Administrative Domain Security Gateway for File Transfer

by

David Sze Yuen Maw

S.B. Computer Science and Engineering,
Massachusetts Institute of Technology
(1993)

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Science in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 1994

Author ....
Department of Electrical Engineering and Computer Science
July 8, 1994

Certified by ....
John T. Wroclawski
Research Scientist
Thesis Supervisor

Accepted by ....
Professor Frederic R. Morgenthaler
Chairman, Departmental Committee on Graduate Students

Administrative Domain Security Gateway for File Transfer

by

David Sze Yuen Maw

Submitted to the Department of Electrical Engineering and Computer Science
on July 8, 1994, in partial fulfillment of the
requirements for the degree of
Master of Science in Electrical Engineering and Computer Science

Abstract

As the Internet becomes larger, domain administrators want to protect their hosts without sacrificing the convenience the Internet provides. This research uses an application gateway for securing file transfer between hosts of an administrative domain and the outside networks. The thesis introduces mechanisms for setting administrative domain security rules, configuring finer-granularity access control, and presenting the hosts in a domain as a single virtual file system to outside clients. Comparisons between existing file transfer systems and the new file transfer system show that the new model not only strengthens the security of the file transfer services, but also adds convenience in accessing files at different host machines in an administrative domain.

Thesis Supervisor: John T. Wroclawski
Title: Research Scientist

Acknowledgments

I would like to thank the following people in helping me throughout the final weeks of finishing this thesis in different aspects: John Wroclawski, Kelli Foster, Jimmy Lai, King Yu, Irene Fung, Patrick Chan, Anita Chan, Sin Yan Law, Brian So, Ann Chen and Paul Ngan.

Thanks for the prayer support from the HKSBS fellows. Afterall, it must have been His providence and grace.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In the early days of computer internetworking, convenience and efficiency dominated efforts to improve computer networks. Certainly, a number of computer developers raised concerns about the problems of computer network security,[1] but seldom found security to be a big issue when the whole community was pushing for a faster and more versatile computer networking environment. Some computer administrators tried their best in configuring the security options offered by each of their host machines, and the users trusted and relied on the security functions provided by their hosts. Then, in the late 1980s, a few Internet security problems and break-ins, such as the Wiley Hacker and the Internet Worm, were well-publicized [1][2][3]. Many people, even the computer administrators, started to realize that the networking security functions their host machines could offer were limited. Some completely disconnected their machines from the network, and many withdrew direct connections of their machines to the network while keeping only e-mail exchanges through a network gateway. Since then, more computer developers are aware of security concerns regarding computer networking, and some have started to put their efforts into developing network security gateways.

Despite the limited security of the Internet, many new organizations are connecting to the Internet each month for the convenience it provides; consequently,

[1] For example, the National Security Agency (NSA) and the Department of Defense (DOD) of the United States have always been concerned about computer network security.

the environment are becoming more dispersed and diverse. No one can or would like to effectively police such a fast-growing and widely-distributed environment as a whole; therefore, unrestricted connectivity of a host machine to the network is not recommended and is done only at the risk of exposing the host machine—and other connected hosts within the organization—to outside tampering. Instead, computer administrators of an organizational domain should secure their hosts and set policy boundaries for limiting connectivity between their hosts and the outside network.

This thesis focuses on the usage of gateways to provide security measures for limiting network connectivity by filtering unwanted services. There are other fundamental technical issues in computer network security, ranging from the application of cryptography on communication channels to the usage of a reliable host operating system for providing authorized access on the different files. More detailed discussions on these other technical issues can be found in the book, "Computers at Risk" [4].

1.1 Security Gateway

The concept and usage of gateways have existed in the internetworking environment from its beginning. There are two types of network gateways commonly in place: routers and application gateways. Application gateways refer to devices that convert between different protocol suites; routers refer to devices that relay and route packets between networks without converting between protocols at any layer higher than the network layer.[2]

Over the past five years, there have been a considerable number of projects in building gateways for network security. Most of them use routers to filter packets. There are now well-known existing commercial routers capable of filtering packets.[3] Some projects have attempted to add security capabilities to application gateways. The advantages and limitations of the two types of security gateways are discussed.

[2] Computer networking has a standard Open System Interconnection (OSI) seven-layer model. The network layer is Layer 3 [5].

[3] For example, the Cisco Systems routers provide a packet-filtering capability.

1.1.1 Packet-Filtering Router

Different existing packet-filtering routers[4] operate in the same basic fashion. They look into the information in the packet header and apply rules from an administrator-directed rule base to determine whether to permit routing of the packet or to drop it. With the current Internet Protocol (IP) [7], packet type (TCP[5], UDP[6], etc.), source IP address, destination IP address, and destination port are available in the packet header for applying the technique of filtering packets.

A packet-filtering capability is easy to implement and apply in routers. Depending upon the flexibility and usability of the designed language for writing the security rules, a packet-filtering router can provide a simple way for administrators to apply access control policies on which protocols, which hosts and which service ports may be accessed from remote locations. As a result, unneeded and unwanted packet traffic can be reduced and network resources can be protected from blacklisted sites in the network. However, the granularity of access control achievable by filtering packets is naturally limited to only the three or four parametric fields available in the packet header. There is also some concern about the reliability of the IP source address accuracy. As discussed in the article, "Security Problems in the TCP/IP Protocol Suite" [10], the IP source addresses can be faked. Fraudulent source address of a packet can seriously undermine the practical security function of a packet-filtering router.

1.1.2 Application Gateway for Security

Contrary to the large number of existing packet-filtering routers, not many application gateways for security have been built[7], most likely because more effort must be put into building an application gateway for security than in implementing a packet-

[4] For example, the Digital Equipment Corporation's *screend* packet-filtering program, the Cisco Systems' routers, and the Telebit Corporation's NetBlazer dial-up Internet Protocol routers all operate in the same basic fashion [6].

[5] Transmission Control Protocol [8].

[6] User Datagram Protocol [9].

[7] Two projects of building an application gateway for security can be found in [11] and [12].

filtering router. An application gateway is designed to be capable of interpreting the passing data; therefore, more effort is needed to implement the data processing unit. On the other hand, with the interpreted data, an application gateway can authorize different types of services with a finer-granularity access control than a router, and serve as an authentication check point. An application gateway for security works differently from a packet-filtering router by providing proxy access to the Internet on behalf of the host machines protected by the gateway; consequently, an application gateway for security can hide the addresses of the organization hosts from the external network. The processing of data at an application gateway, however, has the drawback of hindering the usage of a truly end-to-end encryption as well as the speed of the application service.

1.2 The Approach of the Modular Application Service Substrate

Since 1991, the Advanced Network Architecture (ANA) group of the Laboratory for Computer Science (LCS) has proposed a new approach to application-level architecture, known as the Modular Application Service Substrate (MASS), which provides some insights into network security management [13]. With this architectural approach, the application-level service is no longer structured in a purely end-to-end manner as it traditionally has been; instead, the local network and the hosts cooperate to provide the application-level services. The MASS approach suggests the decomposition of an application into its set of functional and policy requirements. The security functionality of an application service can then be seen as distinct from other functionalities. The local network enforces administrative policy and other security measures through an administrative domain gateway; the other functionalities of the application service are provided by the individual host.

1.2.1 Advantages of the MASS Approach

One important advantage of this approach to securing a domain of hosts is to help relieve many distributed, individual hosts or host applications within the domain from having to provide trustworthy environments with correct implementations for security functionalities. It also has the advantage of providing a simple, uniform management of administrative domain policies for the domain community, thus achieving administratively-directed access control, rather than relying only on the user-directed access control for the administrative domain file system. Using a gateway as a concentration point for imposing security functionality on activities into and out of a domain, the MASS approach can also be used to keep a record of different activities of controlled access for a later review.

1.2.2 Using Application Gateway for the MASS Approach

This research work provides simple and flexible mechanisms for administrators to set security policies and filtering configurations for applying the MASS approach using an application gateway. A packet-filtering router is not suitable for use in the MASS approach because of its limited capability of interpreting application services in order to provide a fine-grained access control.[8] On the other hand, the existing application gateways do not provide capabilities for administrators to specify a fine-grained access control. This research explores the potential of an application gateway to provide a finer-granularity access control.

In order to test the MASS approach, an application service must be chosen for building a prototype. Many current network security gateways only allow e-mail exchange service. File transfer, another important internetworking application service, is often not allowed to pass through an administrative domain gateway because of the nature of this service to provide capabilities of accessing to the restricted data in the file system and tampering with it. File transfer service has

[8] If only a limited access control security is needed, there are a considerable number of available fast packet-filtering routers. An article in the UNIXWorld magazine [14] provides a simple to understand tutorial for employing such technology.

been chosen for this research project because much can be learned from designing an application gateway using the MASS approach to secure this versatile application service.

1.3 File Transfer

File transfer service is one of the application services which concerns the administrators regarding domain security. Organizations fear having outsiders damage their existing files or put in extra files to create security loopholes. Organizations would also like to restrict the readability of their domain file system to users from outside of the domain.

There is a standard file transfer protocol, simply known as the File Transfer Protocol (FTP), widely used on the Internet today. This FTP has evolved from its first proposal in 1971 by the Project MAC [15] to the widely-recognized, updated specifications reported in 1985 [16]. The basic FTP model is an end-to-end, client/server model. The authentication service is provided by the server host. Once authenticated, a user may access and modify the server host file system according to its underlying file system security.

However, domain administrators are often reluctant to allow their domain hosts to provide Internet services other than just sending and receiving e-mail because they do not have enough confidence in the underlying file system security of the hosts. First, the security provided by the host file system may be largely inadequate, which is a problem for most personal computers today. In order to protect the entire domain, the administrators must find a way to provide a common set of security policies in addition to the security provided by different hosts in the domain.

Second, many file systems specify protection at the level of individual files and directories with the user-directed access control approach. This time-consuming and complex task of specifying the protection of each file may cause file users to leave some of their files inadvertently unprotected; therefore, the domain administrators may wish to impose an additional set of security policies for protecting the domain file system against security holes carelessly or even intentionally left.

Third, the underlying file system security may not provide fine-granularity protection. Consequently, the security functionality of the FTP, which is totally dependent on the underlying file system security, cannot be fine grained either. The domain administrators may want to have a set of security policies which provides a finer-granularity access control to the file system than the underlying file system security can offer. For a specific example, the domain administrators may wish to allow a user from a remote site to delete any file owned by the user, but to disallow the user from inserting any new file to the local system; however, host file systems seldom provide capability to express such a fine-grained protection.

The MASS approach suggests a methodology of using a security gateway in the local network to enforce the additional set of security policies needed by the domain administrators for protecting the file system in the presence of file transfer service to the the outside network.

1.4 Design Focus

This thesis proposes a way to secure an entire administrative domain by letting administrators set security policies at a domain gateway for file transfer between any host in the local domain and the outside network. This proposed design will be implemented in software. UNIX[9] has been chosen to be the target development environment with the standard Internet FTP as the target application for improvements. A prototype security gateway for an administrative domain will be built.

There are two different aspects of evaluation for this newly-designed system: the functionality evaluation and the assurance evaluation. The former refers to what better set of functionalities the new design provides as compared to existing designs. The latter refers to how well the implementation actually performs the functions specified in the design.

[9] UNIX is a trademark of AT&T.

This thesis does not, however, intend to introduce a security gateway that meets the level of network security required by the military. First, the time and resources necessary for developing such a highly-secured gateway are beyond the scope of this research. Second, the objective of the research is to provide an easy-to-deploy and affordable security gateway for many organizations to immediately enhance the security of their connectivity to the Internet.

Chapter 2

Design

This thesis proposes a way to secure an entire administrative domain by setting policies at a domain gateway for file transfer between a host in the administrative domain and the outside network.

2.1 Essential Ideas in the Design

The new design is based on the following ideas:

- As with the MASS approach, only the security services are distilled from the host file transfer application and are provided by the domain security gateway. Through the security gateway, the domain administrators secure the entire domain according to what they determine is suitable.

- In addition to the security of each host in the domain, the security service provided at a domain application gateway is enhanced with a finer-granularity access control policy. This enhancement, which incorporates the address-filtering capability of a packet-filtering router, is designed to increase the security of file transfer between the domain hosts and the outside network.

- The mechanisms for configuring the security policies are designed to be simple and flexible. At the same time, the file transfer service to different hosts available in an administrative domain is enhanced by employing a virtual file system model.

2.2 Basic Model for the Security Rules

The security rules are designed to express control of remote users' activities on the domain hosts. The domain administrators configure the access control policies on the domain hosts by writing a set of security rules. The domain security gateway then refers to these security rules for exerting administratively-directed access control. A new language is defined for writing security rules. The grammar of this language is designed to be simple and flexible, yet descriptive enough to make fine-granularity access control possible.

The designed security rules follow the most commonly used access control model which consists of three parts: objects, subjects and access control rights.[1] An object is the resource to be protected; an access control right allows a subject to have a certain access capability on an object. In the following subsections, the three different elements of the designed security rules are discussed in detail.

2.2.1 Subject: Combination of User and Source Address

The first element of an access control security rule to be defined is its subject. In most host security models, the subject of an access control list is a single user or a group of users.[2] In a packet-filtering router, the subject of an access control list is a source host address or a source subnet address [12][17]. In a network environment, a packet-filtering router applies the access control lists to drop packets from blacklisted sites in order to protect local domain hosts, as well as to reduce network traffic caused by unwanted attempts to request services. With its ability to interpret the data in

[1] Another well-known model is the flow model, which is derived from the Department of Defense computer security policy [4].

[2] For example, the nine-bit protection of each UNIX file represents three lists of access control on the file. The subjects of the three access control lists are the file owner, a specific group of users and the universal set of any users.

greater detail, an application gateway used in this new security system design can filter requests of specific services asked by a specific user from a specific source host. This kind of filtering is desirable, because finer-granularity access control can be imposed on the user, depending upon the trustworthiness of different combinations of originating sites and users.

Trustworthiness can be of technical concern or administrative concern. For example, if a site is well-known for its weak security or its past record of being broken into, then no service requests originating from this site may be allowed. This example illustrates how the new design incorporates the functionality of filtering packet adopted from a packet-filtering router. Another example is to deny a service request of user A from an originating site S, but to allow the same service request of user B from the same site S because an administrative policy is set to limit the leakage of information from the domain file system. In this case, user B may be a manager who is granted the convenience to access the organizational data from site S, but user A is a general staff member who is not trusted to access the data from site S. This second example illustrates how the combination of user and source address as the subject of an access control list can provide finer-granularity access control.

2.2.2 Access Control Rights: A Fine-grain Set

The second element to be defined is the set of access control rights used in a security rule. This set must be comprehensive in order to describe fine-grain access control on the files and directories in the administrative domain file system. A fairly fine-grain set of access control rights is specified in the Andrew File System (AFS) developed by Carnegie Mellon University (CMU) in the mid 1980s [18]. The AFS access control model has seven kinds of access control rights: read (r), write (w), lookup (l), insert (i), delete (d), administer (a) and lock (k). Among these rights, "the lock right has turned out not to be a particularly useful right in the AFS, but continues to to be supported for historical reasons." [18] The remaining set of access control rights provides a reasonable basis for use in the new design. Since the security gateway has a fine-grain set of access control rights, domain hosts—whose operating systems do not provide such a comprehensive set of access control rights—can rely on the security gateway to provide fine-grain authorization on different services.

2.2.3 Object: Destination Hosts in Local Domain

The last element of a security rule to be defined is the range of objects for protection. In the host operating system environment, the object for protection is usually an individual file or directory;[3] conversely, in a packet-filtering router, the object of the access control list is a destination address. For a gateway protecting a local network, the primary targets for protection are the local host machines; therefore, the destination hosts must be included in the object of an access control list. On the other hand, it would be a tedious job for the domain administrators to specify the protection of each of the hundreds of individual files and directories in the domain file system, and it would require a lot of work from the gateway in checking the protection of each file in the domain. The level of access control on individual file or directory, therefore, remains user-directed in this design, and the object of an access control list used by a domain security gateway contains only the destination hosts in the local network.

However, when there are hosts in the domain that do not have their own file protection mechanisms, it may be desirable for the language to have the capability to express individual directory or file protection despite the heavy burden checking the protection of individual files would put on the gateway. A language with this additional capability is suggested for future extension, and is discussed in Chapter 5.

In addition, one specific type of directory protection is desirable to have in the gateway security rules for any type of hosts. It is the protection of the whole file system tree in the destination host by limiting a user from a remote site to access only the subtree under the user's home directory at the destination host. This protection can be included in either the object description or the access control right description of a security rule, but it seems to fall more naturally in the latter. An additional

[3] The AFS protection only applies to directories; while the original UNIX protection applies to all types of files, which include directories.

access control right, called the right to "go up" (u) is therefore included in this new design, on top of the six suggested by the AFS. Similarly, the set of access control rights is extended to include the right to mount (m), which protects the domain file system by preventing a user without this right to mount another part of the file system subtree for accessing.

2.3 Syntax Details for the Security Rules

In deciding how the three different parts of the access control model are represented in a security rule, flexibility and simplicity are the main concerns. A simple and flexible language can increase the chances for the administrators to correctly configure the security rules for the whole system. Simplicity in the security rules is achieved by providing the ability to specify groups.

2.3.1 Subjects Syntax

User Field

In a security rule, a user name is used for the user field of the subject. In this language, a user name is expressed in the common format—a string of non-white space characters.[4]

Starting a user field with "#G:", however, signifies that it is a group name instead of a single user name; the security rule parser then looks up another administrator-defined file for the listing of users in the group. In this file, a group name is followed by a colon and a list of users in the group separated by commas. For example, in the group listing file, "Mgrs:A,B,C" and "Team1:B,D,E,F" are specified, then a security rule with a subject user field of "E" would apply to user E alone, and a security rule with a subject user field of "#G:Mgrs" would apply to users A, B and C.

[4] "White space characters are blank, tab, newline, carriage return, vertical tab and formfeed." [19]

Source Address Field

The source address field of a security rule can be represented by either the conventional four-byte address format or the conventional host name format[5]. A source address expressed in the host name format is more comprehensible than a source address expressed in the four-byte address format, but the four-byte address format is more easily processed than one expressed in the host name format. For simplicity and efficiency of the design, the four-byte address format was chosen to represent the source address field of a security rule. The capability of expressing the source address in the host name format is not included in this design, but is discussed in Chapter 5 for future extension of the design.

A group of source addresses is expressed by specifying the number of network and subnet address bits along with a source address in the group [20].[6] For example, "18.26.0.36" with a specified 24 network and subnet address bits matches both "18.26.0.188" and "18.26.0.82" in the subnetwork. The specified number of network and subnet address bits must be within the range of 8 to 31, because the network address is at least 8 bits long (for a Class A network) [7], and a group with 32 bits of network and subnet address bits is strictly speaking not a group, but a single host address. For example, the pair of "18.26.0.188 32" should be written simply as a single host address "18.26.0.188".

2.3.2 Object Syntax

The destination host (object) of a security rule is expressed in its local name within the domain. Unlike the host name of a source outside of a local domain, the name of a local host can be resolved quickly; therefore, using the host name format for the local destination host is both comprehensible and efficient. Moreover, subnet grouping

[5] For example, "18.26.0.188" and "sesame.lcs.mit.edu" express the same address written in two different conventional styles.

[6] In most other implementations with an address-filtering capability, a pair of addresses are used and the second one serves as a mask. For example, the pair of "18.26.0.188" and "255.255.255.0" matches any address with a prefix of "18.26.0". In the new design, a single integer replaces the slightly confusing mask address. For example, 24 represents "255.255.255.0" and 21 represents "255.255.248.0".

representation is insufficient when domain administrators want to divide up the local hosts within a subnetwork into subgroups for applying different security policies.

The destination host name, like the user name, is expressed in a string of non-white space characters. As in the subject user field, grouping of the destination host names is manifested using a similar method. Starting a destination field with "#G:" signifies that it is a group name instead of a single host name, the security rule parser then looks up another administrator-defined file for the listing of destination hosts in the group. In this file, a group name is followed by a colon and a list of hosts in the group separated by commas. For example, in the group listing file, "mainsrvs:ha,hb" and "decws:,ha,hc,hd" are specified, and a security rule with an object of "hc" would apply to destination host *hc* alone, and a security rule with an object of "#G:mainsrvs" would apply to both hosts *ha* and *hb*.

The design also recognizes a specific destination group named *public*, which contains the names of all the domain hosts that are designated to be publicly-known outside the local network. Details on the special feature of this *public* group are discussed in sections 2.6.1 and 2.6.5.[7]

2.3.3 Resolving Rule Conflicts

All granted access control rights are listed or grouped in a single security rule for each combination of subject and object. The capability of specifying the subjects and objects in groups is very powerful and can greatly simplify the descriptive security rules and reduce the number of rules needed. The usage of groups, however, also increases the chance of having conflicting rules.

Two General Resolution Principles

One resolution principle applied in this design is that a rule with a more specifically-matched field overrides one with a less specifically-matched field. An ordering of the

[7] Moreover, if the domain administrators want to allow anonymous FTP to a designated group of local hosts, the domain administrators can specify that group of local hosts to be accessible by user *anonymous* (from certain groups of source addresses) with a specific set of access control rights, such as "lr".

Rule 1:	#G:Team2	137.1.0.0	16	#G:sunos	l
Rule 2:	#G:Team2	137.1.0.0	16	hc	lr
Rule 3:	#G:Team2	137.1.15.0	24	#G:sunos	lri
Rule 4:	#G:Team2	137.1.15.0	24	hc	lriw
Rule 5:	C	137.1.0.0	16	#G:sunos	lriwd
Rule 6:	C	137.1.0.0	16	hc	lriwda
Rule 7:	C	137.1.15.0	24	#G:sunos	lriwdau
Rule 8:	C	137.1.15.0	24	hc	lriwdaum

Figure 2-1: A Sample Set of Security Rules

importance of the fields in matching is also put in place for resolving conflicting rules: user name first, source address second, and destination host last. This order is to allow a user-oriented capability list.[8] It is therefore recommended that a security rule is structured according to this ordering.

The need for these two principles is best illustrated in examples. For example, given user group "Team2:A,B,C" and destination group "sunos:ha,hb,hc", when user *C* from "137.1.15.33" tries to access local host *hc*, all the security rules in Figure 2-1 apply, and thus create a conflict of rules.

Security Rules 5–8 override Rules 1–4, because they match more closely in the most important field of matching, user name; and security Rules 7 and 8 have preference over Rules 5 and 6, because they match more closely in the second most important field of matching, source address; and finally, Rule 8 overrides Rule 7, because it matches more closely on the destination, and the access control rights of "lriwdaum" is granted to the authenticated user *C* accessing destination host *hc*.

Two Special Types of Rule Conflicts

There are still two more types of conflicts. The first conflict may arise if one user is involved in two different user groups or if one destination host is included in two different destination host groups. For example, there are an additional user group "Team3:B,D" and an additional destination group "servers:hb,hd", and Rule 9 and

[8] Another commonly used choice is the access control list oriented for each object.

Rule 9:	#G:Team3	137.1.15.0	24	hc	lr
Rule 10:	#G:Team2	137.1.15.0	24	#G:servers	lriwdu
Rule 11:	C	137.1.15.0	24	hc	lr
Rule 12:	A	137.1.15.0	24	ha	lr
Rule 13:	#G:Team2	137.1.0.0	16	hb	-

Figure 2-2: An Additional Set of Security Rules

Rule 10 are added (see Figure 2-2), then Rule 4 conflicts with Rule 9 in allowing user *B* to access the host *hc* with different access control rights ("lriw" and "lr" respectively) and Rule 3 conflicts with Rule 10 in allowing user group *Team2* to access the host *hb* with different access control rights ("lri" and "lriwdu" respectively). Another type of conflict would result from the carelessness of the administrator writing the rules. For example, if a Rule 11 has exactly the same specifications of user name, source address and destination host but a different set of access control rights as in Rule 8, then only one out of the two rules (8 and 11) is used (see Figures 2-1 and 2-2).

For these two types of conflicts, the two general conflict resolution principles are not sufficient to completely resolve the conflicts. An additional principle is used in these cases—a security rule overrides its preceding security rules. This principle is simple to comprehend, and allows the domain administrators to have control over the precedence of the security rules.

A Special Access Control Denial Specification

In order to further enhance the flexibility of the language for security rules, an extra access control right must be defined for specifying access denial (-). After the users and destination hosts have been grouped together and the security rules have been written for these groups, there may still be a few exceptions when a user in a group should not share with others in the group the same access rights to a specific destination host.

In one case, Rule 3 allows a user of group *Team2* from the subnetwork of "137.1.15" to access host *ha* of group *sunos* with access rights "lri". The administrators may want to grant user A from the subnetwork a tighter set of access rights—"lr"—on host *ha* than other users in group *Team2*. This additional restriction can be achieved by adding Rule 12 (see Figure 2-2).

In another case, the administrators may want to disallow user group *Team2* from the subnetwork "137.1" to access host *hb* of the destination group *sunos*. In this second case, an explicit specification of access denial is desirable, instead of replacing the security rule for the destination group with a number of security rules for each individual destination host except the excluding one. The administrators can express the explicit access denial by adding Rule 13 (see Figure 2-2). In order for this access denial to take effect, the design specifies that a rule for access denial (-) always wins in a conflicting situation. Therefore, Rule 13 overrides all other rules when a user of the group *Team2* from the subnetwork "137.1" to access the host *hb*, and Rule 13 successfully prevents group *Team2* from accessing host *hb*.

In general, the security rules specify the granted access control rights. If there is no rule for an attempted request, the request is denied. For example, user B from "137.33.2.1" trying to access host *hb* would fail according to the twelve rules in the previous examples, because none of them matches the attempt. Therefore, the explicit access denial (-) is recommended for use only in excluding a member in a group from sharing a security rule for the group.

Advantages of a Grouping Capability in the Syntax

This section acknowledges that there is some complexity added to the security rule specifications when a grouping capability is incorporated. The complexity comes from the potential conflicting effects induced by the grouping capability, as well as the fact that two more files—a user group file and a destination host group file—are needed. First, the grouping capability is an option, not a compulsory part, of the language syntax. If the number of users is small, then the rules may not need to be expressed using any user group name and the user group file does not necessarily exist. Similarly, if the number of destination hosts is small, then the rules may not need to be expressed using any destination host group name and the destination host group file does not need to exist. However, if the number of users is large or the number of destination hosts is not very small, then the users and destination hosts should be grouped according to the administrative policies set on them. The user group file and the destination host group file are then easy to write accordingly. The number of security rules can also be quite easily written without many conflicting rules. An example is shown in Appendix A. The number of the security rules can be largely reduced with the grouping capability maintained in the language. This simplification can practically reduce the chance of careless mistakes made by the administrators in configuring the security gateway. The usability of the specification language is thus largely increased.

2.4 Mapping of Access Control Rights to FTP Commands

A simple and flexible language has been defined for writing security rules to protect the file system of a local network domain. The subject and the object of the security rule are specifically chosen for use in a local network environment. The access control rights are extended from the AFS set of access control rights to facilitate a finer-granularity access control on the file system in a local network domain. This general set of access control rights still needs to be mapped to different FTP commands to enable the requesting action on the domain host file system. The table in Figure 2-3 shows the mapping of the access control rights to the FTP commands.[9]

In addition, the access control right to go up (u) affects the processing of the CWD, CDUP and PWD commands and works with other access control rights to allow FTP services to any linked files. Explanations for the effects of the right to go up (u) are discussed in detail in the Sections 2.6.4 and 2.6.6. The mapping of the designed fine-grain set of access control rights to the different FTP service commands shows that a finer-granularity of FTP service can be authorized to a subject by the

[9] The meaning of each FTP command is described in the FTP specifications, RFC 959 [16].

In order for a subject to request the following FTP commands on an object,	the subject needs to have the following access control right(s) on the object
LIST, NLST, STAT <pathname>	l
RETR	r
STOU, MKD	i
ALLO, RNTO	i OR w
STOR, APPE	w
RNFR, DELE, RMD	d
SMNT	m

Figure 2-3: Mapping of the Access Control Rights to the FTP Commands

security gateway.

The general access control right to administer (a) used in the AFS is not particularly useful for any command of the current version of FTP. This access control is useful, however, for a recommended command, change mode (CMOD), to be included in the next standard FTP specifications. This command allows a FTP user to change the access control modes/attributes of a file in the file system, and this action itself requires an access control right to administer.

Except for PASV and SITE, all other commands of the current FTP version are not explicitly controlled by the set of access control rights in the security rules; however, a user from a source address needs to match with one of the access-granting security rules in the domain gateway in order to successfully log into a destination host to make these unrestricted FTP requests. The reasons that the FTP commands, PASV and SITE, are disallowed will be discussed after the new FTP model is described.

2.5 The New FTP Model with Security Gateway

The standard FTP model specified in the RFC959 [16] maintains two connections between a FTP user and the FTP server: control connection and data connection. In the new secure FTP model using the MASS approach, an application gateway for security can

1. intercept both connections to enforce a fine-grain access control authorization on the FTP service, or
2. intercept only the control connection while letting the FTP server and the FTP user establish their own data connection, or
3. tap on the control connection and intervene when a restricted FTP service is attempted.

The second and the third methods are harder to design and implement than the first one even though they can potentially maintain the speed of FTP services better than the first one. The first method was chosen for the new FTP gateway design because it provides the most secure model by preventing any direct connections between a server and a user. An FTP user wanting to access a host in an administrative domain actually initiates a control connection to the domain security gateway for exchanging FTP commands and replies, and the security gateway opens another local control connection on behalf of the user to the destination host in the administrative domain.

2.5.1 General Processing of an FTP Command

In general, the security gateway refers to the local administrative security rules to check if an FTP user requesting FTP service on a local destination host has been granted the corresponding access control right. If not, the gateway responds to the FTP user immediately with a negative completion reply; otherwise, the FTP command is propagated to the destination host along the internal control connection established between the security gateway and the destination host. The reply to the FTP command is then issued from the destination host to the gateway, and the gateway propagates the FTP reply back to the FTP user. This is the general pattern for processing most of the FTP commands in the new FTP model with a security gateway in the middle of the FTP user and server. There are some exceptions which are discussed in Sections 2.5.2 and 2.6.

2.5.2 Processing Data-transferring FTP commands

There are several FTP commands that require the opening of an extra data connection for transferring data between an FTP user and an FTP server. They are the RETR, STOR, STOU, APPE, LIST and NLST commands. In the new FTP model, when the security gateway receives one of these commands, it performs the routine checking for the authorization of the user to request the command on the destination host. If the action is not authorized, the gateway immediately responds to the FTP user with a negative completion reply; otherwise, the gateway opens a data connection to the FTP user and also requests the opening of an internal data connection to the destination host. Data are transferred along both of the separated data connections, and the data are copied from one data connection to another in the gateway. In this setting, with both the control connection and the data connection terminated at the domain security gateway, the destination hosts in the local domain are protected from potential tampering along any direct connection from the outside network.

2.6 The New FTP Model with a Virtual File System for the Administrative Domain

A special feature is added in this new FTP model to facilitate file transfer access to different destination hosts in a local domain. In the above descriptions of different FTP command processing, one key design choice has not been mentioned: how to express the desired destination host once a FTP user has logged into the domain gateway. In one reported design of application gateway for security [12], the FTP user must issue a special gateway SITE command to request the gateway to open a proxy connection to the destination host for the user. This is a simple, straightforward design for expressing the destination host. In the new FTP model presented here, a subtly different approach is taken to for enhancing access to different hosts in the domain while not compromising the domain security. The processing of the CWD, CDUP and PWD commands do not follow the general processing pattern; they have been modified to implement a virtual file system which accesses different hosts in the local domain. The enhancement feature can be more easily explained and illustrated through some examples of how the CWD, CDUP and PWD commands work.

2.6.1 Initial Login Setting at the Gateway

In this new FTP model, once a user, let his user name be *david*, has been authenticated to the gateway, he logs in at the virtual root (top) directory, /.[10] Requesting LIST, NLST or "STAT /", user *david* can see a listing of all the public hosts in the domain. The public hosts are those included in the *public* host group specified in the host group file (see Section 2.3.2). In most cases, the LIST, NLST and "STAT <pathname>" commands are processed in the general pattern 2.5.1. Listing the root directory information is the only instance when the gateway does not check for the access control right to list (l) and automatically generates a listing of the host names for replying to the LIST, NLST and "STAT /" commands. In addition, when a user is at the virtual root, none of the commands that require one of the "riwdam" rights is allowed. This limitation on the FTP service at the gateway machine itself protects the many security files stored in the gateway machine from outside tampering.

2.6.2 Using CWD to Access a Destination Host

Now if user *david* at the virtual root wants access to a destination host, let its name be *hx*, he needs to request "CWD *hx*". The gateway checks the security rules to see if user *david* has the any access control right to the host *hx*. If not, the gateway immediately responds to the FTP user with a negative completion reply; otherwise, the gateway uses the password provided by user *david* at the gateway login to try to open a connection to the host *hx* on behalf of user *david*. The host *hx* decides whether user *david* and his password is acceptable. If not, the gateway replies a negative response to the CWD attempt, and leaves user *david* at his original directory. On the other

[10] The symbol for the root directory is different for different operating system, for example, / in UNIX and \ in MS-DOS. The UNIX notation is used throughout the rest of the discussions.

hand, if the host *hx* accepts the FTP connection for user *david*, an internal control connection between the host *hx* and the gateway is established for user *david*. Once the host *hx* has allowed user *david* to log in, the gateway continues to check to see if user *david* has a home directory at the host *hx* or the access right to go up (u). If user *david* lacks both conditions, the internal control connection between the host *hx* and the gateway for user *david* is closed and a negative response to the CWD attempt is replied; otherwise, the gateway notifies the FTP user that "CWD *hx*" is successful.

User *david* may have a different password for logging into the security gateway than that used for logging into his desired destination host, *hx*. In this case, user *david* must notify the security gateway that he wants to use a different password for the "CWD *hx*" request.[11] In this design, the notification is accomplished by issuing the FTP login sequence once again. User *david* first sends a "USER *david*" command to the gateway. The gateway, knowing that user *david* has already been authenticated, maintains his current login information and prompts user *david* to send his password using a PASS command. The new password is stored for future attempts to establish connections to other local destination hosts in the domain. Instead of using a less confusing new SITE command, the standard FTP login sequence of USER and PASS commands is used in this scenario simply because displaying the sensitive password information included in any new SITE command cannot be prevented on the client host machine. For example, if a simple "SITE PASS" can be used by an FTP user to change her/his password for later "CWD <destination-host>" attempts during the FTP session, the client machine cannot be expected to handle this SITE command specially by hiding the password typed in following "SITE PASS". Displaying a password plainly on any machine screen is strongly not discouraged; therefore, the new FTP model relies on the client FTP implementation of the PASS command, which hides the sensitive password information according to the standard recommendations [16].

[11] The security gateway can also use a non-traditional way of authenticating a user to another local host, instead of requiring the user from a remote site to send the user's correct password to the destination host. For example, the security gateway can be a trusted server to authenticate the user to another local host using the Kerberos authentication scheme.

2.6.3 PWD at a Destination Host

Once user *david* has logged into the host *hx*, he is initially at his home directory of the host *hx* if he has one; otherwise, the host *hx* decides at which directory user *david* is initially located at. A PWD command request is propagated through the gateway to the host *hx*, and the current working directory pathname is included in the reply given by the host *hx*. This pathname is processed at the gateway, and the processing depends on whether the administrators' security rules grant user *david* the access control right to go up (u) at the host *hx*. If not, the gateway hides user *david*'s home directory pathname prefix at the host *hx*, and notifies the FTP user that he is simply at the directory */hx*.[12] On the other hand, if user *david* has the access control right to go up (u) at the host *hx*, then the pathname in the reply from the host *hx* is appended with the prefix */hx* and the combined pathname is used in a reply from the gateway to the FTP user. For example, if the home directory of user *david* at host *hx* is */user/david*, then user *david* sees */hx/user/david* as the reply to his PWD request. In either case, the first part of the pathname in the reply to the PWD command shows the destination host at which the user is currently working.

The design uses the UNIX notation of root directory and subdirectory separating symbol, but falls short of converting other subdirectory separating symbols used in non-UNIX operating systems. For example, if host *hx* runs MS-DOS, user *david* sees either */hx\user\david* or */hx* depending on whether he has the access right to go up (u) or not. Suggestions for converting path names to a standard format in the virtual file system is discussed in Chapter 5 for future extension of the design.

2.6.4 CDUP at a Destination Host

If user *david* executes the CDUP command when he is still at his home directory at host *hx*, the gateway takes different actions depending on whether user *david* has the access control right to go up (u) at the host *hx*. If not, the gateway suspends

[12] As mentioned in Section 2.6.2, if user *david* does not have a home directory at the host *hx*, he must have an access control right to go up (u) in order to successfully access the host *hx*, and thus the path name hiding procedure is not applied.

its connection to the host *hx* for user *david*, and user *david* is taken back from the */hx* directory to the virtual root directory, /. On the other hand, if user *david* can go up at the host *hx*, the CDUP command is propagated to the host *hx* through the gateway. According to the example in the previous paragraph, user *david* goes from the */hx/user/david* directory to the */hx/user* directory, that is, the */user* directory at the host *hx*. It should be noted that user *david* reaches a directory beyond the subtree of his home directory at the host *hx*. In this latter case, user *david* may continue to CDUP to the */hx* directory, that is, the / directory at the host *hx*. Then, another CDUP command requested by user *david* suspends the gateway connection to the host *hx* for user *david* and he is taken back to the virtual root directory / at the gateway.

2.6.5 More Specifications for the Directory Listing at the Virtual Root

Once a user has successfully made an FTP connection to a destination host in the domain, the host name should be included in the directory listing at the virtual root, even if the host is not administratively declared to be in the *public* host group. Consequently, the directory listing at the virtual root may have one more entry than it had in the very beginning of the FTP session. The *public* host group is designed to hide all the unreached, private hosts in the administrative domain. Once a user successfully reaches a private host with correct login information, there is no need to hide the fact that this private host is one of the hosts in the domain. Moreover, the detailed listing information given by the "LIST /" and "STAT /" commands indicate which hosts are already connected. All these additional specifications serve to clarify the status of a user's connections in the new FTP model with the virtual file domain system.

2.6.6 More Specifications for CWD

In Section 2.6.2, the application of the CWD command at the virtual root directory to access to a destination host has already been discussed. There are still two more specifications for the usage of the CWD command needed in the virtual file system model: CWD with an absolute addressing pathname as its argument, and CWD at a destination host without access control right to go up.

CWD with an Absolute Addressing Pathname as Its Argument

If the pathname argument of the CWD command is an absolute addressing pathname,[13] the first part of the pathname, such as *p1* in */p1/p2/p3*, must be treated as the name of a destination host in this virtual file system model. With the example of a pathname argument */p1/p2/p3*, the gateway first tries to connect to the specified destination host *p1* on behalf of the user. If the connection attempt is successful, a change of directory to *p2/p3* relative to the user's home directory at the host *p1* is requested. For example, if the change of directory to the whole pathname, */p1/p2/p3*, is successful, the gateway suspends the connection for user *david* to the original host and puts user *david* at the directory *p2/p3* under his home directory at the host *p1*. According to the specification of PWD processing in Section 2.6.3, a PWD command tells user *david* that he is (virtually) at */p1/p2/p3* if he does not have the access control right to go up (u) at the host *p1*; otherwise, he sees himself at */p1/user/david/p2/p3* if */user/david* is the pathname of his home directory at host *p1*. On the other hand, if the change of directory fails anywhere along the whole pathname argument, the gateway returns a negative reply, and leave the user at the his original directory of the original host, where he attempts the CWD command. The same process is taken for "CWD *p1/p2/p3*" when the user is at the virtual root directory. Although the pathname is not an absolute addressing pathname, the fact that the user is at the virtual root directory implies that any pathname argument for the CWD command

[13] An absolute addressing pathname is a complete pathname of a directory with a global reference point at the front, that is, any pathname starting with the root directory notation at the beginning of the pathname, such as */hx/user/david* and */hy/u* in the UNIX notation.

here is to be treated as an absolute addressing pathname.[14]

CWD at a Destination Host without Access Control Right to Go Up

If the user is at a destination host and the pathname argument for the CWD command is a relative addressing pathname,[15] the change of directory service is taken with respect to the current working directory of the destination host at which the user is currently located. The gateway must pay special attention to ensure that a user having no access control right to go up (u) in the current destination host does not change to a directory beyond the subtree of his home directory at the host. The specification for the processing of the CDUP command in discussed in Section 2.6.4 has already taken care of the user's attempt to go above the user's home directory subtree when he is not allowed to do so. There are, however, some operating systems which allow the capability of linking of one directory to another.[16] In this case, there may be directories under the user's home directory which are actually linked to other directories beyond the user's home directory subtree. The gateway disallows attempts of changing to such a directory if the user does not have the access control right to go up (u) at the destination host.[17]

The enhancements made to the directory listing at the virtual root, as well as the commands CWD, CDUP and PWD, enable the use of the virtual file system approach to access the hosts of a local domain. This approach is a highly attractive feature provided in the new secure FTP model.

2.6.7 Disabled FTP Commands in the New Model

In this new secure FTP model, which contains a security gateway and a virtual domain file system, two standard FTP commands are disabled for simplicity: the PASV and

[14] An absolute addressing pathname is a pathname with the prefix of the root directory pathname.

[15] A relative addressing pathname is a pathname without a global reference point, that is, any pathname that does not start with the root directory notation at the beginning of the pathname. The relative addressing pathname is the counterpart of the absolute addressing pathname.

[16] For example, UNIX allows such capability with user command, *ln -s*.

[17] Listing, reading, appending or overwriting a linked plain file is also disallowed.

SITE commands.

The PASV command enables the FTP user to request the FTP destination host to listen to and wait for a third-party data connection. In order to handle this third-party data connection set up for this command, the new FTP model with a security gateway in the middle between the FTP user and server would have to be very complicated. The security of the destination host in the local domain might also be compromised through a third-party data connection if it were not carefully managed. For simplicity and security reasons, the PASV command is disabled in the new FTP model.

The SITE command is used to provide services specific to a local destination host in addition to the standard FTP services. Different hosts in an administrative domain may allow different SITE commands. The capability to handle this wide range of SITE commands would make the security gateway unnecessarily complicated, therefore the SITE command is disabled for simplicity.

2.7 Summary on the Overall New Design

The overall design covers all the essential ideas listed in Section 2.1. First, it follows the MASS approach for network security by employing a security gateway to enforce the administrator-determined security policies on the domain hosts. Second, it enhances the FTP security with a powerful language especially designed for specifying the security policies on a local network domain. This descriptive language is simple, yet flexible and elegant, with a capability to express a group of subjects and objects in the access control rules. A fine-grain set of access control rights has also been expanded from the AFS set of access control rights to provide finer-granularity of access control on the FTP services to different local hosts. Third, a virtual file system model has been developed to largely facilitate FTP service to different hosts in an administrative domain.

Chapter 3

Implementation Notes

An administrative domain FTP gateway program is implemented based on the design described in Chapter 2. The prototype's implementation is entirely software which runs on a UNIX-based workstation.[1] Generally speaking, the implementation for the designed FTP gateway program is a "half-server, half-client" FTP program which incorporates the administrative security rules and enables proxy FTP service using the virtual file system approach.

3.1 Incorporation of the Administrative Security Rules

In order to incorporate the new administrative security rules for the gateway FTP service, the implementation must be able to

- parse the administrative configuration files,
- generate the internal data structures for keeping the parsed access control information, and
- apply the information stored in the internal data structures to limit FTP services.

[1] To be exact, the prototype implementation runs on an Ultrix-based DECstation.

3.1.1 Parsing of the Administrative Configuration Files

There are three administrative configuration files described in the Section 2.3:

- the security rules file, *secu.rul*,
- the user groups file, *user.grp*, and
- the destination host groups file, *dest.grp*.

These files together contain the administrative information for a finer-granularity access control of the FTP service to the local hosts.

The FTP gateway program starts by parsing the *user.grp* and *dest.grp* files and storing the group listings in two internal data structures (see Section 3.1.2). The absence of the *user.grp* file invalidates every security rule with a group user as the subject. Similarly, the absence of the *dest.grp* file invalidates each security rule with a destination host group as the object. The FTP gateway program then parses the *secu.rul* file in order to generate the principal internal data structure, called the security rules database, *securuldb* (see details in Section 3.1.2). This *securuldb* retains all the access control information from the *secu.rul* file. The absence of the *secu.rul* file terminates the FTP gateway program because no subject is granted the right to use the FTP service to any of the local hosts.

Sections 2.2 and 2.3 describe the basic model and the syntax details for the security rules contained in the configuration files. The four fields of a security rule are separated by white space, and each security rule in the *secu.rul* file must end with a newline character. As the program parses each line of the *secu.rul* file, the program checks the correctness of each field according to the security rule syntax. A line of the *secu.rul* file is ignored if it misses a field or fails on any field, and an error message is printed for the scanned line. The parsing of the *secu.rul* file continues on the next line. Any contents following the four correctly parsed fields of each line are ignored and treated as comments. The information of each successfully parsed security rule is retained in the *securuldb* (see Section 3.1.2). If not a single line from the *secu.rul* file can be successfully parsed, the FTP gateway program terminates because no subject is granted the right to use the FTP service to any of the local hosts.

3.1.2 Internal Data Structures for Keeping Access Control Information

The three configuration files are parsed once at the start of the FTP gateway program, and the information retrieved from them is stored in three internal data structures for use throughout the execution of the program. These data structures are:

- the security rules database, *securuldb*,
- the user groups database, *usergrpdb*, and
- the destination host groups database, *destgrpdb*.

User Groups and Destination Host Groups Databases

The *usergrpdb* is a linked list which keeps the information on the member listing of different user groups. The group listing information is parsed from the *user.grp* file. Similarly, the *destgrpdb* is a linked list which keeps the information on the member listing of different destination host groups, and is generated using the parsed information from the *dest.grp* file.

Security Rules Database

The security rules database, *securuldb*, is the principal internal data structure for keeping the access control information contained in the *secu.rul* file. The internal arrangement of the *securuldb* is specially designed to prepare for resolving any conflict of rules according to five conflict resolution strategies.

1. Any explicit access denial specification always overrides other rules in a conflicting situation.

2. A security rule with a specifically matched user field overrides one with a group user.

3. A security rule with a more specifically matched source address field overrides one with a less specifically matched source address field.

4. A security rule with specifically matched destination host field overrides one with a host group.

5. A security rule overrides its preceding security rules in the *secu.rul* file.

In resolving rule conflicts, the five strategies are prioritized in the order in which they are listed. The five strategies follow closely the conflict resolution principles laid out in Section 2.3.3. The *securuldb* contains the security rules information sorted according to these strategies. In general, the security rule information found earlier in the *securuldb* overrides that found later in the *securuldb*.

As the program parses the *secu.rul* file, each parsed security rule applying to an individual user is inserted directly into the *securuldb*. On the other hand, for each parsed security rule applying to a user group, the program first checks with the *usergrpdb* to see if the user group has been recorded in the *user.grp* file. If not, the security rule is ignored; otherwise, the security rule is forked into several security rule instances. Each forked security rule instance carries the same access control information as its parent rule, except that the user group name is replaced by a member of the user group. Each forked security rule instance is then inserted into the *securuldb*. For example, Rule 1 of the sample set of rules used in Section 2.3.3 is first forked into three different security rule instances (see Figure 3-1), then each of the three forked security rule instances is inserted into the *securuldb*.

Original	Rule 1:	#G:Team2	137.1.0.0	16	#G:sunos	1
Forked	Rule 1a:	A	137.1.0.0	16	#G:sunos	1
	Rule 1b:	B	137.1.0.0	16	#G:sunos	1
	Rule 1c:	C	137.1.0.0	16	#G:sunos	1

Figure 3-1: The Set of Forked Security Rule Instances for Rule 1

Having forked each security rule applying to a user group into several of its constituent instances, the security rule instances inserted into the *securuldb* can be grouped by their individual user fields. Each group of security rule instances apply to the same user. The group thus forms the list of capabilities for that user. The security rule instances in a user capability list are sorted according to the five conflict resolution strategies. In a user capability list,

1. security rule instances specifying access denial (-) are grouped before other access-granting security rule instances;

2. in the subgroup of access-granting security rule instances, the instances originally applying to the individual user are grouped are further grouped before those originally applying to a user group (in which the user is a member);

3. in each subgroup of security rule instances, the instances matching fewer source addresses (more specific on the source address field) are further grouped before those matching more source addresses (less specific on the source address field);

4. in each subgroup, the security rule instances applying to a single destination host (more specific on the destination host field) are put in front of those applying to a destination host group (less specific on the destination host field);

5. for any two security rule instances with the same source address field,

   - the one found later in the *secu.rul* file replaces the one found earlier if the two have the same destination host field, or

   - the one found later in the *secu.rul* file precedes the one found earlier if the two have different destination host fields.

These five criteria for the internal arrangement of the *securuldb* facilitate the five conflict resolution strategies. One concrete illustrative example of the *securuldb* internal arrangement is shown in Figure 3-5. This *securuldb* keeps the access control information from the set of sample security rules found in Section 2.3.3 (see Figures 3-2, 3-3 and 3-4).

Rule 1:	#G:Team2	137.1.0.0	16	#G:sunos	l
Rule 2:	#G:Team2	137.1.0.0	16	hc	lr
Rule 3:	#G:Team2	137.1.15.0	24	#G:sunos	lri
Rule 4:	#G:Team2	137.1.15.0	24	hc	lriw
Rule 5:	C	137.1.0.0	16	#G:sunos	lriwd
Rule 6:	C	137.1.0.0	16	hc	lriwda
Rule 7:	C	137.1.15.0	24	#G:sunos	lriwdau
Rule 8:	C	137.1.15.0	24	hc	lriwdaum
Rule 9:	#G:Team3	137.1.15.0	24	hc	lr
Rule 10:	#G:Team2	137.1.15.0	24	#G:servers	lriwdu
Rule 11:	C	137.1.15.0	24	hc	lr
Rule 12:	A	137.1.15.0	24	ha	lr
Rule 13:	#G:Team2	137.1.0.0	16	hb	-

Figure 3-2: Sample Security Rules File, *secu.rul*

```
Team2:A,B,C
Team3:B,D
```

Figure 3-3: Sample User Groups File, *user.grp*

```
sunos:ha,hb,hc
servers:hb,hd
```

Figure 3-4: Sample Destination Host Groups File, *user.grp*

user	from	group	to	rights	granted by
D					
	137.1.15.0	24	hc	lr	Rule 9
C					
	137.1.0.0	16	hb	-	Rule 13
	137.1.15.0	24	hc	lr	Rule 11
	137.1.15.0	24	#G:sunos	lriwdau	Rule 7
	137.1.0.0	16	hc	lriwda	Rule 6
	137.1.0.0	16	#G:sunos	lriwd	Rule 5
	137.1.15.0	24	hc	lriw	Rule 4
	137.1.15.0	24	#G:servers	lriwdu	Rule 10
	137.1.15.0	24	#G:sunos	lri	Rule 3
	137.1.0.0	16	hc	lr	Rule 2
	137.1.0.0	16	#G:sunos	l	Rule 1
B					
	137.1.0.0	16	hb	-	Rule 13
	137.1.15.0	24	hc	lr	Rule 9
	137.1.15.0	24	#G:servers	lriwdu	Rule 10
	137.1.15.0	24	#G:sunos	lri	Rule 3
	137.1.0.0	16	hc	lr	Rule 2
	137.1.0.0	16	#G:sunos	l	Rule 1
A					
	137.1.0.0	16	hb	-	Rule 13
	137.1.15.0	24	ha	lr	Rule 12
	137.1.15.0	24	hc	lriw	Rule 4
	137.1.15.0	24	#G:servers	lriwdu	Rule 10
	137.1.15.0	24	#G:sunos	lri	Rule 3
	137.1.0.0	16	hc	lr	Rule 2
	137.1.0.0	16	#G:sunos	l	Rule 1

Figure 3-5: Security Rules Database, *securuldb*

3.1.3 Applying the Access Control Information

The three internal data structures discussed are used for keeping the overall access control information described in the three configuration files. Another type of internal data structure is created each time an FTP user has successfully been authenticated to the gateway. This data structure contains the capability list for the subject—the specific FTP user from the specific source address. This list, let its name be *subjcapls*, describes which destination hosts the subject may access to throughout the FTP session. The *subjcapls* also describes what granted access control rights the subject may access to the destination hosts. If no access-grant capability is found for a subject, the FTP user is logged out because the user from the specific source address (the subject) is denied of the FTP service to any local host.

A *subjcapls* is generated according to the information contained in the *securuldb*. In generating a *subjcapls*, the program searches through the *securuldb* in order to look for capabilities for the subject and append to the *subjcapls*. When a capability for a host group is found, the program first checks with the *destgrpdb* to see if the destination host group has been recorded in the *dest.grp* file.[2] If not, the capability is ignored; otherwise, the capability is forked into several capabilities. Each forked capability carries the same access control information as its parent, except that the destination host group name is replaced by a member of the destination host group. Each forked capability is then appended to the *subjcapls*.

Moreover, the *subjcapls* contains only one capability for each destination host accessible by the subject. A newly found capability is appended to the *subjcapls* only if a capability for the destination host does not already exist in the *subjcapls*. This strategy for generating the *subjcapls*, in effect, makes use of the special internal arrangement of the *securuldb* to resolve any rule conflicts by assigning the capabilities found earlier in the *securuldb* a higher priority to the ones found later.

[2] There is a tradeoff between expanding the host group at the end of user authentication time and expanding the host group at the program startup time. If the host group were expanded like the user group at the program startup time, the *securuldb* would be much larger. On the other hand, when the host group is expanded at the end of user authentication time, the slow down caused by generating the *subjcapls* can become noticeable to the FTP user.

Subject	host	rights	granted by
user C from 137.1.15.3:			
	hb	-	Rule 13
	hc	lr	Rule 11
	ha	lriwdau	Rule 7
	hd	lriwdu	Rule 10

Figure 3-6: Subject Capability List, *subjcapls*, for User C from "137.1.15.3"

Subject	host	rights	granted by
user C from 137.1.8.9:			
	hb	-	Rule 13
	hc	lriwda	Rule 6
	ha	lriwd	Rule 5

Figure 3-7: Subject Capability List, *subjcapls*, for User C from "137.1.8.9"

Using the *securuldb* found in Figure 3-5, the authenticated user *C* from "137.1.15.3" has a *subjcapls* generated by the gateway FTP program (see Figure 3-6). User *C* can use the capabilities of this *subjcapls* throughout the FTP session. From the administrators' perspective, the *subjcapls* serves to limit the actions of user *C* from "137.1.15.3" on the local hosts according to the mapping of access control rights to FTP commands described in Section 2.4. When user *C* logs out, the corresponding *subjcapls* is destroyed. Another example of *subjcapls* is also shown in Figure 3-7 for a different subject—user C from "137.1.8.9".

3.2 Implementing the New FTP Model

Generally speaking, the gateway FTP program simply implements the new FTP model according to the design described in Sections 2.5 and 2.6. More specifically, there are several choices made for the fine details of the implementation, which are presented in the following subsections.

3.2.1 Connection State for an FTP Session

In order to apply the proxy FTP service in the virtual file system model, connection information must be maintained. After a subject—a user from a source address—has been authenticated to the FTP gateway program. three internal data structures are created: the gateway connection state block, *gcsb*; a list of destination connection state blocks, *dcsbl*; and the subject capability list, *subjcapls.*

First, the *gcsb* keeps the information on:

- the single FTP control connection from the gateway to the FTP client host, *gcsb.cg_control*,

- the opened FTP data connection from the gateway to the FTP client host, *gcsb.cg_data*,

- the currently accessing destination host, *gcsb.cur_dest*, and

- the last-provided user password, *gcsb.passwd.*

Second, for each destination host accessible by the subject, there is one block *dcsb* maintained in the *dcsbl* list for keeping the information on:

- the single FTP control connection from the gateway to the destination host, *dcsbl.gd_control*,

- the opened FTP data connection from the gateway to the destination host, *dcsbl.gd_data*,

- the home directory of the user at the destination host, *dcsb.home_dir*,

- the publicity of the destination host, *dcsb.public*, and

- the type, the mode and the structure for data transfer between the gateway and the destination host, *dcsb.params.*

Third, the subject capability list, *subjcapls* (see details in Section 3.1.3), is referred to whenever the gateway FTP program needs to find out whether the FTP user has the access control right to request a specific FTP service (see Sections 2.4 and 2.5.1).

These three data structures are often referred to in the following subsections when more implementation fine details are discussed. Throughout those subsections, the sample set of security rules found in Sections 2.3.3 and 3.1.2 are used with an additional destination group, "public:ha,hb", defined in the *dest.grp* file. Using the *securuldb* derived from these security rules (see Figure 3-5), three data structures for connection states are created for an authenticated subject, let it be user B from "137.1.15.3", to use throughout its FTP session. The initial states of the three data structures for user B from "137.1.15.3" are shown in Figure 3-8. In the following subsections, the usage of these data structures are explained in several illustrative examples with user B from "137.1.15.3" as the subject.

3.2.2 Initial Virtual Root Directory Listing

As described in Section 2.6.1, the directory list at the virtual root gives the list of hosts accessible to user B from "137.1.15.3" (the subject). Initially, the listing reads:

```
lrwxr-xr-x   1   root   wheel   1   Jun   30   1994   ha ↦ @
lrwxr-xr-x   1   root   wheel   1   Jun   30   1994   hb ↦ @
```

The implementation chooses to imitate the format of directory listing used in UNIX [21]. Only the public hosts (those with *dcsb.public* set to be true) are listed at the start of the FTP session. The non-public destination hosts *hc* and *hd* are hidden even though the gateway program knows from *subjcapls* that user C has access rights to them.[3]

3.2.3 Implementing CWD to Access a Destination Host

Sections 2.6.2 and 2.6.6 discuss how the CWD command is modified to facilitate accessing different destination hosts in the local domain using a virtual file system approach. The implementation checks with the *subjcapls* to validate the user's attempt to change into a new destination host. For example, user B from "137.1.15.3"

[3] In reply to NLST, only the names are listed.

gcsb:			
	cg_control	[created stream]	
	cg_data	*empty*	
	cur_dest	*empty*	
	passwd	[gateway password]	
subjcapls:			
	hb	-	(Rule 13)
	hc	lr	(Rule 9)
	hd	lriwdu	(Rule 10)
	ha	lri	(Rule 3)
dcsbl:			
	hb	gd_control	*empty*
		gd_data	*empty*
		home_dir	*empty*
		public	true
		params	*empty*
	hc	gd_control	*empty*
		gd_data	*empty*
		home_dir	*empty*
		public	false
		params	*empty*
	hd	gd_control	*empty*
		gd_data	*empty*
		home_dir	*empty*
		public	false
		params	*empty*
	ha	gd_control	*empty*
		gd_data	*empty*
		home_dir	*empty*
		public	true
		params	*empty*

Figure 3-8: Initial Connection State for User *B* from "137.1.15.3"

gcsb:		
	cg_control	[created stream]
	cg_data	*empty*
	cur_dest	ha
	passwd	[last-provided password]
dcsb for ha:		
	gd_control	[created stream]
	gd_data	*empty*
	home_dir	/user/b
	public	true
	params	[initial parameters]

Figure 3-9: Updated Connection State after "CWD */ha*"

can change directory to */ha*, */hc* and */hd*, but not *hb* or any local host not listed in the *subjcapls* (see Figure 3-8).

After the gateway FTP program successfully logs a FTP client into the local host, the program immediately sends a PWD command to the destination host to inquire about the user's home directory at the destination host, and the *dcsb* is updated in response to the host's reply. For example, after user *B* just logs into the gateway FTP program, a "CWD */ha*" command is issued. The FTP gateway program then tries to establish a proxy connection to host *ha* for user *B*. Having logged user *B* into host *ha*, the program prompts host *ha* to identify the home directory path name of user *B* at host *ha*, and the result, let it be */user/b*, is stored in the corresponding *dcsb.home_dir*. The program also updates the *gcsb.cur_dest*, to keep track of the current host user *B* is accessing, and the corresponding *dcsb.gd_control*, to keep track of the connectivity to a host. As a result, the *gcsb* and *dcsb* for *ha* are updated (see Figure 3-9). This sequence of procedures is taken only once when the user first logs into a new host. Subsequent change of directory to the same host makes use of the information stored in the *dcsb* for the host.

As mentioned in Section 2.6.2, "CWD */ha*" fails if user *B* has a different password for logging into the security gateway than that used for logging into host *ha*. In that case, the user may change the stored *gcsb.passwd* by executing the login sequence of USER and PASS without affecting the user's authentication at the gateway. Precautions are taken, however, to limit the number of times this USER–PASS log in sequence may be executed. Moreover, the program limits the number of times an FTP user may attempt to log into the same new destination host. These restrictions prevent intruders from recklessly trying different passwords for different local hosts.

3.2.4 Implementing PWD for the Virtual File System Approach

Section 2.6.3 describes the responses to the PWD in the FTP model with a virtual file system approach. The gateway FTP program implements accordingly and uses the information in *gcsb.cur_dest* and the *dcsb.home_dir* and *subjcapls* of the corresponding host. For example, using the information in Figure 3-9 and user *B*'s *subjcapls* for *ha* (which does not have access right to go up), the */ha* is the response to the PWD command. Now if user *B* executes “CWD */hd/mail*” and the command is successful after following the instructions described in the first half of Section 2.6.6, then the *gcsb* and *dcsb* for *ha* are updated as shown in Figure 3-10, and the response to a PWD command is */hd/u/bb/mail* for user *B* has an access right to go up (u) at host *hd* according to the *subjcapls*. Furthermore, if user issues “CWD */*”, the *gcsb.cur_dir* is nullified, which indicates that the user is currently at the virtual root directory.

3.2.5 More Notes on Virtual Root Directory Listing

Having successfully logged into hosts *ha* and *hd*, user *B* sees the virtual directory listing to be:

```
lrwxr-xr-x  1  root  wheel  1  Jun  30  1994  ha ↦ ~
lrwxr-xr-x  1  root  wheel  1  Jun  30  1994  hb ↦ @
lrwxr-xr-x  1  root  wheel  1  Jun  30  1994  hd ↦ /u/bb
```

This listing differs from the initial listing shown in Section 3.2.2 in two ways. First, a host pointing to the symbol @ indicates that a host is not connected yet for the user. Therefore, both hosts *ha* and *hb* were pointing to the symbol @ in the initial virtual

gcsb:		
	cg_control	[created stream]
	cg_data	*empty*
	cur_dest	hd
	passwd	[last-provided password]
dcsb for hd:		
	gd_control	[created stream]
	gd_data	*empty*
	home_dir	/u/bb
	public	true
	params	[initial parameters]

Figure 3-10: Updated Connection State after “CWD */hd*”

root directory listing. Now *ha* is connected and is no longer pointing to the symbol @, but is pointing the symbol ~, signifying that the user can access the home directory at the host *ha*. Second, the listing now has an entry for host *hd* which was not shown in the initial listing. The previously non-public host *hd* is “discovered” by user *B* after user *B* has been successfully authenticated to host *hd* with a correct password; as a result, host *hd* is listed now.[4] Moreover, host *hd* in the directory listing points to the *dcsb.home_dir* of user *B* at host *hd* signifying user *B* can access any directory of host *hd* at which user *B* has the access right to go up (u). These implementation choices follow the design laid out in Section 2.6.5.

3.2.6 Path Name Analysis

According to the standard FTP specifications in RFC 959 [16], there are many commands taking a path name as an argument. In the new FTP model with a virtual file system approach, a gateway FTP program must be carefully analyze any path name argument in order to match the intention of the user providing the argument in the virtual file system setting.

[4] Figure 3-10 carefully shows that *dcsb.public* is reset to true for *hd* after user *B* has successfully logged into host *hd*.

Argument for CWD

The second half of Section 2.6.6 illustrates with examples some complications of using different types of path name arguments for the CWD command. The CWD command is implemented accordingly to take in these different types of path names as its arguments. Moreover, the implementation of the CWD command carefully deals with two special substrings in an UNIX path name, / and "..". The first is used as the name of the virtual root directory as well as the directory separator in a path name; the second is used as the name of the parent directory [21].

Upon receiving a path name argument for CWD, the implementation first reduces sequence of / as a single /. For example, *///hd////mail//* is reduced to /hd/mail/. This interpretation is entirely an implementation choice.

The implementation then searches the reduced path name for ../ in the beginning, /.. in the end and /../ anywhere in the path name, and splits the path name into substrings of path name which are either entirely a parental path name or totally free of of any parental path name. For example, *../ha/mail/inbox/../drafts/..* is split into "..", *ha/mail/inbox*, "..", *drafts* and "..". The program then carries out a CWD for each substring path name. The "CWD .." command is treated the same as the CDUP command described in Section 2.6.4. If any CWD command for the subpath name fails, the CWD command for the entire path name fails, and the user is taken back to the original destination host and directory.

The special treatment for the ".."-string in the CWD path name argument must be in place to carry out the intention of the user in the virtual file system setting, as well as preventing a user without the access right to go up (u) to reach directories other those under the user's home directory. For a user without the access right to go up (u), the gateway FTP program sends an additional PWD request to the destination host before replying the user. The gateway FTP program checks the PWD reply from the destination host against the corresponding *dcsb.home_dir* to see if the user has reached directories other those under the user's home directory at the destination host. If not, the gateway returns a positive reply to the user's CWD command; otherwise, the user is taken back to the original destination host and directory and the gateway returns a negative reply to the user's CWD request.

Path Name Argument for Other Commands

There are many other FTP commands which take in a path name argument: SMNT, RETR, STOR, APPE, RNFR, RNTO, DELE, RMD, MKD, LIST, NLST and STAT. The last three list the information of the path argument, and the rest manipulate the file represented by the path name in the argument.

For simplicity of enforcing access control, the path name argument for all these file-manipulating FTP commands are restricted to a "single-level" path name with no / and "..". The implementation relies on the CWD and the CDUP commands to enforce the access right to go up (u). If the user wants to manipulate a file, the user must change to the appropriate directory to do so. When executing the CWD command, the implementation can check whether the access right to go up (u) is required for the user to change to the desired directory.

The same restriction applies to the commands LIST, NLST and STAT except that a single / or a single ".." is allowed for use in its argument. The commands LIST, NLST and STAT that have a single / argument request the directory listing of the virtual root (see Sections 3.2.2 and 3.2.5. The determination of the parent directory for listing in the command "LIST ..", "NLST .." or "STAT .." follows the same guidelines that are used for the CDUP command (see Section 2.6.4).

In most cases, this arrangement is good for preventing a user without access right to go up (u) from manipulating and accessing files under the user's home directory. However, sometimes a linked file needs to be restricted. For simplicity of implementation, the program restricts a user without the access right to go up (u) to list (l), read (r) or write-over (w) any linked file. If the target of the linked file is under the user's home directory, the user should instead change the appropriate subdirectory; otherwise, the user should not be allowed to the target file outside of the user's home directory.[5]

[5] The gateway FTP program performs the checking by sending a STAT command a to check if the file is a linked file.

gcsb:		
	cg_control	[created stream]
	cg_data	[created stream]
	cur_dest	hd
	passwd	[last-provided password]
dcsb for hd:		
	gd_control	[created stream]
	gd_data	[created stream]
	home_dir	/u/bb
	public	true
	params	[initial parameters]

Figure 3-11: Updated Connection State after a PORT command

3.2.7 Implementing Data Transfer

Section 2.5.2 discusses the processing of data-transferring FTP commands: RETR, STOR, STOU, APPE, LIST and NLST. Two new data connections are established every time one of these commands is requested with a preceding PORT command, and they are kept in the *gcsb.cg_data* and the corresponding *dcsb.gd_data.* For example, when user *B* is currently located at host *hd*, and one of the data-transferring FTP commands is requested, the *gcsb* and the *dcsb* for *hd* are updated (see Figure 3-11).

The gateway does not interpret the type of data being transferred, but simply copies data from one connection stream to another. For the RETR, LIST and NLST commands, the data are copied from the corresponding *dcsb.gd_data* to the *gcsb.cg_data*; on the other hand, for the STOR, STOU and APPE commands, the data are copied from the *gcsb.cg_data* to the corresponding *dcsb.gd_data.*

The type, the mode and the structure of data in transfer are interpreted by the two end-to-end hosts, the FTP client host and the FTP server host. If a user wants to change the parameter for transferring data, the user sends a TYPE, STRU or MODE command to the destination host through the FTP gateway. The FTP gateway program does keep track of the parameters for data transferring for each destination host in the corresponding *dcsb.params.* For example, user *B* is currently accessing *dcsb* for ha:

gd_control	[created stream]
gd_data	[created stream]
home_dir	/user/b
public	true
params	type=I,[initial mode and structure]

Figure 3-12: Updated Connection State after "TYPE I"

host *ha*, and user *B* changes the type for data transfer to image type by issuing the "TYPE I" command [16], the *dcsb* for *ha* is then updated (see Figure 3-12). The program, however, keeps track of this information only for notifying which parameters the destination host is using for data transfer.

3.2.8 Gateway FTP for Multiple Users and Multiple Servers

Like other FTP server programs, the gateway FTP program is intended for multiple concurrent users. In the current implementation, a new UNIX process is forked for each logged-in user [21]. There are one control connection and one data connection between each FTP user and the FTP gateway, but the FTP gateway can establish multiple pairs of FTP control/data connections to different servers for the same FTP user. As a result, the virtual file system approach is enabled for multiple users to multiple local servers in the new FTP system (see Figure 3-13).

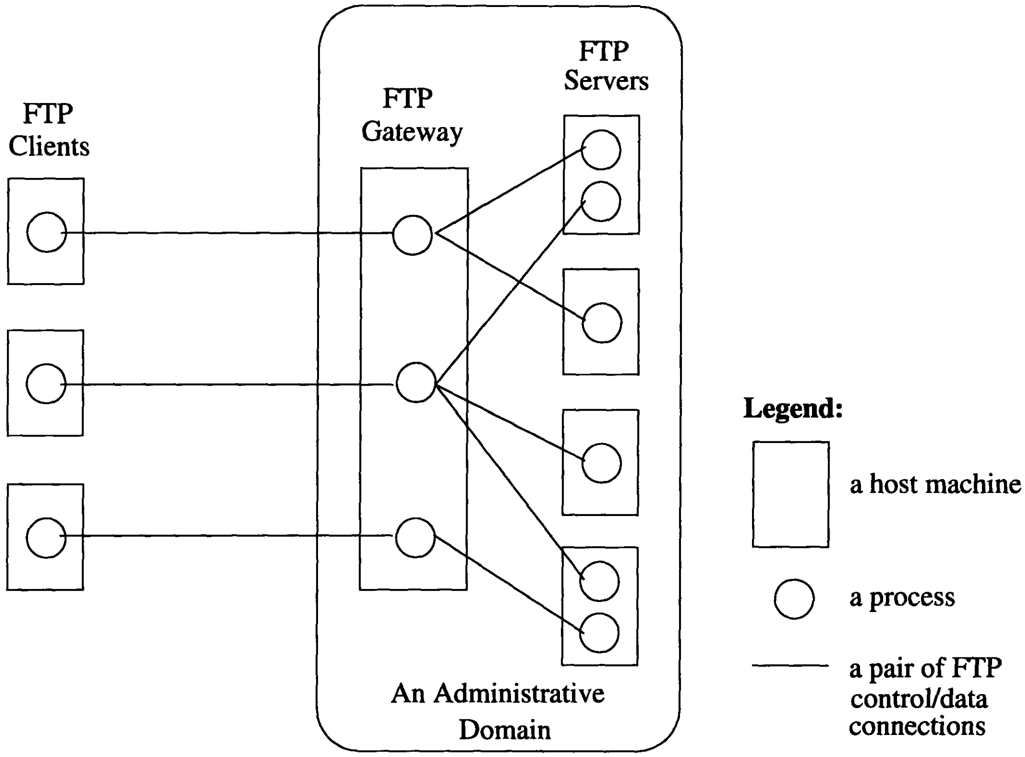

Figure 3-13: Multiple Users Accessing Multiple Servers in the New FTP Model

Chapter 4

Evaluation

In evaluating the implementation of a new security system, two different aspects must be considered: functionality and assurance [4]. First, the functionalities provided by the new security system must be justified by comparing them with those provided by the existing systems. The implementation is then tested to make sure that it performs the security functions correctly according to the design. The speed of service provided by the new security system is also evaluated against existing systems.

4.1 Functionality

4.1.1 Advantages

The new secure FTP model combines several advantageous ideas existing in different systems:

- The gateway in the new system is an application gateway, and therefore has the ability to hide the host addresses within an administrative domain from the outside world while allowing FTP service to the domain hosts.

- The gateway incorporates the simple address-filtering functionality used by most packet-filtering routers to eliminate unwanted access attempts.

- With a central gateway to the local domain, the new system allows uniform management of administrative policies to be imposed on FTP service to the domain community following the MASS (Modular Application Service Substrate [13]) approach.

- The new system expands on the fine-grain set of access control rights put forward by the AFS (Andrew File System [18]) to provide a finer-grain access control on the FTP service on top of the security functions provided by the domain hosts.

One of the reasons that the new gateway works better than the existing gateway designs is that it provides a simple language for writing descriptive security rules. The language has a fine-grain set of access control rights and the capability to specify groups in the subject and object of each security rule. These features facilitate the configuration and maintenance of the specifications of security policies for an administrative domain, thus increasing the likelihood of configuring the policies correctly and completely. The example given in Appendix A shows how the administrative policies can be written in a straightforward manner with this new language.

Another innovative feature that makes the new FTP system work better than other FTP systems is the incorporation of the virtual file system approach on the FTP service to an administrative domain of hosts. This special functionality allows users to request FTP service to different hosts in the same domain. Combining the access control right to go up (u) with the virtual file system feature, the new system provides the administrators with the ability to specify a new dimension of access control.

4.1.2 Limitations

With all its desirable functionalities, the new secure FTP system does have the drawback of not providing all the FTP commands prescribed in the standard FTP specifications [16]. As discussed in Section 2.6.7, the PASV and SITE commands are not included in the new system. Since the domain security gateway hides the addresses of domain hosts, the new FTP model prevents the usage of the PASV command to an outside third-party server. However, suggestions for future extension to the new FTP system are discussed in Chapter 5 to allow an FTP client to request data transfer between two server hosts inside the administrative domain. Moreover, for simplicity in implementing the new design prototype, several additional advanced features have not been included. These extensions are discussed in Chapter 5.

4.2 Assurance

For a new security system, testing the correctness of the implementation functions is particularly important. There are two different sets of functionalities provided by the implementation of the new FTP model—security functionality and virtual file system functionality.

Sets of both correct and erroneous FTP command inputs were used to test the robustness of the key parts of the new FTP program. Moreover, the new FTP program was kept running on a local machine for a one-month testing period. A large number of different FTP service requests were made to other neighboring hosts in the local domain through the new FTP program to test and verify the correctness of many different parts of the implemented functionalities.

4.2.1 Security Functionality

The correctness of the overall security functionality provided by the new implementation depends on the accuracy of two operations: the generation of access control data structures from the parsed information in the security configuration files, and the application of the stored access control information to restrict FTP services.

Correct Parsing of Configuration Files

A special debugger has been implemented so that the administrators can check the access control data structures generated for their security rules. The debugger not only lists the information stored in the internal data structures, but also identifies all occurrences of rule conflicts and indicates how they are resolved. With the debugger, the administrators can analyze their defined security rules and make sure desirable security policies are employed at the FTP gateway.

All the figures on the *securuldb* data structures shown throughout the thesis were generated by the debugger; they show the correctness of the parser in resolving rule conflicts according to the design descriptions. Moreover, an especially problematic sample set of security rules is used for testing the correctness of the security rules parser. The rules were run through the debugger and all bugs were successfully caught.

Correct Application of the Access Control Information

Not only can the debugger list the *securuldb* generated from the parsed information in configuration files, but it can also take a subject—a user from a source address—as its argument to generate the corresponding subject capability list, *subjcapls*. The administrators can quickly check and make sure what capabilities have been granted to a subject by their defined security rules. Again, all the figures on the *subjcapls* shown throughout the thesis were generated by the debugger; they show the correctness of the gateway FTP program in collecting subject capabilities from the access control information described in the configuration files.

The correct application of the new access control model on the FTP service was further tested with a large number of subject capabilities composed of different combinations of access control rights among "lriwdaum-". The implementation of the mapping of the access control rights to the FTP commands (see Figure 2-3) has been confirmed through the tests.

4.2.2 Virtual File System Functionality

The correct operation of the virtual file system functionality of the new FTP model largely depends on the correct implementation of the PWD, CWD and CDUP commands, as well as the implementation of the virtual root directory listing enabled through "LIST /", "NLST /" and "STAT /" requests. The implementation accuracy of the PWD command, the CDUP command and the virtual root directory listing was confirmed during the month-long testing period. At the same time, the CWD command was also substantially tested with simple path name arguments. As described in Section 3.2.6, the implementation of the CWD command is intended to be robust in taking its path name arguments; therefore, numerous obscure path names were used to test the robustness of the implementation of the CWD command and the results were positive.

The success of the virtual file system functionality also depends on: the appropriate implementation of the USER–PASS login sequence to change the stored password, the proper setup of connections between the gateway and the destination hosts in the domain, and the correct passing of control or data information between the internal (gateway–domain host) connection and the external (gateway–outside host) connection. All these functions were assured in the testing period.

4.3 Speed of Service

Although there is a classical tradeoff between security and speed of service in any system, in order for a new security system to be usable, it must not suffer too much in speed of service as a result of security enhancements. The fundamental service of the FTP is transferring data. Two different sets of tests were performed to compare the speed of data transfer provided by the new FTP system with that of the original FTP system. Another set of tests was run to measure the overhead delay caused by individual enhanced functionality. All the results were collected over nights on automated FTP services to minimize the influence of other competing processes.

A set of files was used for the tests: 1.txt, 10K.txt 100K.txt, 1M.txt, 10M.txt, 10K.exe 100K.exe, 1M.exe and 10M.exe. The ".txt" files contained plain text characters as well as newline characters; the ".exe" files contained binary codes. The ".txt" files were transferred as "ASCII-type" data in the FTP, and the ".exe" files were transferred as "image-type" data in the FTP [16]. Each file name was chosen to match the number of bytes contained in the file.

```
TYPE A N
RETR 10K.txt
RETR 100K.txt
RETR 1M.txt
RETR 10M.txt
STOR 10K.txt
STOR 100K.txt
STOR 1M.txt
STOR 10M.txt
TYPE I
RETR 10K.exe
RETR 100K.exe
RETR 1M.exe
RETR 10M.exe
STOR 10K.exe
STOR 100K.exe
STOR 1M.exe
STOR 10M.exe
```

Figure 4-1: Repeating Sequence of FTP Commands for Testing Data Transfer Rates

4.3.1 Data Transfer Rate for One Pair of FTP Client/Server Machines

First, a set of tests checked the data transfer rates from an FTP client at a machine ("18.58.0.25") to one FTP server at a machine (*sesame*) in a local domain of subnetwork "18.26.0". Repeatedly using the sequence of FTP commands shown in Figure 4-1, a control test collected the time used for each data transfer session in the original FTP model. The results were used as the control for comparisons.

Another test was run between the same pair of FTP client/server machines, but with the FTP requests processed by the new security FTP program running on *ginger* as the gateway. The same sequence of FTP commands was used with an additional "CWD *sesame*"command in the beginning to access the desired FTP server machine through the gateway. The time used for each data transfer session in the new FTP model was collected.

The new FTP model performs additional data copying from one data connection to another. The number of additional copying actions taken depends on the size of data in transfer, but at least one is needed per data transfer session.

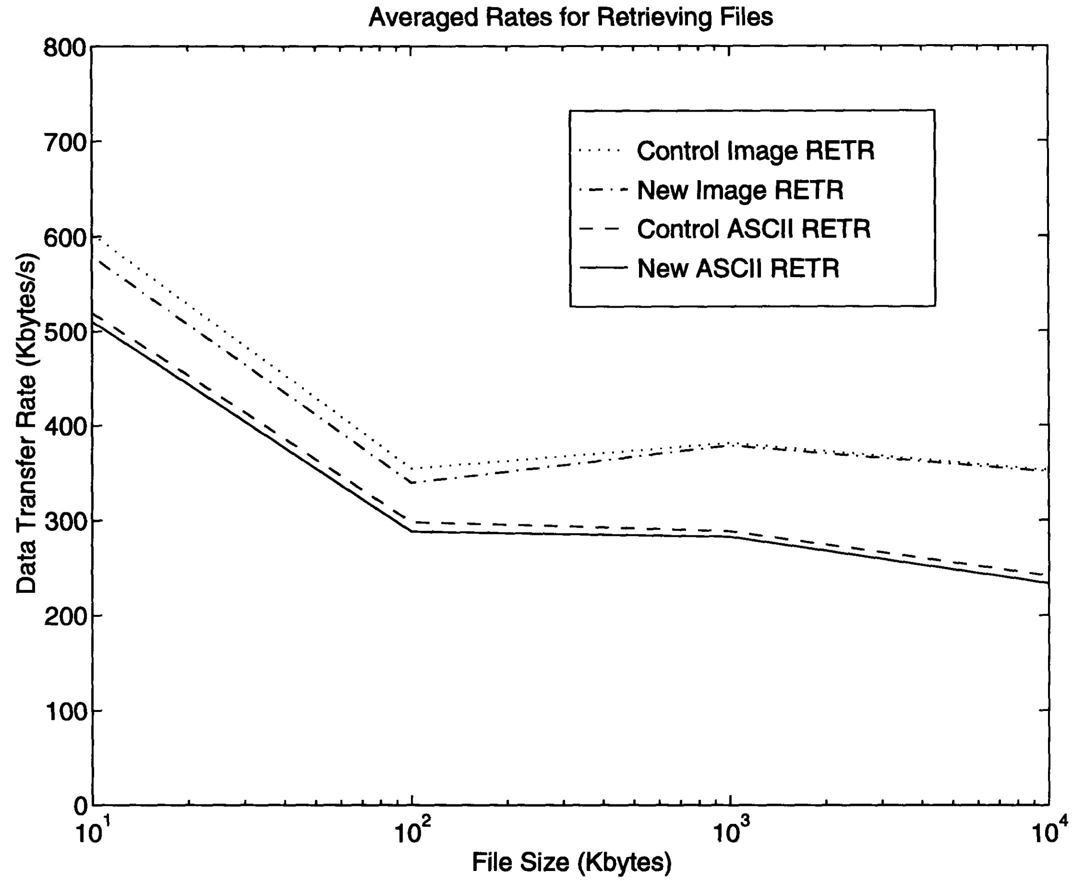

Figure 4-2: Averaged Rates for Retrieving Files

The comparisons of the two sets of results (see Figures 4-2 and 4-3 and Table 4.1) show that the averaged data transfer rates in the new FTP system equaled at least 90% of the averaged rates in the original FTP system.[1] These general results indicate that the copying of data from one connection to another by the new FTP program is very efficient. Moreover, the standard deviations of data transfer rates (shown in Table 4.2) were about the same in each case. This result implies that the gateway does not impose additional variations on the rates of transferring data.

[1] The high rates of transferring the 10K file is partly due to the inaccuracy in timing very short intervals, and partly due to the effect of buffer caching of the files at the receiving side. When a file is not large, the file data are transferred to memory, rather than disk.

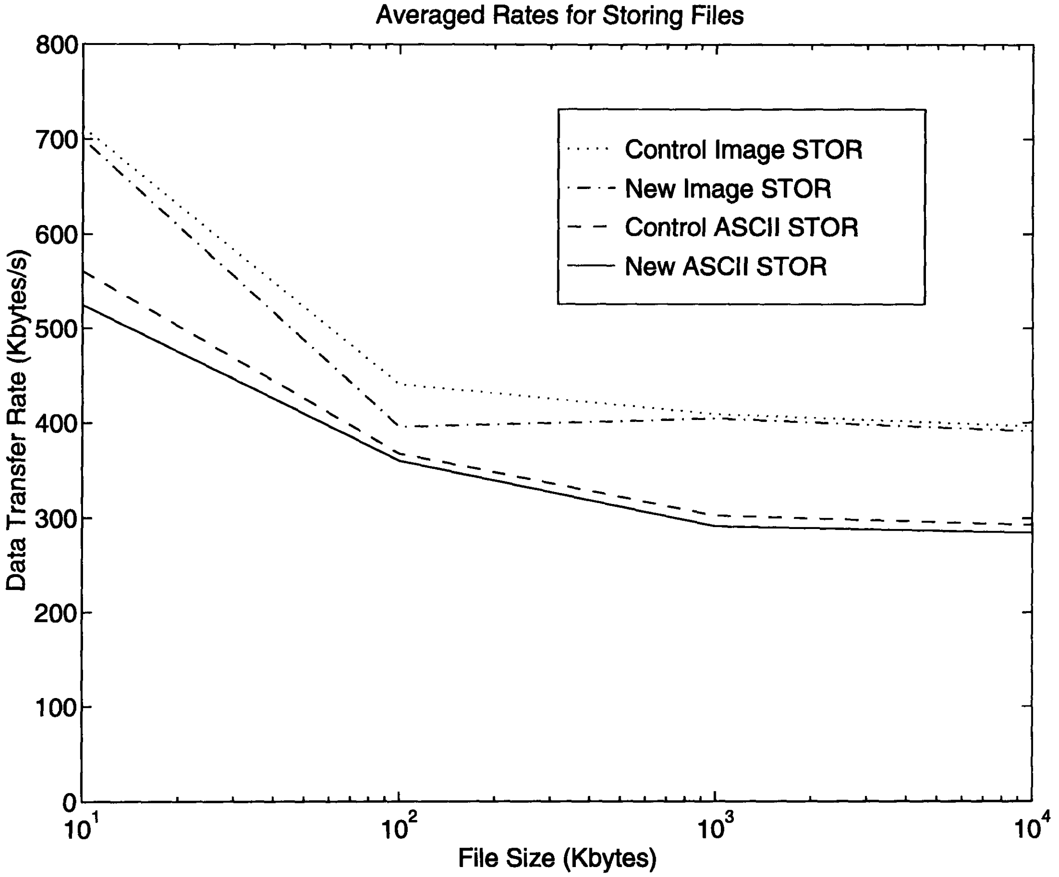

Figure 4-3: Averaged Rates for Storing Files

File	Control RETR (Kbytes/s)	New RETR (Kbytes/s)	Percentage	Control STOR (Kbytes/s)	New STOR (Kbytes/s)	Percentage
ASCII:						
10K.txt	519.5	519.5	100.0%	560.4	524.6	93.6%
100K.txt	298.2	288.0	96.6%	367.4	359.8	97.9%
1M.txt	288.3	282.4	98.0%	302.6	291.2	96.2%
10M.txt	241.2	233.4	96.8%	292.8	284.6	97.2%
Binary:						
10K.exe	603.1	579.2	96.0%	712.5	700.0	98.3%
100K.exe	354.6	339.4	95.7%	440.1	395.9	90.0%
1M.exe	381.3	378.6	99.3%	409.0	404.5	98.9%
10M.exe	352.8	351.1	99.5%	396.4	391.1	98.7%

Table 4.1: Data Transfer Rates for One Pair of FTP Client/Server Machines

File	Control RETR (Kbytes/s)	New RETR (Kbytes/s)	Percentage	Control STOR (Kbytes/s)	New STOR (Kbytes/s)	Percentage
ASCII:						
10K.txt	63.8	54.9	86.1 %	62.9	53.6	85.2%
100K.txt	71.9	76.7	106.7%	59.0	47.9	81.2%
1M.txt	53.5	50.0	93.5%	76.1	61.3	80.6%
10M.txt	35.5	41.0	115.5%	40.4	33.5	82.9%
Binary:						
10K.exe	64.2	60.3	93.9%	73.6	72.8	98.9%
100K.exe	72.8	83.4	114.5%	70.4	73.4	104.3%
1M.exe	89.8	94.0	104.7%	50.0	40.0	80.0%
10M.exe	30.1	35.4	117.6%	32.3	26.1	80.8%

Table 4.2: Standard Deviations of Data Transfer Rates for One Pair of FTP Client/Server Machines

Machine Pair	Control RETR (Kbytes/s)	New RETR (Kbytes/s)	Percentage	Control STOR (Kbytes/s)	New STOR (Kbytes/s)	Percentage
18.58.0.20 —adrastea						
100K.txt	200.1	182.1	91.0%	357.4	325.7	91.1%
100K.exe	220.1	194.6	88.4%	403.1	358.8	89.0%
18.58.0.25 —mintaka						
100K.txt	199.5	183.8	92.1%	253.3	216.4	85.4%
100K.exe	382.1	328.5	86.0%	499.5	433.7	86.8%
18.58.0.33 —sesame						
100K.txt	301.7	256.5	85.0%	287.5	271.1	94.3%
100K.exe	316.5	301.1	95.1%	362.6	335.7	92.6%

Table 4.3: Data Transfer Rates for Three Pairs of FTP Client/Server Machines

4.3.2 Data Transfer Rate for Multiple Pairs of FTP Client/Server Machines

Second, a set of tests checked the data transfer rates between three FTP clients from the subnetwork "18.58.0" and three FTP servers (at *adrastea*, *mintaka* and *sesame*) in the "18.26.0" subnetwork. The same repeating sequence of FTP commands was used in this set of tests (see Figure 4-4). A control test was performed using the original FTP model, with the three pairs of FTP client/server machines establishing their own control and data connections (see Figure 4-5). Each data transfer session was timed, and the results were used as the control for comparisons. Another test was run using the new FTP system, with the three clients connected to the gateway FTP program at *ginger* for FTP service to the three servers (see Figure 4-6). Again, each data transfer session was timed. The comparisons of this second set of results against the control set (see Table 4.3) show that the averaged data transfer rates in the new FTP system equaled at least 85% of the averaged rates in the original FTP system. These general results imply that the new FTP gateway program is successful in processing multiple FTP connections efficiently.

```
TYPE A N
RETR 100K.txt
STOR 100K.txt
TYPE I
RETR 100K.exe
STOR 100K.exe
```

Figure 4-4: Repeating Sequence of FTP Commands Used by Multiple Pairs of FTP Client/Server Machines

However, when earlier tests were conducted using a slower machine with less memory as the FTP gateway, the data transfer rates for multiple pairs of FTP client/server machines were lower. This degradation of performance is a result of the implmentation strategy of using one process for each FTP user (see Section 3.2.8). With this strategy, the performance for multiple simultaneous access is quite dependent on CPU and memory resources at the gateway. A different implementation strategy is to have one single process for the gateway program which uses the *select* UNIX system call to multiplex the I/O activities for different users. This implementation strategy would not have as significant a performance limitation as the one currently used.

4.3.3 Individual Functionality Overhead Delay

The tests discussed in the previous two sections show that the new FTP system is capable of providing competitive data-transferring service—the fundamental service of the FTP. Individual tests were conducted to measure the overhead delay caused by each enhanced functionality of the new FTP system.

Enhanced Security Mechanism

The new FTP gateway checks access control for many of the FTP service requests (see Figure 2-3). Two tests were run in order to measure the run-time overhead delay caused by the enhanced security mechanism. A repeating pair of FTP commands—“MKD *dir*” and “RMD *dir*”—was used in these two tests. The first test was conducted

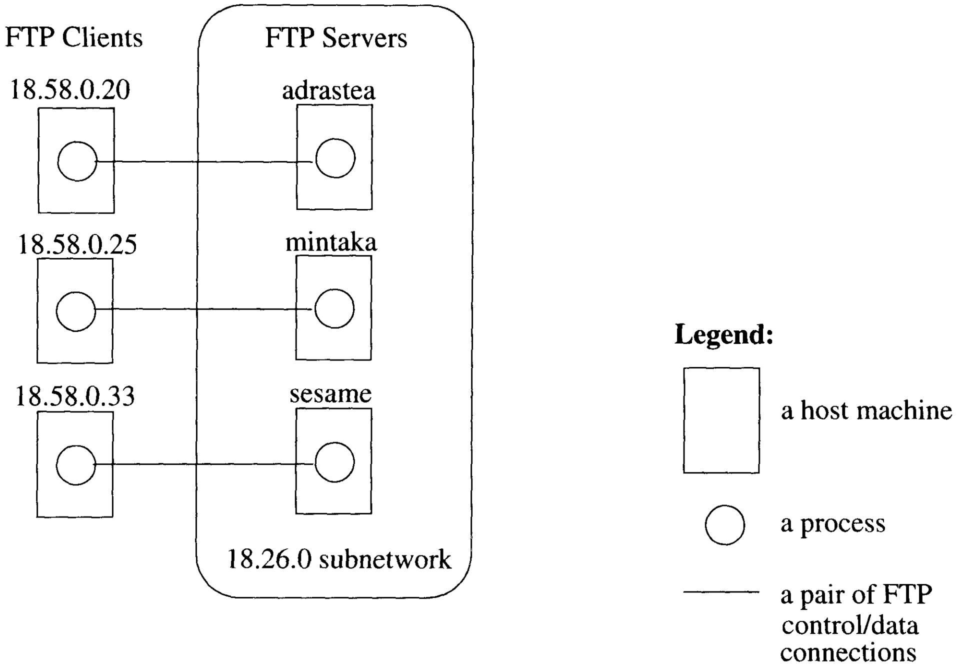

Figure 4-5: Three Pairs of Client/Server Machines in the Original FTP Model

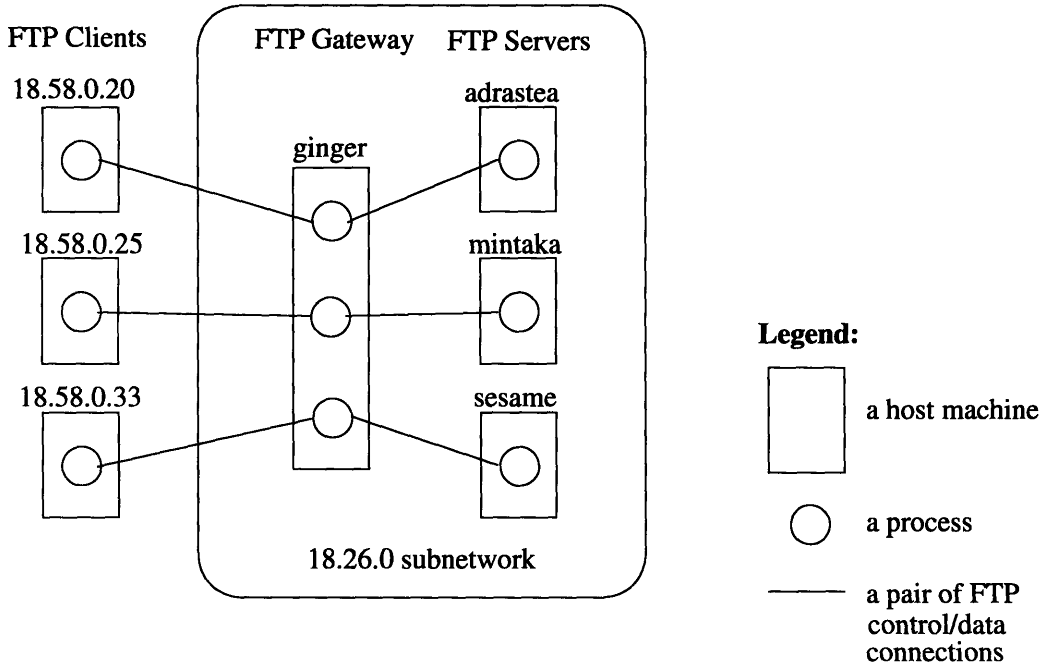

Figure 4-6: Three Pairs of Client/Server Machines Going through an FTP Gateway in the New FTP Model

using the original FTP model; the second test was run using the the new FTP system with the enhanced gateway checking the access control. Both tests involved running the pair of FTP commands 1500 times, and the overall time was recorded. The first test consistently yielded 105 seconds, and the second yielded 106 seconds. These results show that only a small amount of overhead delay (about 0.67 milliseconds) is added to each FTP service request with the addition of the enhanced access control checking.

A Separate Connection

The new FTP system protects the local domain servers by preventing direct FTP connections to these servers from the outside machines; as a result, additional data connections must be set up for each data transfer request (see Figure 4-6). Two tests were run in order to measure the overhead delay caused by establishing additional data connections. A repeating pair of FTP commands—"RETR 1.txt" and "STOR 1.txt"—was used in these two tests. The first test was performed using the original FTP model, and the second test was run using the the new FTP system, which established separate data connections to the local server. Both tests involved running the pair of FTP commands 500 times, and the overall time was recorded. The averaged time of first test was 107.2 seconds, and the averaged time of the second test was 108.7 seconds. These results show that an overhead delay of about 3.0 milliseconds is added to each data-transferring request. About 0.67 milliseconds of overhead delay is caused by the the enhanced access control checking, and the remaining overhead delay of 2.33 milliseconds is caused by the establishment of a separate data connection.

Virtual File System Mechanism

The new FTP system includes a virtual file system approach. As discussed in Sections 3.2.3 and 3.2.6, the CWD command was greatly modified to enable this enhancement. Two tests were run to compare the speed of service for the CWD request in the original and new FTP systems. The same repeating sequence of CWD commands was used in these two tests to traverse the same file system subtree (see Figures 4-7 and 4-8). The first test was performed using the original FTP model; the second test was run using the new FTP system, in which a virtual file system approach was included. Both tests involved running the sequence of FTP commands 27 times, and the overall time was recorded. The averaged time of first test was 15.9 seconds, and the averaged time of the second test was 41.9 seconds. As described in Section 3.2.6, the new FTP system splits the “CWD *../d*” into two CWD commands, “CWD ..” and “CWD *d*” for execution, thus the new system takes at least twice as much the time to execute “CWD *../d*” than the original FTP system. In addition, the new FTP system enables the virtual file system approach by checking carefully whether a change of destination host is necessary with each CWD request. Therefore, the processing of the CWD command is much longer when a virtual file system approach is used. Fortunately, the averaged 0.035 second of overhead delay for each CWD request is barely noticeable to an FTP user.

```
CWD a
CWD ../b
CWD ../c
CWD ../d
CWD ../e
CWD ../f
CWD ../g
CWD ../h
CWD ../i
CWD ../j
CWD ../k
CWD ../l
CWD ../m
CWD ../n
CWD ../o
CWD ../p
CWD ../q
CWD ../r
CWD ../s
CWD ../t
CWD ../u
CWD ../v
CWD ../w
CWD ../x
CWD ../y
CWD ../z
CWD ..
```

Figure 4-7: Sequence of FTP Commands for Measuring Processing of the CWD Command in the New FTP System

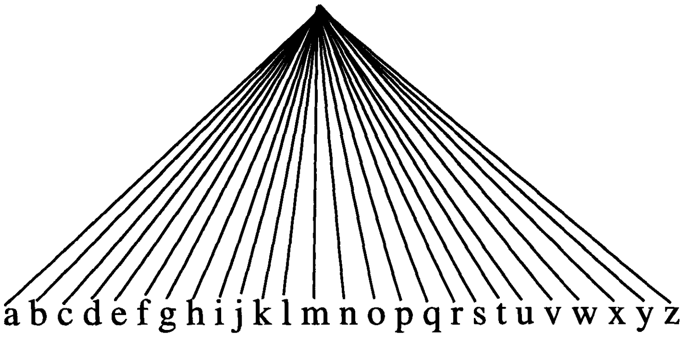

Figure 4-8: The Sample File Tree for Testing the CWD command

Chapter 5

Conclusions and Future Work

The prototype FTP gateway program has been evaluated to provide a better set of functionalities, to correctly implement the newly designed FTP model, and to provide service at as competitive a rate as in original FTP model. The prototype has been made as transparent as possible in terms of performance, user awareness and application awareness of the security measures. The prototype puts the MASS approach in use for a uniform, administratively directed access control of the domain hosts. The simple, yet descriptive language provided by the new design facilitate the domain administrators in writing the fine-grain security rules to be enforced on the FTP service to their domain. A new security system rarely reinforces security functionality without sacrificing convenience in usage; however, the new FTP model not only strengthens the security of the FTP services, but also adds convenience in the usage through a virtual file system approach to different host machines in a local domain.

5.1 Future Work

In the near future, several features may be added to enhance the design of the new FTP gateway program. These features have not been included in the current design for simplicity in implementing a working prototype.

5.1.1 Protecting Individual Directories

When there are domain hosts which do not offer protection for their files, it is desirable to have the security rule language extended in order to allow domain administrators to specify a subject's access rights to individual host files or directories. In the current design, the object of a security rule is a destination host or a group of destination hosts. Including a specification of file in the object of a security rule would provide a finer-grained access control.

In the enhanced design, the object of a security rule is a pair: a destination host field with a file field. The file field can specify either a plain file or a directory. A few extended security rules are shown to illustrate the new capability of the security rule language:

Rule 1x:	Y	123.7.0.0	16	hz	∼/proprietary/	i
Rule 2x:	Y	123.7.0.0	16	hz	∼/.login	lr
Rule 3x:	Y	123.7.0.0	16	hz	∼/	lriwd
Rule 4x:	Y	123.7.0.0	16	hz	/confidential/	-
Rule 5x:	Y	123.7.0.0	16	hz	/	lr

The subject of each of the three rules is user *Y* from source hosts in the "123.7" subnetwork, and the destination host is *hz*. The symbol ∼ is shorthand for a user's home directory. In general, the access control of a plain file is inherited from the access control of the directory at which it is located. However, if there is a specified security rule for the file, the access control rights specified in the security rule is used for the file. Similarly, the access control of a directory is inherited from the access control of its parent directory unless there is a specified security rule for the directory.

As a result of the five rules working together, Rule 1x allows the subject to only insert (i) new files into the *proprietary* subdirectory under user *Y*'s home directory; Rule 2x only allows the subject to list (l) and read (r) the *.login* file at user *Y*'s home directory; Rule 3x allows the subject to access all files and subdirectories under user *Y*'s home directory (except the *.login* file and the files under the *proprietary* subdirectory) with the access rights of "lriwd"; Rule 4x restricts the subject from accessing the *confidential* file subtree under the root directory; and Rule 5x allows the subject to list (l) and read (r) files under the root directory (except those under the *confidential* subdirectory) at host *hz.*

The *securuldb* and the *subjcapls* are extended to maintain this extra field of information and to prioritize the security rules by taking into account the extra field. The program can then rely on the processing of the CWD and CDUP commands to control access to the host directories. The program prompts the destination host to identify the current directory at which a user is located after each successful CWD and CDUP request. With the *subjcapls* extended to contain directory-specific and file-specific capabilities for a subject, the program can find the closest matched capability for the subject to use at the current directory of the destination host. When the gateway processes a file-manipulating FTP command (see Figure 2-3), it further checks for specific access control for the file; if there is no specified security rule for the subject to access the file, the subject's capability on the current directory is used.

5.1.2 Writing Source Address Field in Host Name Format

As mentioned in Section 2.3.1, the source address field can also be written in the conventional host name format. For example, "18.26.0.188" can written as "sesame.lcs.mit.edu". Allowing the source address to be represented in the host name format is recommended for readability. Moreover, in the future mobile networks setting, where a host machine may not be assigned with a fixed integer value address, it will be desirable to write a security rule with the source address field expressed in the host name format, rather than in any kind of integer value address format. In parsing the source address fields written in the host name format, the gateway program must resolve the host names using the Domain Name System [22][23]. However, resolving the host name of a remote site can take a rather long time; therefore, it is recommended that the capability of expressing the source address field in the four-byte address format is retained when the extra capability is added.

In the host name format, a group of source addresses can be expressed using a similar method for grouping destination hosts. For example, a source host group "dec.dialup.mit.edu:alfredo.mit.edu,carbonara.mit.edu,bolognese.mit.edu,primavera.mit.edu" can be listed in the additional source host groups file *src.grp*, and the administrators can specify a security rule for this group of source hosts using the notation "#G:dec.dialup.mit.edu" in the source address field of the rule.

5.1.3 Converting Path Name Formats of Different File Systems

As described in Section 2.6.3, the current design uses the UNIX path name format, but non-UNIX file systems may have different path name format. In a virtual file system approach, it is desirable to maintain a single format for the path name. Path name information returned from a destination host using a different path name format can be first converted to a standard path name format at the gateway before being forwarded to the FTP user. Once connected to a destination host, the gateway can prompt the destination host with the SYST command in order to identify the type of file system used at the host. The modified gateway FTP program can then convert all the returned path name information from the host.

5.1.4 Transferring Files between Two Domain Hosts

The original FTP model allows an FTP user to transfer files between two FTP servers. This service is arranged by allowing the FTP user to send the PASV request to one FTP server. In response to the PASV command, the FTP server runs in the passive mode by listening to a data port for the transfer of data, and the passive FTP server must reply to the FTP user with the host and port address to which it is listening.

For security reasons, the PASV command is disallowed in the new FTP system. The new FTP gateway hides the host and port address of a domain host from outside networks. The new FTP gateway also protects the domain of hosts by preventing direct connections between the domain hosts and the outside machines.

However, allowing an FTP user to request data transfer service between two domain hosts can be achieved without compromising the enhanced security for the

FTP. An additional FTP command specific to the gateway is suggested for improving the functionality of the new system. The command is called "SITE COPY". This command takes in two path name arguments—the source file name and the intended target directory path name. The gateway program checks the *subjcapls* to make sure the subject have the right to read (r) the source file, as well as the right to insert (i) into the target directory or the right to overwrite (w) a file in the target directory. The target directory path name is an absolute addressing virtual path name which contains the name of the destination host. For example, according to the *subjcapls* shown in Figure 3-8, user *B* from "137.1.15.3" working at host *hc* can request "SITE COPY a.out /hd/u/bb" in order to copy a file from the current directory at host *hc* to the */u/bb* directory at host *hd*.

A data connection may be established directly between the two hosts to transfer the data without having to go through the gateway because the two hosts are within the same protected administrative domain (see Figure 5-1). When the suggested "SITE COPY" command is processed, the gateway uses the PASV command internally in order to set a local domain host to passively accept data transfer initiation from another local domain host.

The new FTP gateway design was kept simple for ease of implementation, but allows for many modifications. The enhancements discussed in this Chapter are immediately feasible. There may be more features desired for use in securing the FTP system as the computer industry becomes more aware of the importance of security in the networking environment.

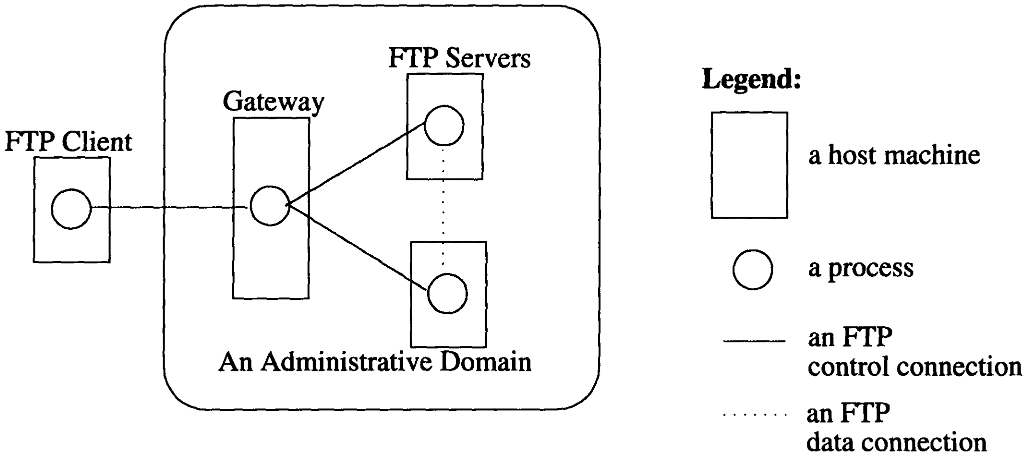

Figure 5-1: Transferring Files between Two Domain Hosts

Appendix A

A Practical Example Illustrating the Usage of the Security Rules and Group Files

In the following example, the usage of the security rules and group files are illustrated. There are three co-leaders (ld1, ld2 and ld3) five senior staff (ss1, ss2, ss3, ss4 and ss5) and eight junior staff (js1, js2, js3, js4, js5, js6, js7 and js8) in an administrative division. There are two working teams in the division. Leader ld2 leads the first team which consists of two senior staff (ss2 and ss3) and four junior staff (js1, js2, js3 and js4). Leader ld3 leads the second team which consists of three senior staff (ss3, ss4, ss5) and four junior staff (js5, js6, js7 and js8). Leader ld1 is the overseer of both teams and senior staff ss1 is the technical supporting staff for the division. With this organizational structure, a user file can be easily written (see Figure A-1).

The division have a domain gateway G and eight host machines (h1, h2, h3, h4, h5, h6, h7 and h8) open for accessing from the outside. Host machine h1 is the main server for the first team and host machine h2 is the main server for the second team. The first team uses the host machines h1, h3 and h5, and the second team uses the host machines h2, h4, h6 and h8. Host machine h7 is a back-up machine for all seven other machines. With this scheme of resources distribution, a destination host group file can be readily written (see Figure A-2).

```
leaders:ld1,ld2,ld3
seniors:ss1,ss2,ss3,ss4,ss5
juniors:js1,js2,js3,js4,js5,js6,js7,js8
team1:ld2,ss2,ss3,js1,js2,js3,js4
t1seniors:ss2,ss3
t1juniors:js1,js2,js3,js4
team2:ld3,ss3,ss4,ss5,js5,js6,js7,js8
t2seniors:ss3,ss4,ss5
t2juniors:js5,js6,js7,js8
```

Figure A-1: A Practical User Group File

```
servers:h1,h2,h7
team1:h1,h3,h5
team2:h2,h4,h6,h8
```

Figure A-2: A Practical Destination Group File

In addition, the organization is in the "147.15" network, and the division is in the "147.15.31" subnetwork. The co-leaders set the following administrative policies on accessing their domain host machines from the outside:

1. A member in a team can access the team's hosts from subnetwork "147.15" with "lr" rights.

2. A senior staff can access his team's hosts from subnetwork "147.15" with "lriw" rights.

3. A senior staff can access his team's server from subnetwork "147.15" with "lru" rights.

4. A leader can access his team's hosts from subnetwork "147.15" with "lriwdu" rights.

5. A leader can access any server from subnetwork "147.15" with "lriwdaum" rights.

6. A leader can access his team's hosts from his machine at home with "r" rights.

7. Leader ld1 can access any server from the sponsor's subnetwork "152.7.1" with "lri" rights.

8. Senior staff s1 can access any hosts from subnetwork "147.15" with "lau" rights.

Rule 1	#G:team1	147.15.0.0	16	#G:team1	lr	Policy #1
Rule 2	#G:team2	147.15.0.0	16	#G:team2	lr	:
Rule 3	t1seniors	147.15.0.0	16	#G:team1	lriw	Policy #2
Rule 4	t2seniors	147.15.0.0	16	#G:team2	lriw	:
Rule 5	t1seniors	147.15.0.0	16	#G:h1	lriw	Policy #3
Rule 6	t2seniors	147.15.0.0	16	#G:h2	lriw	:
Rule 7	ld2	147.15.0.0	16	#G:team1	lriwdu	Policy #4
Rule 8	ld3	147.15.0.0	16	#G:team2	lriwdu	:
Rule 9	leaders	147.15.0.0	16	#G:servers	lriwdaum	Policy #5
Rule 10	ld2	201.2.11.9		#G:team1	r	Policy #6
Rule 11	ld3	201.8.11.7		#G:team2	r	:
Rule 12	ld1	152.7.1.0	24	#G:servers	lri	Policy #7
Rule 13	ss1	147.15.0.0	16	#G:team1	lau	Policy #8
Rule 14	ss1	147.15.0.0	16	#G:team2	lau	:
Rule 15	ss1	147.15.0.0	16	#G:servers	lriwdaum	Policy #9

Figure A-3: A Practical Security Rules File

9. Senior staff s1 can access any server from subnetwork "147.15" with "lriwdaum" rights.

Then the security rules file is simple (see Figure A-3). These rules are not free of conflicts, but an additional simple strategy is used to resolve the rules after the general resolution principles of Section 2.3.3 have been applied. This additional strategy is to have a rule overrides its the similarly-matched preceding rules in the listing. For example, Rules 3 overrides Rule 1 for members of user group t1seniors acknowledging policy 2 overrides policy 1; Rule 5 overrides Rule 3 for members of user group t1seniors acknowledging policy 3 overrides policy 2. Without the capability to express groups, these fifteen rules would have to be expanded to ?? of rules. This whole example shows how the security rules can be practically used and how the grouping capability can be effective without inducing many confusions through an administrative approach to specify groups.

Bibliography

[1] Clifford Stoll. Stalking the Wiley Hacker. *ACM Communications*, Volume 31, Number 5, May 1988.

[2] Eugene H. Spafford. The Internet Worm Program: An Analysis. *Purdue University Department of Computer Science Technical Report*, CSD-TR-823, 1989.

[3] Peter G. Neumann. *"A Perspective from the RISKS forum", Computers Under Attack: Intruders, Worms, and Viruses.* ACM Press, New York, 1990.

[4] U.S. National Research Council. *Computers at Risk.* National Academy Press, Washington D.C., 1991.

[5] M.T. Rose. *The Open Book: A Practical Perspective on OSI.* Prentice Hall, Englewood Cliffs, NJ, 1990.

[6] Bruce Corbridge and Charles Slater Robert Henig. Packet Filtering in an IP Router. *Proceedings of the Fifth Large Installation Systems Administration Conference*, September 1991.

[7] Jon Postel. Internet Protocol DARPA Internet Program Protocol Specification. *Requests For Comments: 791*, September 1981.

[8] Jon Postel. Transmission Control Protocol DARPA Internet Program Protocol Specification. *Requests For Comments: 793*, September 1981.

[9] Jon Postel. User Datagram Protocol. *Requests For Comments: 768*, August 1980.

[10] S. M. Bellovin. Security Problems in the TCP/IP Protocol Suite. *Computer Communications Review*, Volume 9. Number 2; April 1989.

[11] Bill Cheswick. The Design of a Secure Internet Gateway. *Proceedings of the Summer 1990 USENIX Conference*, June 1990.

[12] Herve Schauer and Christopher Wolfhugel. An Internet Gatekeeper. *UNIX Security Symposium III Proceedings*, September 1992.

[13] David D. Clark, Karen R. Sollins, and John T. Wroclawski. Paradigms for Universality: Networking in the Information Age. Technical report, Massachusetts Institute of Technology Laboratory for Computer Science, July 1991.

[14] Smoot Carl-Mitchell and John S. Quarterman. Building Internet Firewalls. *UNIXWorld*, February 1992.

[15] Abhay Bhushan. A File Transfer Protocol. *Requests For Comments: 114*, April 1971.

[16] Jon Postel and J. Reynolds. File Transfer Protocol (FTP). *Requests For Comments: 959*, October 1985.

[17] D. Brent Chapman. Network (In) Security Through Packet Filtering. *UNIX Security Symposium III Proceedings*, September 1992.

[18] M. Satyanarayanan. Integrating Security in a Large Distributed System. *Carnegie Mellon University Computer Science Technical Report*, CMU-CS-87-179, November 1987.

[19] B. W. Kernighan and D. M. Ritchie. *The C Programinig Language.* Prentice Hall, Englewood Cliffs, NJ, 1988.

[20] J. Mogul and J. Postel. Internet Standard Subnetting Procedure. *Requests For Comments: 950*, August 1985.

[21] D. M. Ritchie and K. Thompson. The UNIX Time-Sharing System. *ACM Communications*, Volume 17, Number 7, July 1974.

[22] P. Mockapetris. Domain Names - Concepts and Facilities. *Requests For Comments: 1034*, November 1987.

[23] Finlayson and Mann. A Reverse Address Resolution Protocol. *Requests For Comments: 903*, June 1984.