Distributed Security Management Using LDAP Directories

Edgard Jamhour
PPGIA, PUCPR - Pontifícia Universidade Católica do Paraná
email: jamhour@ppgia.pucpr.br

Abstract

Presently, many companies share business information by interconnecting their networks through the Internet. However, this advanced degree of connectivity also increases the network security management complexity. Most of this complexity results form the need of controlling the connectivity of each network with respect to the others and the Internet. Also, it is necessary to take into account changes on users, shared resources and services, not only in the local network, but also in the interconnected networks. Because of these changes, network administrators are systematically confronted with firewall and other network elements reconfiguration. This paper proposes the use of a LDAP global directory service to simplify the task of managing the security in large-scale networks. By taking advantage of the distributed features of directory services, the paper defines a strategy for managing a group of interconnected networks as a single entity, without removing the administration autonomy of each independent network.

1. Introduction

The main issue of this paper is the access control management of very large networks, formed by a group of autonomous networks, managed independently, and interconnected by a shared network infrastructure, such as Internet. The Internet permits, at first sight, that any client accesses any server using any application protocol implemented over the TCP-UDP/IP protocol stack. However, by connecting a corporate network to the Internet one raises a problem: how to avoid that internal computer resources become unintentionally exposed to the other hosts connected to the Internet? The network strategy used to protect internal hosts consists in creating a selective connectivity between the corporate network and the Internet. Part of this selective connectivity is implemented using well-known firewall techniques; other part is implemented by the operating system of the hosts that require protection. To simplify the security administration task, one assumes that external hosts should be able to access only computers that offer standard services such as web pages and electronic mail. Therefore, the firewall function in this context is highly restrictive: it blocks all access attempts from external hosts but those related to standard Internet services.

By the other hand, standard services such as e-mail and web pages are just a small part of the services that can be shared through a global network. Strictly speaking, Internet is an IP network, being possible to implement over its infrastructure the same services available in a corporate network. In fact, the possibility of exchanging information between information systems of different companies is one of the foundation stones of modern techniques of e-business, process automation and endless perspectives of the "digital economy". Presently, it's perfectly possible for a group of independent companies to communicate in order to manage a full supply chain, acting as a single virtual organization (see Figure 1). In this case, even though the networks are physically independent and autonomous administered, clients and services from one network should be able to transparently access shared resources and services in the other network.

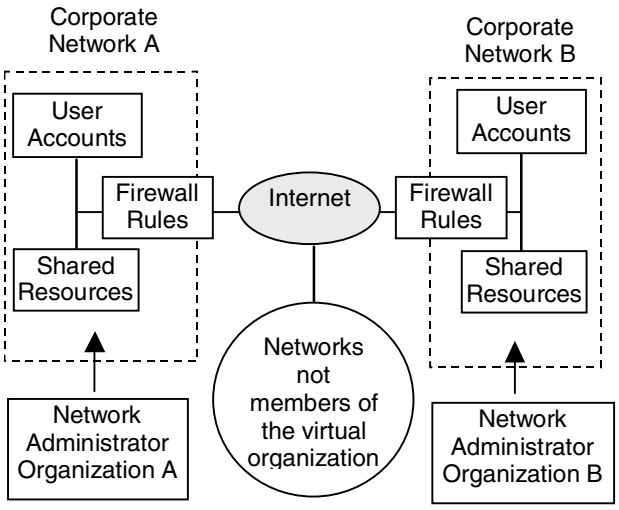

Figure 1. A virtual organization formed by independent corporate networks.

To permit the services exchanged between corporate networks to be increased, one must implement mechanisms to liberate the access to resources in the corporate network, selectively, to specific hosts and users located at external networks. Because each corporate

network is independently administered from the others, it is not a simple task. Independent administrators manage users accounts and control access to shared resources in an autonomous and decentralized way. It creates a difficulty for keeping a unique user account recognized in all networks. Other key point is to permit authenticated users to pass through the firewall barriers without exposing the corporate network to attacks from the external network. This last point is particularly difficult to implement, because the access control is implemented independently by the firewall and the operating system of the computer that hosts the shared resource. Also, it is necessary to take into account changes on users, shared resources and services, not only in the local network, but also in the other members of the virtual organization. Because of these changes, network administrators are systematically confronted with firewall and other secure related network elements reconfiguration.

1.1. Related Works

Many firewall products and research projects have addressed the problem of simplifying the task of firewall configuration. Current versions of Firewall products such as Check Point [10], for example, offers quite elaborated techniques for firewall configuration, including the capacity of managing several firewall devices from a single management server. In this case, the firewall rules are stored in a management server and transferred to one or more enforcement points (firewalls) distributed in the network. In this approach, the network administrator does not need to interact with each firewall in the enterprise in order to configure the security policy.

Another important current trend is the policy-based management of firewall devices. The policy-based approach differs from standard firewall configuration techniques that describe filtering rules based on IP addresses and TCP/UDP ports. Instead, policy-based rules describe classes of services that allow to group features that are naturally managed together [1,11]. For example, an email server requires receiving SMTP, POP3 and IMAP4 connections. By defining the service "email server", a network administrator can set all the firewall rules required for an email server to operate with a single command. Policy-based management does not store the security rules directly into the firewall. Instead, the rules are defined in an implementation neutral format and stored in a Database Server. When required, the neutral rules are compiled to a specific enforcement device. This approach is described in details in [1]. Another important work in this field is the Cisco Security Policy Manager [11]. In the Cisco's approach, the policy rules are extended to accommodate parameters such as time of day, detected attack, or network load. This approach permit not only to define the security policy of the enterprise, but also to define differentiated Quality of Service (QoS) for selected users and services in the network. In Cisco's approach, the policy rules are stored in a central policy database. A policy server is responsible for compiling and transferring the rules to the enforcement devices (see Figure 2).

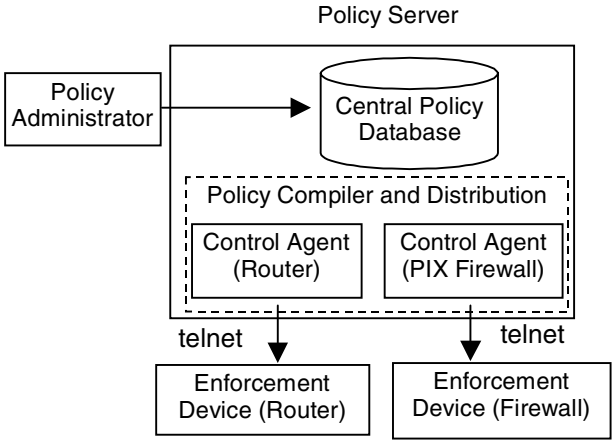

Figure 2. Cisco Secure Policy Manager.

Policy compilation requires accurate information about the network topology in order to translate neutral policy-rules to specific enforcement devices. For example, determine if a firewall interface is connected to the internal network or the Internet is fundamental for defining the packet filtering rules. It is also necessary to define the IP addresses of server and client hosts. In [1], an entity-relationship model is proposed for representing both the security policy and the network topology. In this approach, the network administrator describes the network topology and the policy rules using a "model definition language - MDL". A parser is responsible for interpreting the MDL and storing the described network elements and rules into a relational database. A model compiler is then used for converting the information in the database into configuration files specific for each firewall implementation. A different compiler is required for each type of firewall. That is basically the same approach described by Cisco [11]. Cisco's approach however, requires real-time updating of the enforcement rules because of the time and context-depended nature of its policy rules. In order to support real-time update of the enforcement rules, control agents are defined for automating the task of configuration creation and configuration deployment (see Figure 2). The control agents are notified when the policy rule list have been modified by the network administrator. The control agent then reads, compiles and transfers the policy rules to the enforcement device using telnet, as a human administrator would do.

From the previous examples, we determine that a security management system must define methods for:

a) Describing network topology and policy security rules for enforcement devices.
b) Storing policy rules and network topology information.
c) Compiling implementation neutral policy security rules into specific enforcement devices rules.
d) Distributing specific policy security rules for one or more enforcement devices.

An important issue related to the design security management systems is the format used for representing and storing policy rules. Recently, much standardization effort has started in this area. An important publication is RFC 3060 [12], which defines the Policy Core Information Model (PCIM). PCIM defines a generic object-oriented model intended to describe policy rules in the security and QoS domain. PCIM in conjunction with DEN (Directory Enable Networks) [14] is one more step in the direction of defining strategies for describing network configuration that can be stored in LDAP directories. PCIM is a very recent work, and it is not found yet in current security management systems implementations. However, many vendors manifested the intention of adopting this standard in their future releases. Also, many vendors have also declared their intention of adopting LDAP directories for storing configuration information. LDAP directories are described in the section 2 in this paper.

1.2. Related Technologies

Before continuing, it is important to understand the relationship between firewall configuration and VPN (Virtual Private Networks). Hardware and software vendors have exhaustively explored the problem of interconnecting networks through the Internet. The problem is usually referred as "building Extranets", i.e., setting up VPN links over the Internet. VPN are basically encrypted tunnels, implemented using protocols such as IPsec, which can be established between routers or computers. Building VPN, however, does not eliminate the problem of firewall configuration. In fact, VPN aims to permit authenticated users to safely access resources on a foreign network. However, for most applications, the foreign users can no gain the same access rights than home users. Network Administrators cannot rely only on the operational system authorization capabilities in order to protect the network. Once unencrypted, incoming packets from VPN links must also be submitted to firewall rules in order to determine if they can follow or not their path inside the network. The approach of filtering packets from VPN links is usually referred as VPN firewalls [13].

Other important issue is the use of private addresses. Most networks use private address for implementing IP networks. Two networks with private address can communicate with each other through the Internet by using VPN solutions. In this case, network administrators must care to not overlap private addresses in their networks. NAT can be employed in this case to translate private addresses of packets arriving from foreign networks to avoid overlapping. Observe that, in this case, firewall rules must be applied over private addresses. From the firewall viewpoint, however, there is no difference between packets with public or private addresses, once overlapping addresses are avoided.

The work presented in this paper does not explicitly consider private IP addresses. Our assumption is that all IP addresses are public or non-overlapping private addresses are employed through VPN Firewall links.

1.3. Our Approach

This paper proposes the use of a LDAP global directory service to simplify the task of managing security in virtual organizations. This work uses the same concepts explored in [1,10,11], i.e., policy rules are stored in an implementation neutral format and them compiled for specific devices. In our proposal, however, the policy rules and network topology information is not centralized, but stored in a distributed set of LDAP servers. By taking advantage of the distributed features of directory services, the paper defines a strategy for managing a virtual organization from a central viewpoint, without removing the administration autonomy of each independent member. The solution described in this paper is not intended to be a general framework for firewall configuration. Instead, only the information that needs to be shared among the member of the virtual organization is managed by our system. Each network administrator independently manages a LDAP server, publishing the information about shared resources to the other members of the virtual organization.

The remaining of this paper is organized as follows. Section 2 presents a short review of the distribution capabilities of LDAP directories. Section 3 present a directory schema for representing virtual organizations. Section 4 and 5 presents the implementation framework. Section 6 illustrates the approach with an example.

2. LDAP Directories

A directory service is a repository that stores information used by other applications and services in a distributed system. Two parts basically form a general-purpose directory service: a database and an access protocol. A directory database is a specialized version of a DBMS (*Database Management System*). It differs form a

general purpose database because it is specially designed to optimize read operations, but its performance for write operations is poor. A second very important feature of database directories is its scalability. A directory database can be implemented by a single server or transparently distributed by several servers. This feature is achieved by adopting an object oriented model and a hierarchical structure for storing information entries (database information tree). Figure 3 illustrates the hierarchical structure.

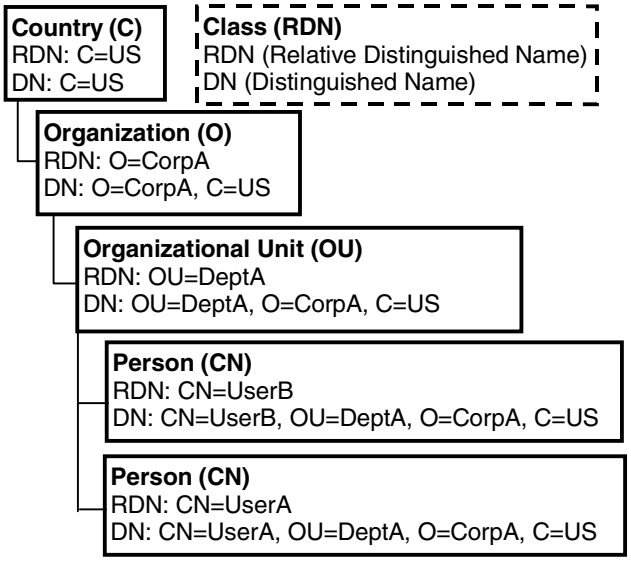

Figure 3. Example of hierarchical directory structure.

Most of the terms employed in Figure 3 are defined in the X500 standards [4]. X500 defined a completed directory service designed to operate over an ISO/OSI network. X500 defines that each element stored in a directory database is an object. A directory object belongs to an object class that defines a set of attributes. For example, "Person" in Figure 3 is an object class. An object is an instance of the object class. The set of classes and attributes that defines how the information is stored in a directory is called "directory scheme". One of the attributes of the object is used to give the name of the object. For example, CN is used to give the name for objects of class Person. This attributed is called RDN (Relative Distinguished Name) by X500. A RDN does not uniquely define a directory entry. The unique identifier of a directory entry is called DN (distinguished name). A DN is defined by concatenating all RDN of the objects defined in a directory hierarchy, as shown in Figure 3.

In practice, X500 became too complex to be implemented for commercial purposes. In the beginning of the 90's, the Michigan University developed a lighter version of X500 designed specifically to work with a TCP/IP network. This lighter version was called LDAP (*Lightweight Directory Access Protocol*) [5,6]. LDAP promptly became an Internet open standard, being published in a series of RFC (Request For Comments) standards. LDAPv1 was published by RFC 1487 in July 1993, LDAPv2, published by RFC 1777 in Mars 1995 and LDAPv3 published by RFC 2251 in December 1997. Many commercial products already supported LDAPv3 in 1999.

The initial idea of LDAP specification was to simplify the communication protocol between a client and a X500 directory server. LDAP assumed that the directory service would still be implemented under X500 standards. A LDAP server would work as a gateway that translated LDAP protocol into X500 standards. However, commercial implementations decided to simplify this approach by developing stand-alone LDAP implementations, where the LDAP server completely replaces the X500 directory, by also assuming the function Directory Database (see Figure 4).

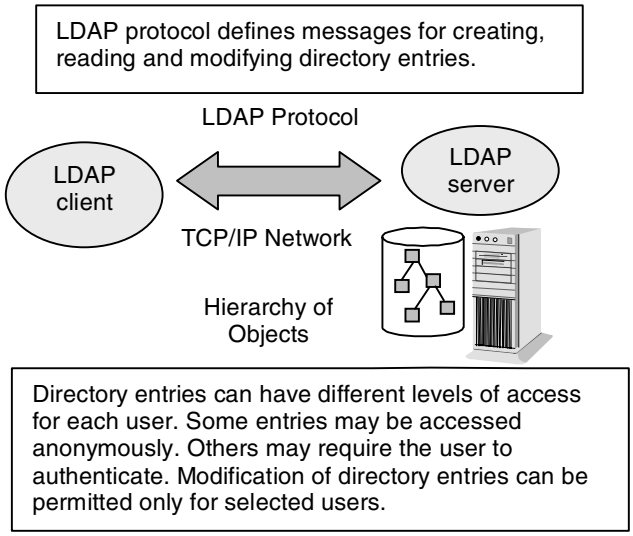

Figure 4. Stand-alone LDAP.

The mechanism employed by LDAP to distribute the information tree among several servers is called "referral". This mechanism defines a redirecting to another server when a client requests a directory entry that does not exist in the local server. The redirecting is implementing by comparing the DN of the requested entry with respect to the DN of the local directory. Figure 5 shows this approach.

The ldap.abc.com server stores the directory root. One of its entries was redirected to another server using a special directory entry called "smart referral". The smart referral implies that all entries bellow "Organization B" will be redirected to the ldap.xyz.com LDAP server. For

example, when the client request the directory entry: ldap://ldap.abc.com/ S=Email Server, pd=Organization B, o=virtual.com, it receives the answer: ldap://ldap.xyz.com/pd=Organization B. Then, it must reapply its request to another server exchanging the domain as indicated: ldap://ldap.xyz.com/ S=Email Server, pd=Organization B.

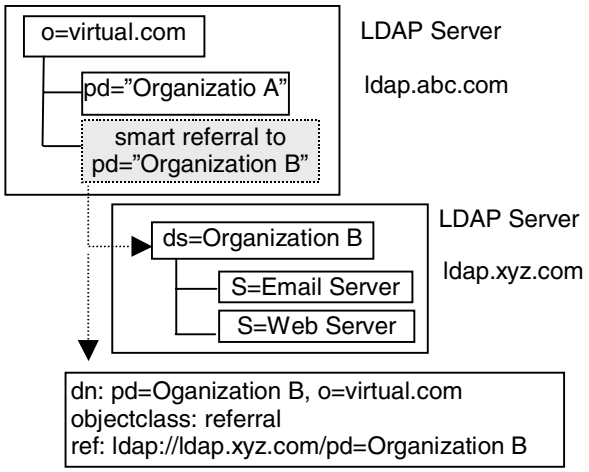

Figure 5. Smart-Referral.

3. Virtual Organization Modeling

This section shows how a LDAP directory can be used to represent a virtual organization. Figure 6 presents the proposed scheme of classes used to represent the security aspects of a virtual organization. The scheme has been significantly simplified for didactic purposes. The classes are described in Table 1.

The main idea behind the proposed scheme in Figure 6 is that a virtual organization is a collection of "protected domains". A protected domain is a group of hosts secured by the same enforcer device, such as a firewall. Because the computers inside an organization can have different level of protections, as in DMZ approaches, a protected domain can contain other protected domains. This idea is illustrated in Figure 7.

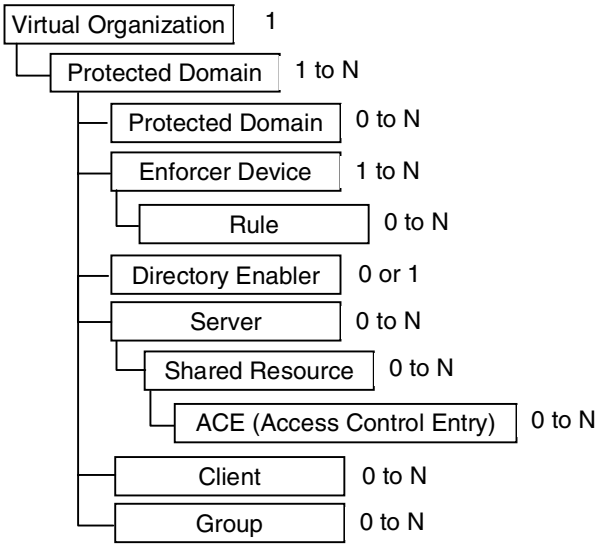

Figure 6. Directory scheme for describing the virtual organization security policy.

Table 1. Virtual Organization Object Classes

Class	Description
Virtual Organization RDN=O	Represents a collection of corporate networks that cooperates by sharing resources.
Protected Domain RDN=PD	Represents a collection of one or more hosts protected by the same enforcer device.
Enforcer Device RDN=ED	Typically, a firewall implemented in a host with multiple interfaces or a router.
Security Rule RDN=SR	Typically, a firewall rule such as: "Permit TCP inbound packets with ACK=1"
Directory Enabler RDN=DE	Computer responsible for offering directory enabled capabilities for other devices in the network (see section 4).
Server RDN=S	A computer that hosts a shared resource.
Shared Resource RDN=SR	Any sharable resource such as files, printers and services (e.g. email, web servers and databases).
Access Control Entry RDN=ACE	An explicit permission for a client or a group of clients to access a shared resource (e.g. "group A", "read, write, execute").
Client RDN=C	A user or service that can access a shared resource in the protected domain.
Group RDN=G	A group of clients with similar access rights.

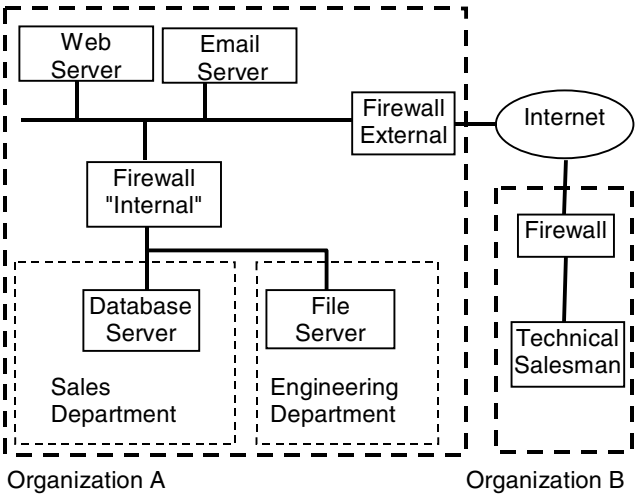

Figure 7. Example of virtual organization.

Figure 8 shows the object representation of the network illustrated by Figure 7. The model includes the representation of shared resources in the Database and File Server, located in DN= "PD=Internal", PD="Organization A", O="Virtual.com". Each of these shared resources can be associated to one or more Access Control Entries, indicating which clients have permissions to access these resources.

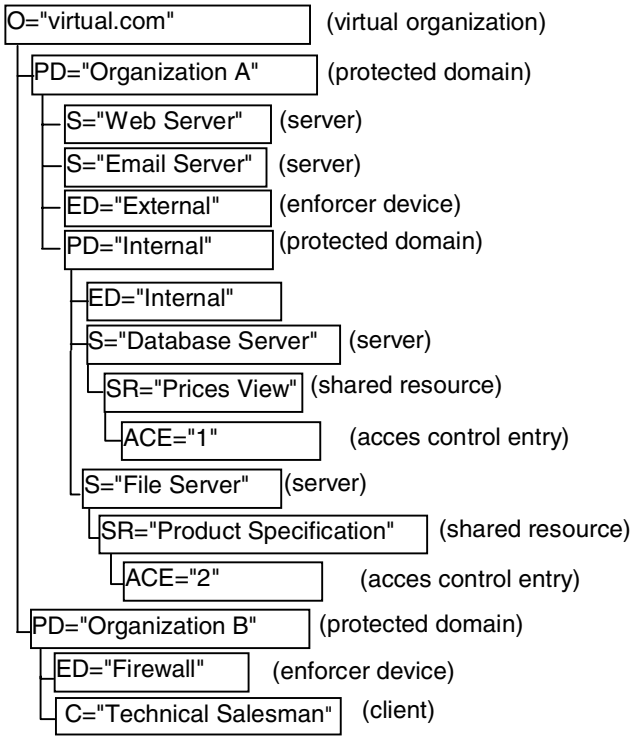

Figure 8. Object representation of the virtual organization.

4. Implementation

In order to simply the implementation in heterogeneous networks, this paper uses a synchronization approach for enforcing the security policy defined in the directory into the network elements. In this work, the device responsible for synchronizing the information in the directory with the actual network elements is called "directory enabler" and it is shown in Figure 9. The Directory Enabler has basically the same function than the Policy Server in Cisco's approach [11].

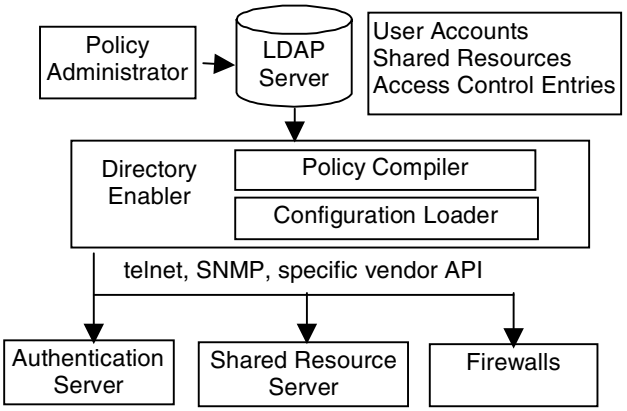

Figure 9. Enforcement with Directory Enable Synchronization Device.

The implementation of the directory enabler assumes two requisites. First, the directory enabler must "know" how to configure each network element under its control. Second, the directory enabler must have administrator rights for each network element under its control. The interaction between the directory enabler and the network elements can be defined in term of three fundamental operations:

I) Add, modify or remove a shared resource hosted by a server.
II) Add, modify or remove an access control entry (ACE) of a shared resource (see Table 1).
III) Add, modify or remove a packet-filtering rule.

In the absence of generic configuration protocols, it will be necessary to implement different versions of these operations according to the element to be configured. The policy compiler in Figure 9 is responsible for this operation. Second, it is necessary to send these configuration commands to the network elements. The configuration loader performs this operation. Network devices, as packet-filtering routers, are usually configured by SNMP (Simple Network Management Protocol) or by standard commands send through telnet Operating systems offers several resources for remote

administration, including telnet and specific API that simplify the task of writing configuration scripts.

To accommodate a heterogeneous network, the directory enabler can be implemented as a library of policy compilers and configuration loaders. By consulting the LDAP directory, the directory enabler gets the information about the element to be configured, i.e. operating system or model of network device and also the configuration method available. After that it needs just to select the proper policy compiler and configuration loader.

The administration rights offered to the directory enabler is a critical point for the overall network security of the virtual organization. In fact, it is totally undesirable that a unique host could store all the administration passwords of all network elements of the virtual organization. A more logical approach consists in defining an independent directory enabler for each member of the virtual organization. In order to do that, one must define the scope of each directory enabler with respect to the directory scheme that represents the virtual organization (see Figure 6 in section 3). Observe in that scheme that a protected domain can or not be associate to a directory enabler. Say that, one defines the following scope rule:

SCOPE RULE: "A directory enabler is responsible for managing its own protected domain and all protected domains that are hierarchically below its domain and don't have their all directory enabler".

To illustrate the approach developed in this paper, consider the network illustrated by Figure 10. Two members form the virtual organization in this example: "Organization A" and "Organization B".

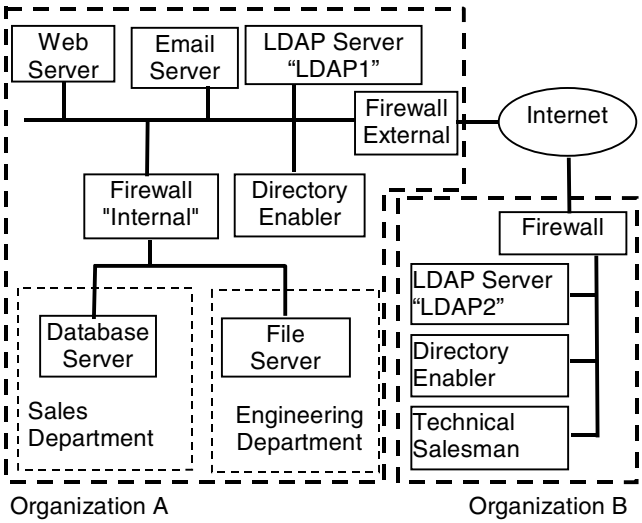

Figure 10: Enforcement with Directory Enable Synchronization Device.

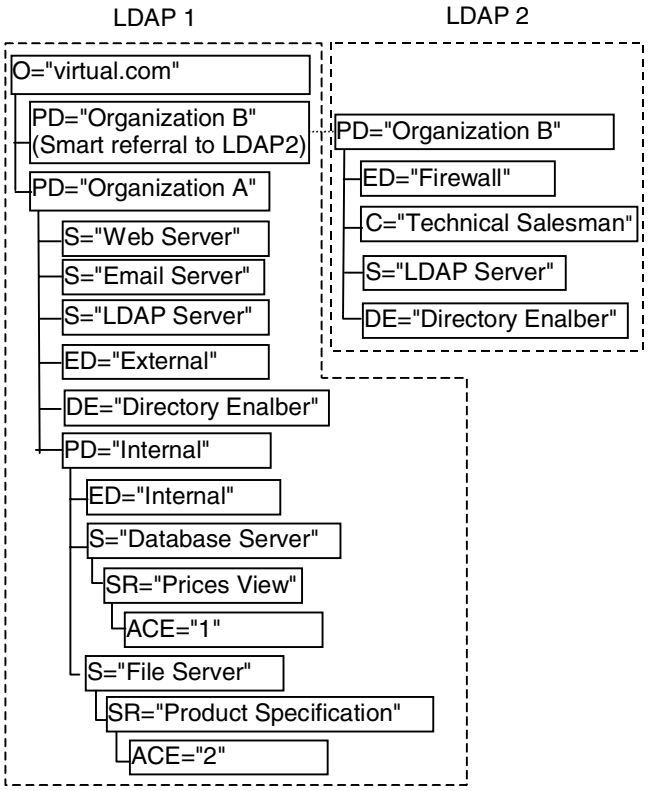

Figure 11: Object Representation of the Virtual Organization.

Each member has a directory enabler and a LDAP server. Both elements are protected by the firewall. Only the network administrator of each member has the administration rights for creating and modifying directory entries of its LDAP server. "Organization A" has two protected domains, but only one directory enabler. That means that the directory enabler of "Organization A" is responsible for managing both protected domains. Figure 11 illustrates the object representation of the virtual organization show in Figure 10. The LDAP server in the Organization A (LDAP 1) is the root of the virtual organization tree.

5. Authentication and Confidentiality issues

The proposed architecture leaves two questions without answer:

a) What happens with a Directory Enabler queries a malicious LDAP server, i.e., a LDAP server that's not part of the virtual organization?

b) How a local server can authenticate users that belong to other organizations?

The approach for solving both problems is based on SSL (Socket Security Layer). Originally developed by

Netscape, SSL has been universally accepted on the World Wide Web for authenticated and encrypted communication between clients and servers. SSL has not become a RFC. Instead, a new protocol suite called TLS (Transport Layer Security) has been adopted by IETF. Currently, TSL is in version 1 [9]. TLS 1.0 is based on the SSL 3.0 Protocol Specification as published by Netscape. Even though SSL 3.0 and TSL 1.0 are quite similar, they do not interoperate. The solution adopted in this paper is based on SSL 3.0 [8], although, TLS 1.0 could be adopted with slightly modifications. Many commercial LDAP servers, such as Netscape Directory Server used in our implementations, support SSL. The discussion next supposes that the reader is familiar with SSL 3.0 concepts as PKI – Public Key Infrastructure and digital certificates [8,9]. The SSL operation can be described by three phases:

PHASE 1: Client and Server Certification: The client starts a connection with a LDAP server in a secure port "sldap" (TCP port 636). Non-secure connections are established through the TCP port 389. The LDAP server sends an X.509.v3 certificate to the client. Optionally, the server can request a certificate from the client. The client validates the server's certificate by applying the Certification Authority's public key on the digital signature field in the certificate. The client, optionally, sends its certificate to the LDAP server, that validates the client's certificate. If the server's certificate or the client's certificate is not successfully validated, the SSL connection is aborted.

PHASE 2: Private Key Generation: The client generates a random 48 bytes secret, encrypts it with the public key received through the server's certificate, and sends it to the server. The server generates a private key by applying specific calculations on the secret received from the client. It sends a confirmation to the client informing it finished the operation. The client also generates a private key, applying the same operations on the secret, and sends a confirmation that it finished the operation to the server.

PHASE 3: Secure Communication: Client and Server exchanged encrypted information using the private key (symmetric encrypting algorithm). This private key is used only for one session.

In our approach SSL is used for bi-directional authentication between a directory enabler and a LDAP server. All directory enablers from a virtual organization must have a client's SSL certificate. All LDAP servers must have a server's certificate. All communication between a directory enabler and a LDAP server must be encrypted using SSL. Figure 12 illustrates this procedure. When a LDAP server receives a directory enabler's certificate, it maps the domain name stated in the certificate to a local account. This procedure permits to control witch LDAP entries a directory enabler can read. All directory enablers from a virtual organization are manually configured to read the LDAP server that represents the root of the virtual organization's information three. A directory enabler check the domain name stated in the certificate to be sure it's connected to the "right" LDAP server. Directory Enablers code can be implemented in Java or C, both languages have SSL API available.

SC: Server's Certificate; CC: Client's Certificate;
CA: Certification Authority

1. Directory Enablers are configured with client's certificates.
2. LDAP Servers are configured with server's certificates.
3. All communication between a Directory Enabler and a LDAP server must be encrypted.
4. Directory Enablers read user accounts entries from foreign LDAP servers, including standard login and password information, and create user accounts in home authentication servers.
5. Directory Enablers read ACE information from LDAP servers and configure home resource servers accordingly.

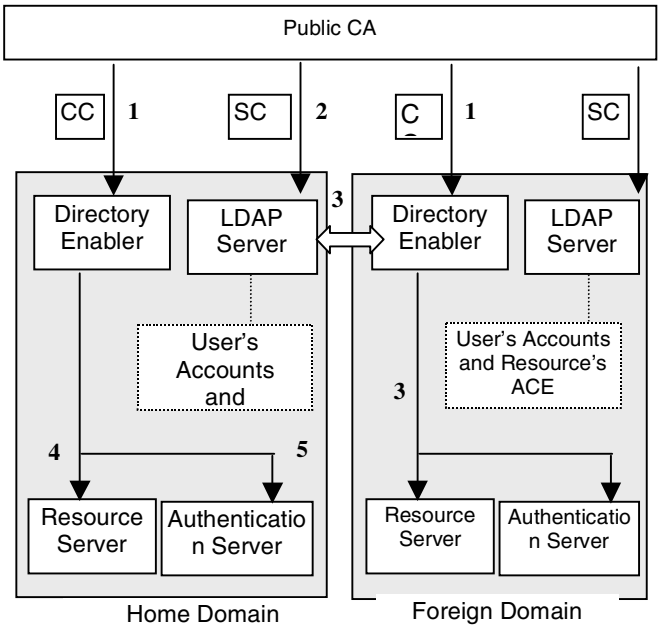

Figure 12: Interaction Between a Directory Enabler and a LDAP Server.

The directory enabler of each protected domain is responsible for synchronizing the information from the virtual organization LDAP tree into the local resource and authentication servers. This approach allows a single user account to be used across different protected domains and organizations. In fact, the account is created at all protected domain where the user from the virtual organization has access rights (defined by ACE entries).

A sort of convention must be establishing to users from the virtual organization in order to avoid login conflicts. Including the domain name from the home organization user in the user login's name can easily avoid this conflict.

6. Example

To illustrate the use of the directory information, this section will present the configuration procedure executed by the directory enabler. The procedure described in this section assumes the following security policy: "What is not explicitly permitted is denied." In order to simplify the access control configuration of each network, the network administrator needs to configure only the directory entries corresponding to the shared resources (i.e. ACEs). The directory enabler will automatically create the firewall rules in order to accommodate the ACEs found in the directory. When the network manager creates, modifies or removes an ACE, he affects the configuration of the firewalls at the client side and the server side. The firewall at the client side needs to be configured to liberate outbound packets with the client request and inbound packets with the server response. The firewall at the server side makes the opposite, i.e., it liberates inbound packets form client and outbound packets form server.

Each directory enabler consults the LDAP service to check for ACE related to resources in its all domain, but also checks ACE in other domains referring its users. This information is obtained with a single LDAP query. For the object model described in Figure 11 this query would be:

Base DN: o=virtual.com
scope: sub
filter: (&(objectclass=ACE) (modifytimestamp >=19991026155341Z))

The query described above follows the standard notation defined in [7]. LDAP queries are basically formed by three parameters. The Base DN defines the initial point of the search (the search is always implemented downward with respect to the directory tree). The scope parameter defines which objects in the tree should be tested in the query. The scope sub defines that all objects below the Base DN should be tested. The filter defines the search criteria. In this case, the filter indicates all ACE entries create after a certain time. Observe the use of the attribute "*modifytimestamp*" in the filter definition. This attribute is used to identify entries created since the last time the directory enabler checked the ACE entries. The value "19991026155341Z" is a *timestamp* created by the Netscape LDAP server [2], used to implement this work. For the object model defined in Figure 11, this query would return the following objects:

DN*:* ACE=1, SR=Prices View, S=Database Server, PD=Internal, PD=Organization A, o=virtual.com
Rule*:* permit(read)
DNclient*:* CN=Technical Salesman, PD=Organization B, o=virtual.com

DN: ACE=2, SR=Product Specification, S=File Server, PD=Internal, PD=Organization A, o=virtual.com
RULE: permit(read, list)
DNclient*:* CN=Technical Salesman, PD=Organization B, o=virtual.com

The next step is to program the rules for the packet filtering routers. To determine how to generate the packet filtering rules, the directory enabler compares the attribute DN of the shared resource with the DN of the client (i.e. DNClient). If none of these attributes have a reference to the protected domain under the control of the directory enabler, the ACE entry is discarded. On the contrary, it is necessary to determine witch modifications should be enforced in the network device to implement the ACE.

First consider the directory enabler in "Organization A". By analyzing the attributes DN and DNClient it is easy to conclude that both entries require liberating an internal shared resource for an external client. The required information about IP addresses, transport protocol and ports can be obtained from the LDAP server, by consulting the "S", "C" and "SR" entries. These attributes have not been detailed in this paper for didactic reasons. The results rules are listed bellow:

1. Accept Outbound TCP from "File Server IP, Port 139" to "Client IP, Port > 1023" and ACK flag 1.
2. Accept Inbound TCP from "Client IP, Port > 1023" to "File Server IP, Port 139" and ACK flag any.
3. Accept Outbound TCP from "Database Server IP, Port 8080" to "Client IP, Port > 1023" and ACK flag 1.
4. Accept Inbound TCP from "Client IP, Port > 1023" to "Database Server IP, Port 8080" and ACK flag any.
5. Reject anything.

Observe that the creation of an ACE entry implies that the filtering rules of all networks devices between the client and the server should be regenerated. The ACE's attribute DN shows that there are two protected domains between the shared resource and the Internet. It means that the above rules should be applied for the firewalls "External" and "Internal".

Consider now the directory enabler in the "Organization B". By comparing the attributes DN and

DNClient from the ACE entries it determines that the firewall should be reconfigured to permit a internal client to access external resources. The rules of the firewall are listed below:

1. Accept Inbound TCP from "File Server IP, Port 139" to "Client IP, Port > 1023" and ACK flag 1.
2. Accept Outbound TCP from "Client IP, Port > 1023" to "File Server IP, Port 139" and ACK flag any.
3. Accept Inbound TCP from "Database Server IP, Port 8080" to "Client IP, Port > 1023" and ACK flag 1.
4. Accept Outbound TCP from "Client IP, Port > 1023" to "Database Server IP, Port 8080" and ACK flag any.
5. Reject anything.

To deduce the firewall rules from ACE entries is not a hard task. However, one should consider that the rules generated from ACE entries wouldn't be the only rules for the firewalls. To avoid conflict with existent rules this work has assumed that the ACE defines only "permit rules" and what is not permitted is denied by default. This approach considerable simplified the algorithm for generating the rules. A problem raises when the network administrator deletes and ACE entry. In this case, the directory enabler should also remove the firewall permissions associate to the ACE. Because of this, ACE entries should be marked "deleted", and not delete immediately from the directory. The entry could be deleted only after all directory enablers in the network had updated their packet filtering rules. That poses some synchronization problems and requires additional directory entries to be implemented. Another approach to avoid inconsistencies is to regenerate all rules at each time. It is important to note, in this case, that only the rules generated by the Directory Enable are removed from the firewall. Other rules that are not related to the virtual organization management are not affected.

7. Conclusion

This paper describes a framework for managing very large networks created by connecting independently managed networks in a commercial relationship. The proposed framework is based on two open standards LDAP and SSL. LDAP directory are used to build a shared information tree about users accounts and shared resources in multiples networks. SSL is used for proving identity across members of the virtual organization, and for providing confidentiality for transmitting user accounts and passwords. The proposed approach is based on synchronization methods, i.e., the user accounts from foreign networks are created in home authentication servers. LDAP supplies ways for building the same account in different domains, achieving a transparent authentication for users in the virtual organization. This approach can be implemented in a quite scalable way, and not all elements in the network need to be managed by the virtual organization framework. Future implementations will include more generic policy based rules and redefine the LDAP scheme according to the general directives proposes by RFC 3060.

References

[1] Y. Bartal, A. Mayer, K. Nisim, A. Wool. "Firmato: A Novel Firewall Management Toolkit", *IEEE Symposium on Security and Privacy*, Oakland, California, May 1999, pp. 17-31.

[2] Netscape Communications Corporation, "An Internet Approach to Directories", 1997, http://developer.netscape.com/docs/manuals/ldap/index.html.

[3] B. Moskowitz, D. Karrenberg, G. J. de Groot, E. Lear, "Address Allocation for Private Internets", *Request For Comments 1918*, February 1996.

[4] "The Directory: Overview of Concepts, Models and Service", *CCITT Recommendation X.50*0, 1988.

[5] W. Yeong, T. Howes, S. Kille, "Lightweight Directory Access Protocol",. *Request For Comments 1777*, March 1995.

[6] T.A. Howes, M.C. Smith, "A Scalable, Deployable, Directory Service Framework for the Internet", *Internet Society, INET'95*, Honolulu, Hawai, 1995.

[7] T. Howes, "A String Representation of LDAP Search Filters", *Request For Comments 1558*, December 1993.

[8] A.O. Freier, P. Karlton, P.C. Kocher, "The SSL Protocol Version 3.0", *Internet Draft*, March 1996 (expired).

[9] T. Dierks, C. Allen, "The TSL Protocol Version 1.0", *Request For Comments 2246*, January 1999.

[10] Check Point Firewall-1, Technical Overview, October 2000. http://www.checkpoint.com/products/downloads/fw1-4_1tech.pdf

[11] S. Hinrichs, "Policy-Based Management: Bringing the Gap", *Proceedings of the 15th Annual Computer Security Applications Conference*, December, 1999 Phoenix, Arizona.

[12] B. Moore, E. Elleson, J. Strassner, A. Westerinen, "Policy Core Information Model", *Request for Comments RFC 3060*, February 2001.

[13] J.Halpern, S.Convery, R. Saville, "IPSec Virtual Private Networks in Depth VPN", *Cisco Systems White Paper*, June 2001. http://www.cisco.com/warp/public/cc/so/cuso/epso/sqfr/safev_wp.htm

[14] DMTF: Distribution Management Task Force, "Directory Enable Networks (DEN) Initiative", http://www.dmtf.org/standards/standard_den.php