
  >i< All - permit                             permit All
      permit all     any 0 any 0 both       both  both     l=y f=y
  >o< All - permit                             permit All
      permit all     any 0 any 0 both       both  both     l=y f=y

Permit FTP outbound from firewall to secure Permit FTP outbound from firewall to secure
  >o< FTP Client 2/2                           TCP/ACK in >1023 from 21
      permit tcp/ack =21   >1023 secure     local inbound  l=n f=y
  >o< FTP Client Data 1/2                      TCP in port 20 secure
      permit tcp     =20   >1023 secure     local inbound  l=n f=y
  >i< FTP Client Data 2/2                      TCP/ACK out port >1023 to 20 secure
      permit tcp/ack >1023 =20   secure     local outbound l=n f=y
  >o< FTP Client 1/2                           TCP in port >1023 from 21 secure
      permit tcp     =21   >1023 secure     local inbound  l=n f=y
  >i< FTP Client 3/2                           TCP from >1023 to 21 secure out
      permit all     >1023 =21   secure     local outbound l=n f=y

Ping                             Permit Ping outbound secure network to anywhere
  >i< Ping                                     ICMP port 8
      permit icmp    =8    =0    both       both  both     l=n f=y
  >o< Ping Response                            ICMP port 0
      permit icmp    =0    =0    both       both  both     l=n f=y

Real Audio proxy  (1/2)          Real Audio from secure to fw
  >i< Real Audio proxy (1/2) in secure         TCP in port 1090 secure
      permit tcp     >1023 =1090 secure     local inbound  l=y f=y
  >o< Real Audio proxy  ACK (1/2) secure       TCP/ACK out port 1090 secure
      permit tcp/ack =1090 >1023 secure     local outbound l=y f=y

Real Audio proxy (2/2)           Real Audio from firewall to world
  >i< Real Audio proxy (2/2)                   TCP out port 7070 non-secure
      permit tcp     >1023 =7070 non-secure local outbound l=y f=y
  >o< Real Audio proxy ACK (2/2)               TCP/ACK in port 7070 non-secure
      permit tcp/ack =7070 >1023 non-secure local inbound  l=y f=y

Real audio route 7070 log on     Real Audio route 7070
  >i< Real Audio in secure 7070 log on         TCP in port 7070 secure
      permit tcp     >1023 =7070 secure     route inbound  l=y f=y
  >i< Real Audio out 7070 non-secure log on    TCP out port 7070 non-secure
      permit tcp     >1023 =7070 non-secure route outbound l=y f=y
  >o< Real Audio ACK  7070 in non-secure log on TCP/ACK in port 7070 non-secure
      permit tcp/ack =7070 >1023 non-secure route inbound  l=y f=y
  >o< Real Audio ACK 7070 out secure log on    TCP/ACK out port 7070 secure
      permit tcp/ack =7070 >1023 secure     route outbound l=y f=y

Real Audio route 80              Real Audio route 80
  >i< Real Audio 80 non-secure                 TCP out port 80 non-secure
      permit tcp     >1023 =80   non-secure route outbound l=n f=y
  >i< Real Audio 80 secure                     TCP in port 80 secure
      permit tcp     >1023 =80   secure     route inbound  l=y f=y
  >o< Real Audio ACK 80 non-secure             TCP/ACK in port 80 non-secure
      permit tcp/ack =80   >1023 non-secure route inbound  l=y f=y
  >o< Real Audio ACK 80 secure                 TCP/ACK out port 80 secure
      permit tcp/ack =80   >1023 secure     route outbound l=n f=y

Real Audio udp >1023             Real Audio udp >1023
  >i< Real audio udp >1023                     Real Audio udp >1023
      permit udp     =80   >1023 both       route both     l=y f=y
  >o< Real audio udp >1023                     Real Audio udp >1023
      permit udp     =80   >1023 both       route both     l=y f=y

RealAudio                        Permit RealAudio connection from secure network to non-secure network
  >i< RealAudio in secure                      TCP in port 7070 secure
      permit tcp     >1023 =7070 secure     route inbound  l=n f=y
  >i< RealAudio out non-secure                 TCP out port 7070 non-secure
      permit tcp     >1023 =7070 non-secure route outbound l=n f=y
  >o< RealAudio Ack in non-secure              TCP/ACK in port 7070 non-secure
      permit tcp/ack =7070 >1023 non-secure route inbound  l=n f=y
  >o< RealAudio out secure                     TCP/ACK out port 7070 secure
      permit tcp/ack =7070 >1023 secure     route outbound l=n f=y
```

```
Remote Client - AIX             Permit Encrypted data flow between Firewall and client
  >i< VPN Authentication                        AH any port non-secure
      permit ah      any 0 any 0 non-secure local both     l=n f=y
  >o< VPN Authentication                        AH any port non-secure
      permit ah      any 0 any 0 non-secure local both     l=n f=y
  >i< VPN Encryption                            ESP any port non-secure
      permit esp     any 0 any 0 non-secure local both     l=n f=y
  >o< VPN Encryption                            ESP any port non-secure
      permit esp     any 0 any 0 non-secure local both     l=n f=y

Remote Logging                  Permit redirect of firewall logs to remote host
  >i< Log - Remote                              UDP out port 514 secure
      permit udp     >1023 =514  secure     local outbound l=n f=y

SDI authentication              Permit connection to SecurID ACE server in the secure network
  >i< SDI Queries                               UDP out port 5500 secure
      permit udp     >1023 =5500 secure     local outbound l=n f=y
  >o< SDI Replies                               UDP in port 5500 secure
      permit udp     =5500 >1023 secure     local inbound  l=n f=y

SNMP query deny                 Deny SNMP query from SNMP manager
  >i< SNMP query deny                           All in port 161
      deny   all     >1023 =161  both       local both     l=y f=h
  >o< SNMP reply deny                           All out port 161
      deny   all     =161  >1023 both       local both     l=y f=h

SNMP query                      Permit SNMP query from SNMP manager
  >i< SNMP query                                All in port 161
      permit all     >1023 =161  both       local both     l=n f=y
  >o< SNMP reply                                All out port 161
      permit all     =161  >1023 both       local both     l=n f=y

SNMP traps                      Permit SNMP Trap service
  >o< SNMPTRAP                                  Permit SNMPTRAP from Firewall 161 to port 162
      permit all     =161  =162  both       local outbound l=n f=y

Socks 1/2                       Permit use of Socks from secure network to the firewall
  >i< Socks TCP                                 TCP in port 1080 secure
      permit tcp     >1023 =1080 secure     local inbound  l=n f=y
  >o< Socks TCP Ack                             TCP/ACK out port 1080 secure
      permit tcp/ack =1080 >1023 secure     local outbound l=n f=y

Socks deny non-secure           Deny Socks from non-secure Adapters
  >i< Socks deny                                port 1080 non-secure
      deny   tcp     any 0 =1080 non-secure both  both     l=y f=h

SSL Server Secure               Permit SSL server traffic to remote SSL agents
  >o< SSL Server in Secure                      TCP in port 4005
      permit tcp     any 0 =4005 secure     local inbound  l=n f=y
  >i< SSL Server out secure                     TCP/ACK out port 4005
      permit tcp/ack =4005 any 0 secure     local outbound l=n f=y

SSL Server                      Permit SSL server traffic to remote SSL agents
  >i< SSL Server in non-secure                  TCP in port 4005
      permit tcp     any 0 =4005 non-secure local inbound  l=n f=y
  >o< SSL Server out non-secure                 TCP/ACK out port 4005
      permit tcp/ack =4005 any 0 non-secure local outbound l=n f=y

Telnet direct out               Permit Telnet outbound from secure network to non-secure network
  >i< Telnet in secure                          TCP port 23 permit
      permit tcp     >1023 =23   secure     route inbound  l=n f=y
  >i< Telnet out non-secure                     TCP port 23
      permit tcp     >1023 =23   non-secure route outbound l=n f=y
  >o< Telnet Ack in non-secure                  TCP/ACK port 23
      permit tcp/ack =23   >1023 non-secure route inbound  l=n f=y
  >o< Telnet Ack out secure                     TCP/ACK port 23
      permit tcp/ack =23   >1023 secure     route outbound l=n f=y

Telnet Internet to FW           Telnet Internet to FW
  >i< >1023 to 23 inbound Non-Secure            Telnet to FW from Internet 1/2
      permit all     >1023 =23   non-secure local inbound  l=n f=y
  >o< 23 to >1023 outbound tcp/ack Non-Secure  Telnet to FW from Internet 2/2
      permit all     =23   >1023 non-secure local outbound l=n f=y

Telnet proxy in 1/2             Permit Telnet inbound from non-secure network to firewall
```

```
  >i< Proxy Telnet in non-secure 1/2          TCP port 23
      permit tcp     >1023 =23   non-secure local inbound  l=n f=y
  >o< Proxy Telnet Ack out non-secure 1/2     TCP/ACK port 23
      permit tcp/ack =23   >1023 non-secure local outbound l=n f=y

Telnet proxy in 2/2             Permit telnet in from firewall to secure network
  >i< Proxy Telnet out secure 2/2             TCP port 23
      permit tcp     >1023 =23   secure     local outbound l=n f=y
  >o< Proxy Telnet Ack in secure 2/2          TCP/ACK port 23
      permit tcp/ack =23   >1023 secure     local inbound  l=n f=y

Telnet proxy out 1/2            Permit Telnet out from secure network to firewall
  >i< Proxy Telnet in secure 1/2              TCP port 23
      permit tcp     >1023 =23   secure     local inbound  l=n f=y
  >o< Proxy Telnet Ack out secure 1/2         TCP/ACK port 23
      permit tcp/ack =23   >1023 secure     local outbound l=n f=y

Telnet proxy out 2/2            Permit telnet out from firewall to non-secure network
  >i< Proxy Telnet out non-secure 2/2         TCP port 23
      permit tcp     >1023 =23   non-secure local outbound l=n f=y
  >o< Proxy Telnet Ack in non-secure 2/2      TCP/ACK port 23
      permit tcp/ack =23   >1023 non-secure local inbound  l=n f=y

Telnet routed from Internet     Telnet routed from Internet
  >i< Telnet route in non-secure              telnet from Internet
      permit tcp     =23   =23   non-secure route inbound  l=n f=y
  >i< Telnet route out secure                 telnet from Internet
      permit tcp     =23   =23   secure     route outbound l=n f=y
  >o< Telnet route ACK in secure              Telnet from Internet
      permit tcp/ack =23   =23   secure     route inbound  l=n f=y
  >o< Telnet route ACK out non-secure         Telnet from Internet
      permit tcp/ack =23   =23   non-secure route outbound l=n f=y

Tunnel (307) / inbound / telnet  0
  >o< Proxy Telnet Ack out secure 1/2         TCP/ACK port 23
      permit tcp/ack =23   >1023 secure     local outbound l=n f=y
  >i< Proxy Telnet in secure 1/2              TCP port 23
      permit tcp     >1023 =23   secure     local inbound  l=n f=y

Tunnel 309 all
  >i< permit icmp all
      permit icmp    any 0 any 0 secure     local both     l=y f=y

tunnel 309 telnet
  >i< Proxy Telnet Ack in secure 2/2          TCP/ACK port 23
      permit tcp/ack =23   >1023 secure     local inbound  l=n f=y
  >o< Proxy Telnet out secure 2/2             TCP port 23
      permit tcp     >1023 =23   secure     local outbound l=n f=y

VDOLIVE Direct In               Permit non-secure client to secure server
  >i< VDOLIVE TCP in non-secure               VDOLIVE TCP Session from any port ( > 1023) to 7000
      permit tcp     >1023 =7000 non-secure route inbound  l=n f=y
  >i< VDOLIVE TCP out secure                  VDOLIVE TCP Session from any port ( > 1023) to 7000
      permit tcp     >1023 =7000 secure     route outbound l=n f=y
  >o< VDOLIVE TCP/ACK in secure               VDOLIVE TCP/ACK Session from 7000 to any port (>1023)
      permit tcp/ack =7000 >1023 secure     route inbound  l=n f=y
  >o< VDOLIVE TCP/ACK out non-secure          VDOLIVE TCP/ACK Session from 7000 to any port (>1023)
      permit tcp/ack =7000 >1023 non-secure route outbound l=n f=y
  >i< VDOLIVE UDP Forward in non-secure       VDOLIVE UDP Session from any port (>1023) to 7001
      permit udp     >1023 =7001 non-secure route inbound  l=n f=y
  >i< VDOLIVE UDP Forward out secure          VDOLIVE UDP Session from any port (>1023) to 7001
      permit udp     >1023 =7001 secure     route outbound l=n f=y
  >o< VDOLIVE UDP Backward in secure          VDOLIVE UDP Session from 7001 to any port (>1023)
      permit udp     =7001 >1023 secure     route inbound  l=n f=y
  >o< VDOLIVE UDP Backward out non-secure     VDOLIVE UDP Session from 7001 to any port (>1023)
      permit udp     =7001 >1023 non-secure route outbound l=n f=y

VDOLIVE Direct Out              Permit secure client to non-secure server
  >i< VDOLIVE TCP in secure                   VDOLIVE TCP Session from any port ( > 1023) to 7000
      permit tcp     >1023 =7000 secure     route inbound  l=n f=y
  >i< VDOLIVE TCP out non-secure              VDOLIVE TCP Session from any port ( > 1023) to 7000
      permit tcp     >1023 =7000 non-secure route outbound l=n f=y
  >o< VDOLIVE TCP/ACK in non-secure           VDOLIVE TCP/ACK Session from 7000 to any port (>1023)
      permit tcp/ack =7000 >1023 non-secure route inbound  l=n f=y
  >o< VDOLIVE TCP/ACK out secure              VDOLIVE TCP/ACK Session from 7000 to any port (>1023)
```

```
      permit tcp/ack =7000 >1023 secure     route outbound l=n f=y
  >i< VDOLIVE UDP Forward in secure           VDOLIVE UDP Session from any port (>1023) to 7001
      permit udp     >1023 =7001 secure     route inbound  l=n f=y
  >i< VDOLIVE UDP Forward out non-secure      VDOLIVE UDP Session from any port (>1023) to 7001
      permit udp     >1023 =7001 non-secure route outbound l=n f=y
  >o< VDOLIVE UDP Backward in non-secure      VDOLIVE UDP Session from 7001 to any port (>1023)
      permit udp     =7001 >1023 non-secure route inbound  l=n f=y
  >o< VDOLIVE UDP Backward out secure         VDOLIVE UDP Session from 7001 to any port (>1023)
      permit udp     =7001 >1023 secure     route outbound l=n f=y

VPN encapsulation               Permit encrypted data between firewalls
  >i< VPN Authentication                      AH any port non-secure
      permit ah      any 0 any 0 non-secure local both     l=n f=y
  >o< VPN Authentication                      AH any port non-secure
      permit ah      any 0 any 0 non-secure local both     l=n f=y
  >i< VPN Encryption                          ESP any port non-secure
      permit esp     any 0 any 0 non-secure local both     l=n f=y
  >o< VPN Encryption                          ESP any port non-secure
      permit esp     any 0 any 0 non-secure local both     l=n f=y

VPN key exchange                Permit session key exchanges for IBM tunnels
  >i< VPN Key Xchg                            UDP port 4001 non-secure
      permit udp     =4001 =4001 non-secure local both     l=n f=y
  >o< VPN Key Xchg                            UDP port 4001 non-secure
      permit udp     =4001 =4001 non-secure local both     l=n f=y

VPN traffic 1/2                 Permit routed traffic on secure interface (non-encrypted)
  >i< VPNs in secure                          All protocols
      permit all     any 0 any 0 secure     route inbound  l=n f=y
  >o< VPNs out secure                         All protocols
      permit all     any 0 any 0 secure     route outbound l=n f=y

VPN traffic 2/2                 Permit routed traffic on non-secure interface (encrypted)
  >i< VPNs out non-secure                     All protocols
      permit all     any 0 any 0 non-secure route outbound l=n f=y
  >o< VPNs in non-secure                      All protocols
      permit all     any 0 any 0 non-secure route inbound  l=n f=y

WAIS proxy out 2/2              Permit WAIS (z39.50) from firewall to non-secure network
  >i< Proxy WAIS out 2/2                      TCP out port 210 non-secure
      permit tcp     >1023 =210  non-secure local outbound l=n f=y
  >o< Proxy WAIS Ack out 2/2                  TCP/ACK in port 210 non-secure
      permit tcp/ack =210  >1023 non-secure local inbound  l=n f=y
```

Appendix G. Special Notices

This document was written for planners and implementers of firewalls, to help in designing and configuring solutions using the IBM Firewall for AIX. Some knowledge of TCP/IP protocols is assumed. The information in this publication is not intended as the specification of any programming interfaces that are provided by the IBM Firewall for AIX. See the PUBLICATIONS section of the IBM Programming Announcement for IBM Firewall for AIX for more information about what publications are considered to be product documentation.

References in this publication to IBM products, programs or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program, or service is not intended to state or imply that only IBM's product, program, or service may be used. Any functionally equivalent program that does not infringe any of IBM's intellectual property rights may be used instead of the IBM product, program or service.

Information in this book was developed in conjunction with use of the equipment specified, and is limited in application to those specific hardware and software products and levels.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to the IBM Director of Licensing, IBM Corporation, 500 Columbus Avenue, Thornwood, NY 10594 USA.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact IBM Corporation, Dept. 600A, Mail Drop 1329, Somers, NY 10589 USA.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The information contained in this document has not been submitted to any formal IBM test and is distributed AS IS. The information about non-IBM ("vendor") products in this manual has been supplied by the vendor and IBM assumes no responsibility for its accuracy or completeness. The use of this information or the implementation of any of these techniques is a customer responsibility and depends on the customer's ability to evaluate and integrate them into the customer's operational environment. While each item may have been reviewed by IBM for accuracy in a specific situation, there is no guarantee that the same or similar results will be obtained elsewhere. Customers attempting to adapt these techniques to their own environments do so at their own risk.

Any performance data contained in this document was determined in a controlled environment, and therefore, the results that may be obtained in other operating environments may vary significantly. Users of this document should verify the applicable data for their specific environment.

Reference to PTF numbers that have not been released through the normal distribution process does not imply general availability. The purpose of

including these reference numbers is to alert IBM customers to specific information relative to the implementation of the PTF when it becomes available to each customer according to the normal IBM PTF distribution process.

The following terms are trademarks of the International Business Machines Corporation in the United States and/or other countries:

AIX	Current
DB2	IBM
NetView	OS/2
RISC System/6000	RS/6000
SP	SystemView

The following terms are trademarks of other companies:

C-bus is a trademark of Corollary, Inc.

Java and HotJava are trademarks of Sun Microsystems, Incorporated.

Microsoft, Windows, Windows NT, and the Windows 95 logo are trademarks or registered trademarks of Microsoft Corporation.

PC Direct is a trademark of Ziff Communications Company and is used by IBM Corporation under license.

Pentium, MMX, ProShare, LANDesk, and ActionMedia are trademarks or registered trademarks of Intel Corporation in the U.S. and other countries.

UNIX is a registered trademark in the United States and other countries licensed exclusively through X/Open Company Limited.

Other company, product, and service names may be trademarks or service marks of others.

Appendix H. Related Publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this redbook.

H.1 International Technical Support Organization Publications

For information on ordering these ITSO publications see “How to Get ITSO Redbooks” on page 305.

- *Safe Surfing: How to Build a Secure World Wide Web Connection*, SG24-4564
- *Managing One or More AIX Systems - Overview*, GG24-4160
- *IBM Systems Monitor, Anatomy of a Smart Agent*, SG24-4398

H.2 Redbooks on CD-ROMs

Redbooks are also available on CD-ROMs. **Order a subscription** and receive updates 2-4 times a year at significant savings.

CD-ROM Title	Subscription Number	Collection Kit Number
System/390 Redbooks Collection	SBOF-7201	SK2T-2177
Networking and Systems Management Redbooks Collection	SBOF-7370	SK2T-6022
Transaction Processing and Data Management Redbook	SBOF-7240	SK2T-8038
AS/400 Redbooks Collection	SBOF-7270	SK2T-2849
RS/6000 Redbooks Collection (HTML, BkMgr)	SBOF-7230	SK2T-8040
RS/6000 Redbooks Collection (PostScript)	SBOF-7205	SK2T-8041
Application Development Redbooks Collection	SBOF-7290	SK2T-8037
Personal Systems Redbooks Collection	SBOF-7250	SK2T-8042

H.3 Other Publications

These publications are also relevant as further information sources:

- *IBM Firewall 3.1 Version 2.1 Installation, Configuration, and Administration Guide*, SC31-8113-01
- *Building Internet Firewalls,* D. Brent Chapman and Elizabeth D. Zwicky, Published by O'Reilly & Associates Inc, ISBN 1-56592-124-0
- *Firewalls and Internet Security, Repelling the Wily Hacker*, William R. Cheswick and Steven M. Bellovin. Published by Addison-Wesley 1994, ISBN 0-201-63357-4
- *Actually Useful Internet Security Techniques*, Larry J. Hughes Jr., Published by New Riders, 1995, ISBN 1-56205-508-9

How to Get ITSO Redbooks

This section explains how both customers and IBM employees can find out about ITSO redbooks, CD-ROMs, workshops, and residencies. A form for ordering books and CD-ROMs is also provided.

This information was current at the time of publication, but is continually subject to change. The latest information may be found at http://www.redbooks.ibm.com.

How IBM Employees Can Get ITSO Redbooks

Employees may request ITSO deliverables (redbooks, BookManager BOOKs, and CD-ROMs) and information about redbooks, workshops, and residencies in the following ways:

- **PUBORDER** —to order hardcopies in United States
- **GOPHER link to the Internet** - type GOPHER.WTSCPOK.ITSO.IBM.COM
- **Tools disks**

  To get LIST3820s of redbooks, type one of the following commands:

  ```
  TOOLS SENDTO EHONE4 TOOLS2 REDPRINT GET SG24xxxx PACKAGE
  TOOLS SENDTO CANVM2 TOOLS REDPRINT GET SG24xxxx PACKAGE (Canadian users only)
  ```

  To get BookManager BOOKs of redbooks, type the following command:

  ```
  TOOLCAT REDBOOKS
  ```

  To get lists of redbooks, type one of the following commands:

  ```
  TOOLS SENDTO USDIST MKTTOOLS MKTTOOLS GET ITSOCAT TXT
  TOOLS SENDTO USDIST MKTTOOLS MKTTOOLS GET LISTSERV PACKAGE
  ```

  To register for information on workshops, residencies, and redbooks, type the following command:

  ```
  TOOLS SENDTO WTSCPOK TOOLS ZDISK GET ITSOREGI 1998
  ```

  For a list of product area specialists in the ITSO: type the following command:

  ```
  TOOLS SENDTO WTSCPOK TOOLS ZDISK GET ORGCARD PACKAGE
  ```

- **Redbooks Web Site on the World Wide Web**

  http://w3.itso.ibm.com/redbooks

- **IBM Direct Publications Catalog on the World Wide Web**

  http://www.elink.ibmlink.ibm.com/pbl/pbl

  IBM employees may obtain LIST3820s of redbooks from this page.

- **REDBOOKS category on INEWS**
- **Online** —send orders to: USIB6FPL at IBMMAIL or DKIBMBSH at IBMMAIL
- **Internet Listserver**

  With an Internet e-mail address, anyone can subscribe to an IBM Announcement Listserver. To initiate the service, send an e-mail note to announce@webster.ibmlink.ibm.com with the keyword subscribe in the body of the note (leave the subject line blank). A category form and detailed instructions will be sent to you.

Redpieces

For information so current it is still in the process of being written, look at "Redpieces" on the Redbooks Web Site (http://www.redbooks.ibm.com/redpieces.htm). Redpieces are redbooks in progress; not all redbooks become redpieces, and sometimes just a few chapters will be published this way. The intent is to get the information out much quicker than the formal publishing process allows.

How Customers Can Get ITSO Redbooks

Customers may request ITSO deliverables (redbooks, BookManager BOOKs, and CD-ROMs) and information about redbooks, workshops, and residencies in the following ways:

- **Online Orders** —send orders to:

	IBMMAIL	**Internet**
In United States:	usib6fpl at ibmmail	usib6fpl@ibmmail.com
In Canada:	caibmbkz at ibmmail	lmannix@vnet.ibm.com
Outside North America:	dkibmbsh at ibmmail	bookshop@dk.ibm.com

- **Telephone orders**

United States (toll free)	1-800-879-2755
Canada (toll free)	1-800-IBM-4YOU
Outside North America	(long distance charges apply)
(+45) 4810-1320 - Danish	(+45) 4810-1020 - German
(+45) 4810-1420 - Dutch	(+45) 4810-1620 - Italian
(+45) 4810-1540 - English	(+45) 4810-1270 - Norwegian
(+45) 4810-1670 - Finnish	(+45) 4810-1120 - Spanish
(+45) 4810-1220 - French	(+45) 4810-1170 - Swedish

- **Mail Orders** —send orders to:

IBM Publications Publications Customer Support P.O. Box 29570 Raleigh, NC 27626-0570 USA	IBM Publications 144-4th Avenue, S.W. Calgary, Alberta T2P 3N5 Canada	IBM Direct Services Sortemosevej 21 DK-3450 Allerød Denmark

- **Fax** —send orders to:

United States (toll free)	1-800-445-9269
Canada	1-403-267-4455
Outside North America	(+45) 48 14 2207 (long distance charge)

- **1-800-IBM-4FAX (United States)** or **(+1)001-408-256-5422 (Outside USA)** —ask for:

  Index # 4421 Abstracts of new redbooks
  Index # 4422 IBM redbooks
  Index # 4420 Redbooks for last six months

- **Direct Services** - send note to `softwareshop@vnet.ibm.com`
- **On the World Wide Web**

Redbooks Web Site	http://www.redbooks.ibm.com
IBM Direct Publications Catalog	http://www.elink.ibmlink.ibm.com/pbl/pbl

- **Internet Listserver**

  With an Internet e-mail address, anyone can subscribe to an IBM Announcement Listserver. To initiate the service, send an e-mail note to `announce@webster.ibmlink.ibm.com` with the keyword `subscribe` in the body of the note (leave the subject line blank).

Redpieces

For information so current it is still in the process of being written, look at "Redpieces" on the Redbooks Web Site (`http://www.redbooks.ibm.com/redpieces.htm`). Redpieces are redbooks in progress; not all redbooks become redpieces, and sometimes just a few chapters will be published this way. The intent is to get the information out much quicker than the formal publishing process allows.

IBM Redbook Order Form

Please send me the following:

Title	Order Number	Quantity

First name Last name

Company

Address

City Postal code Country

Telephone number Telefax number VAT number

□ Invoice to customer number

□ Credit card number

Credit card expiration date Card issued to Signature

We accept American Express, Diners, Eurocard, Master Card, and Visa. Payment by credit card not available in all countries. Signature mandatory for credit card payment.

Index

L

M

N

O

P

R

S

T

U

V

W

X

ITSO Redbook Evaluation

Protect and Survive Using IBM Firewall 3.1 for AIX
SG24-2577-02

Your feedback is very important to help us maintain the quality of ITSO redbooks. **Please complete this questionnaire and return it using one of the following methods:**

- Use the online evaluation form found at http://www.redbooks.ibm.com
- Fax this form to: USA International Access Code + 1 914 432 8264
- Send your comments in an Internet note to redbook@vnet.ibm.com

Please rate your overall satisfaction with this book using the scale:
(1 = very good, 2 = good, 3 = average, 4 = poor, 5 = very poor)

Overall Satisfaction _____

Please answer the following questions:

Was this redbook published in time for your needs? Yes____ No____

If no, please explain:

_____

_____

_____

_____

What other redbooks would you like to see published?

_____

_____

_____

Comments/Suggestions: **(THANK YOU FOR YOUR FEEDBACK!)**

_____

_____

_____

_____

_____

SG24-2577-02
Printed in the U.S.A.