SAND2008-1257C

Directory Encryption Protocol: A Method for Securing Sensitive Data in Directory Services

William R. Claycomb[1] and Dongwan Shin[2]

[1] Sandia National Laboratories, P.O. Box 5800, MS 0823, Albuquerque, New Mexico, 87185-0823, USA
{wrclayc@sandia.gov}

[2] Computer Science Department, New Mexico Institute of Mining and Technology, 801 Leroy Place, Socorro, New Mexico, 87801, USA
{doshin@nmt.edu}

Abstract. This paper presents a protocol for encrypting information stored in an LDAP directory service. The protocol introduces an additional component between the client and destination server, which handles encrypting and decrypting information as necessary. Limited configuration is required to existing LDAP clients, no configuration is necessary for existing LDAP servers. No additional software is required for either clients or directory servers. Data is protected from insider threats, particularly those from a user with administrative access to the directory. Two models are presented which balance response time with overall security.

Key Words. Security, LDAP, active directory, threats, insider, encryption

1 Introduction

Directory Services are commonly implemented by organizations to handle a variety of tasks. Many use them to store data, as well as distribute information internally. Other applications include sharing directory information, such as addresses and phone numbers, with external organizations. Sometimes this information contains sensitive data and needs to be protected. Methods exist for protecting directory information, but can be difficult to apply and enforce. Additionally, many of these methods are susceptible to a particular type of attack, called an *insider attack*.

An insider attack is performed by a user within the corporation. This user may or may not have administrative privileges, but generally this makes an attack more successful, as well as easier to conceal. Organizations with weak standards for monitoring and auditing administrative tasks will find it difficult to detect these types of attacks. One common form of insider attack involves an administrator modifying an access control list (ACL), viewing and/or copying the protected data, and then replacing the original ACL. Another possible attack involves the administrator "spoofing" a user account, either by compromising the user's password, or by administratively resetting the user's password to something known by the attacker.

As more organizations use directory services to meet various infrastructure needs, the amount of sensitive data contained within these directories will increase. Consequently, these directories will become more valuable targets for insider threats. Preventing administrative insider attacks against sensitive data has become an important topic for system administrators, particularly those involved in protecting directory services.

We present a novel protocol for securing data within directory services. This solution operates in conjunction with the Lightweight Directory Access Protocol (LDAP), the industry standard for directory communications. Our solution meets several key goals, including protecting against administrative attacks, without greatly impacting directory service performance, and eliminating the need for additional applications on either the client or server. We propose a new component which acts as a *proxy* between the client and server. Using encrypted information sent from a client application, this proxy handles all communication with the destination LDAP server. It also handles encrypting and decrypting information that needs to be protected. The proxy can act in either an *active* or *passive* mode, distinguished by the level of interaction required between the user and proxy to bootstrap directory encryption activity.

The remainder of this paper is organized as follows. Section II describes previous work on securing directory services, and outlines recent studies concerning insider threats. Section III presents our design goals and motivation for this proposal. Section IV provides a detailed explanation of the communication protocol, and Section V discusses the impact and advantages of the solution, as well as potential attack scenarios. Section VI concludes the paper with a discussion of future work planned.

2. Background and Related Work

2.1 Directory Services

Directories are collections of information related to objects in an organization. These objects often include users, computers, or contacts. *Directory Services* are the services which make this data available for use by others. Frequently, the intention is to provide a single point of access for various applications and individuals to find information about users and other objects within an organization [1]. The information contained within the directory may come from direct input, and can be manually maintained, but also may be referenced and managed indirectly from other corporate data repositories, such as databases and other information stores. Commonly used directory services are Microsoft Active Directory [2], IBM Tivoli [3], Apple Open Directory [4], Novell eDirectory (formerly called Novell Directory Services) [5], OpenLDAP [6], Fedora Directory Server [7], and Sun Java System Directory Server [8].

2.1.1 Protecting Information in Directory Services

A few techniques exist for actually protecting the information stored within a directory itself. In general, access control lists (ACLs) can be used to implement some form of protection in most directories. For instance, in OpenLDAP, ACL protection can be applied to individual objects, groups of objects, specific LDAP filters, or a list of attributes [9]. Other techniques are almost exclusively implementation-specific.

Microsoft Active Directory [2] provides additional access control features through the use of *confidential attributes* [10]. This is a setting applied to the *searchFlags* component of individual attributes, and is only supported on Microsoft Windows Server 2003 SP1 and later. When processing confidential attributes, the directory server checks for additional access control rights associated with the requesting user. This particular type of access, called "CONTROL_ACCESS," is granted to administrative accounts by default, but can be delegated to other accounts individually.

Another approach to protecting attributes is encrypting them. Fedora Directory Server [7] has the capability to encrypt all instances of specified attributes. This means that for every object containing such attributes, the data in that attribute is encrypted using a symmetric key known to the directory server (the server SSL key). Various encryption methods can be configured, and different attributes can be encrypted using different ciphers. Encryption and decryption are handled by the directory server itself, so access to attributes is not controlled by this method. However, data would be protected from unauthorized access if the actual directory information was stolen or otherwise compromised.

A third approach to protecting directory attributes is described in [11]. This method is not dependent on a particular directory implementation. Rather, it uses public key infrastructure (PKI) to allow users to control the encryption of attributes related to their own directory information. This solution describes different methods for using PKI to ensure either data authenticity alone, or data authenticity combined with confidentiality. Specific solutions are proposed for scalability and usability purposes.

Additionally, [12] proposes encrypting directory information for users based on a *unique-id* chosen for each user. This method applies primarily to public directory servers, and does not address the issue of preventing unauthorized access so much as it addresses the issue of preventing access to the entire directory. For instance, a company could share contact information publicly, and provide selected clients with appropriate unique-ids, without worrying that the entire directory would be scanned for email addresses. One important aspect of the work is to choose a unique-id well, so that it cannot be easily guessed, but can still be easily shared with authorized users.

A general method for encrypting directory attributes individually, and based on user-specific information, is presented in [13]. Two specific methods for encrypting data in directory services are outlined. In the first, clients send LDAP operations to a client runtime application, which handles encrypting and decrypting data based on keys stored within the directory. The second method is similar, except the additional component resides on the server instead of the client. In each case, however, additional software is required by either the client, or server, or both, to carry out the data protection. This solution provides the capability to either sign or encrypt the data to be stored, which does increase its potential application, depending on the needs of the user.

2.2 Insider Threats

The threat of unauthorized access of sensitive data by employees or other authorized users, known as “dedicated insiders”, is well documented [14, 15, 16]. While the psychology and behavioral factors of these individuals is beyond the scope of this paper, their motivation and level of access should be considered. Additionally, it should be noted that the number of offenses committed by insiders is rising each year [16].

In January 2008, the U.S. Secret Service and CERT issued a report titled “Insider Threat Study: Illicit Cyber Activity in the Government Sector” [14]. This study outlines a multi-year project, started in 2002, that explores the activity and threats posed by insiders, defined as “employees who have perpetrated acts of harm against an organization via computer, system, or network to include theft of intellectual property, fraud, and acts of sabotage within critical infrastructure sectors.” Among the key findings of the study that are relevant to this paper are the following:

- “Most of the insiders had authorized access at the time of their malicious activities”
- “Access control gaps facilitated most of the insider incidents, including:
    - The ability of an insider to use technical methods to override access controls without detection
    - System vulnerabilities that allowed technical insiders to use their specialized skills to override access controls without detection”

In addition to outlining the methods and characteristics of the unauthorized access, the study also details findings about the motivation of the insiders, as well as the scope of the problem. In particular, the study notes that “in many cases insiders used authorized access to alter or obtain an individual’s personal data in some manner.” Theft of personal data was noted as a particularly likely target of insider threats, most often to sell to others for financial gain. The study notes that this is useful in “understanding how access to identity-related data might contribute to insider activity in [the government sector].” Additionally, it was noted that “agencies at all levels of government were targets of insider threat,” and that the attacks were successful because of “similar vulnerabilities in business practices and access controls.” [14]

To address these concerns, the study also presented considerations for government agencies with respect to the protection of data, including the following:

- “Electronic storage of citizens’ confidential information necessitates accurate, reliable, and confidential record keeping within government databases and computer systems. Policies and technical controls are implemented to provide a safety net for critical data.”

- "Government agencies at all levels need to remain vigilant against the potential impacts of insider incidents on public trust and the citizens' confidence in government services"
- "Government agencies should have proactive strategies to protect information entrusted to them"
- "Federal agencies are required to comply with Title III of the E-Government Act of 2002 known as the Federal Information Security Management Act."

3. Motivation and Design Goals

3.1 Motivating Factors

The findings outlined in the previous section largely deal with systems that store sensitive personal information. Consequently, the security of directory services has never really been an important consideration for system administrators. After all, isn't the information in a directory *meant* to be shared? In more and more instances, however, information beyond simple names and telephone numbers is being stored in directory services. Some systems require sensitive user attributes, such as clearance level, to be maintained to facilitate access control decisions. Additionally, in the case of automated account provisioning utilities, a method for synchronizing data between the directory and the authoritative data source is needed. Often, this method cannot be based on user names, or any derivation thereof, due to the changeability of that data. Non-changeable data, such as employee IDs, or Social Security Numbers are often used, and must be stored with user account information in the directory. Frequently, this combination of unique identifiers with personal information is considered sensitive.

Additionally, it should be noted that access to directory services is relatively complicated to configure. Microsoft notes that "In the Active Directory directory service for Microsoft Windows Server 2000 and for Microsoft Windows Server 2003, it is difficult to prevent an authenticated user from reading an attribute" [10]. Also, specifying access to individual attributes on a per-user basis could become a very time-consuming task in large organizations, as the task of modifying ACLs of directory attributes would most likely fall to a system administrator, rather than the end user.

Our motivation is to prevent an insider attack from accessing sensitive personal information stored in directory services. Specifically, we will consider users with administrative access, and will address the following techniques for gaining unauthorized access to the data. We will refer to these attacks as *dedicated administrator attacks*:

- An administrator copies information from the underlying data storage and reads the duplicated data
- An administrator modifies the existing ACL to give himself access, then restores the original ACL after reading the data

- An administrator "spoofs" an authorized user, either by
    - Compromising the existing user password, or
    - Resetting the old user password to a new password known to the administrator

The common characteristic to note among these attacks is that the actions may remain largely undetected. More active attacks are certainly possible, such as compromising a user workstation or modifying user PKI information, but these activities are generally monitored and detected with more rigor than those described above. We seek to prevent unauthorized access without relying on detailed detection and access control methods that are sometimes overlooked anyway [14].

3.2 Design Goals

Our aim is to present a solution that is both secure, as well as usable. Security is the primary goal, but without usability, a secure solution is often impractical or impossible to implement. With that in mind, we present the following usability goals:

- The solution must adhere to existing LDAP standards and support standard LDAP operations
- The solution must not require additional software for either the client or server
- The solution must be interoperable with existing LDAP-compliant clients and LDAP directory servers
- The solution must allow users to share access to protected data with other authorized users.
- The solution must be easy to use, with no complicated configuration steps and simple user interaction.
- The solution must provide a way to manage both encrypted and unencrypted attributes with the same client configuration.

4.0 Directory Encryption Protocol

The protocol described in this section is used to encrypt and decrypt attributes in a directory service. The following variables and notations will be used:

U_c	–	Username of the client
P_c	–	Password of the client
P_o	–	Password of the data owner
D_s	–	Destination LDAP server and port
S_c	–	Shared (symmetric) secret key of the client
S_{pd}	–	Secret key used by the LDAP proxy server for directory data
S_{pa}	–	Secret key used by the LDAP proxy server for authentication strings
K'_c	–	Private key of the client

K_p	–	Public key of the LDAP Proxy server
K'_p	–	Private key of the LDAP Proxy server
ACL_R	–	Set of users in ACL with read permission
ACL_W	–	Set of users in ACL with write permission
$\{M\}_H$	–	Hashed message
$\{M\}_{K_p}$	–	Message encrypted by the public key of the LDAP Proxy server
$\{M\}_{K'_p}$	–	Message encrypted by the private key of the LDAP Proxy server

The protocol is described in two parts. The first part is considered to be a *passive* configuration. In this case, the LDAP proxy server only encrypts and decrypts attributes as requested by clients. No prior interaction is necessary between the clients and LDAP proxy servers to initiate encryption and decryption routines. The second part is considered an *active* configuration. Using this method, prior communication is necessary between the client and LDAP proxy before encryption and decryption can commence. This involves the client obtaining an encrypted string from the LDAP proxy server to use for authentication requests.

In both cases, several assumptions are made. First, we assume that all communication between the client and LDAP proxy, as well as between the LDAP proxy and destination LDAP server, are done using SSL, TLS, or some other secure channel of communication. Next, we assume that some prior configuration has taken place, during which certain attributes are identified to the LDAP proxy server as encrypted attributes. We also assume that client applications can be reconfigured to specify username strings of arbitrary length. The current method for LDAP authentication and communication between client and server is shown in Figure 1.

Client	*Secure communication*	Server
Send auth. request (Username and password)	$\longleftrightarrow$	Authenticate client
Send LDAP operation (Search, add, modify, etc.)	$\longleftrightarrow$	Process request and return data

Fig. 1. Standard LDAP authentication and communication protocol

4.1 A Passive LDAP proxy

To manage attribute encryption and decryption using a passive LDAP proxy, the client first creates an *authentication string*, which is an encrypted string based on several components that replaces the regular username component for the client. The construction of this string is shown in Figure 2.

This internal part of this string is the concatenation of the client's username, the destination LDAP server and port, and a shared secret key of the client. These components are then encrypted by the client's private key, K'_c, and further encrypted using the LDAP proxy's public key, K_p. The resulting string is used as the client's username in LDAP authentication requests, and provides the LDAP proxy with information about the intended destination LDAP server, as well as the key to use for encrypting and decrypting directory data, S_c.

$$\left\{\{U_c \mid D_s \mid S_c\}_{K'_c}\right\}_{K_p}$$

Fig. 2. Authentication string for passive LDAP proxy configuration

To authenticate, the client provides the correct password for the username (U_c) included in the authentication string, and sends an authentication request to the LDAP proxy. The LDAP proxy receives this request in the form described in Figure 2 and decrypts the string using the private key of the LDAP proxy, K'_p, and the public key of the client, K_c. Next, the LDAP proxy passes the decrypted username and password to the destination LDAP server and awaits an authentication success or failure notice, which is then relayed back to the client. At this point, the client has authenticated to the destination LDAP server, and can begin LDAP operations[1].

LDAP operations at this point can be split into two categories, *read* and *write*. Read operations include "Search," and "Compare," and write operations include "Add," "Modify," and "Delete." To read and write unencrypted directory attributes, the LDAP proxy simply relays each operation and result between the client to the destination LDAP server. Reading and writing encrypted directory attributes is more complicated. These attributes are identified by the LDAP proxy as encrypted through prior system configuration.

Client	Secure comm.	LDAP proxy	Secure comm.	Dest. server
Send *data*	$\longrightarrow$	Encrypt *data* with S_c and S_{pd}	$\longrightarrow$	$\left\{\{data\}_{S_c}\right\}_{S_{pd}}$

Fig. 3. Encrypted data stored in destination LDAP directory using passive LDAP proxy

Client	Secure comm.	LDAP proxy	Secure comm.	Dest. server
Receive *data*	$\longleftarrow$	Decrypt *data* with S_c and S_{pd}	$\longleftarrow$	$\left\{\{data\}_{S_c}\right\}_{S_{pd}}$

Fig. 4. Encrypted data retrieved from destination LDAP directory using passive LDAP proxy

To write to an encrypted attribute, the LDAP proxy first encrypts the data using the client's shared secret key, S_c, then further encrypts the data using its own secret key, S_{pd}, before storing the encrypted string in the destination directory (Figure 3.)

To read an encrypted attribute, the LDAP proxy first retrieves the data from the destination directory. It then decrypts the information using its own secret key, S_{pd}, and the client's secret key, S_c, and returns the data to the client (Figure 4).

[1] This authentication method describes a "simple bind." Other authentication mechanisms, such as Simple Authentication and Security Layer (SASL) will be detailed later in this paper.

4.2 An Active LDAP Proxy

To manage attribute encryption and decryption using an active LDAP proxy, the *authentication string* is not created by the client, but rather by the LDAP proxy. The interaction between the client and LDAP proxy to accomplish this task should be carried out via a secure user interface, such as a web service. To establish the authentication string, the client must provide the LDAP proxy with the following information:

- Client username, U_c
- Destination server and port, D_s
- Client secret key, S_c
- Client password hash, $\{P_c\}_H$

The LDAP proxy concatenates this information and encrypts it using a symmetric key used only for authentication string functions, S_{pa}. The construction of this string is shown in Figure 5.

$$\{U_c \mid D_s \mid S_c \mid \{P_c\}_H\}_{S_{pa}}$$

Fig. 5. Authentication string for active LDAP proxy configuration

Authentication between the client and the destination LDAP server is slightly different with this configuration. In addition to authenticating the client's credentials with the destination LDAP server, the LDAP proxy checks the encrypted password hash value, $\{P_c\}_H$, against the hash value of the password provided by the client, $\{P'_c\}_H$. If these values match, then authentication is permitted. If not, it fails.

The addition of the $\{P_x\}_H$ component (where $\{P_x\}_H$ is the hash value of the client password, regardless of whether that particular client originally encrypted the data) affords additional capabilities, such as attribute ownership and detailed access control. This can be specified using $\{P_x\}_H$ as either the owner identifier or an ACL entry identifier for an encrypted directory attribute. The additional information is stored along with the data in the destination directory entry. Like authentication string generation, the owner and authorized users for a particular attribute are specified prior to LDAP operations, using a separate secure channel.

Reading and writing unencrypted data functions exactly as described in the passive LDAP proxy configuration. Reading and writing encrypted data, however, is achieved slightly differently. The read and write operations for an active LDAP proxy are shown in Figures 6 and 7.

Client	Secure comm.	LDAP proxy	Secure comm.	Dest. server
Auth. string + P'_c	$\longrightarrow$	Authenticate client using $\{U_c \mid D_s \mid S_c \mid \{P_c\}_H\}_{S_{pa}}$		
		Check $\{P'_C\}_H = \{P_C\}_H$		
Recv. auth.	$\longleftrightarrow$	If match, continue authentication	$\longleftrightarrow$	Authenticate client
$data'$	$\longrightarrow$	Receive $data'$		
		$\{\{data \mid \{P_O\}_H \mid ACL\}_{S_c}\}_{S_{pd}}$	$\longleftarrow$	$\{\{data \mid \{P_O\}_H \mid ACL\}_{S_c}\}_{S_{pd}}$
		Check $\{P_C\}_H = \{P_O\}_H$ or $\{P_C\}_H \in ACL_W$		
		If match, encrypt using S_c and S_p, then save	$\longrightarrow$	$\{\{data' \mid \{P_O\}_H \mid ACL\}_{S_c}\}_{S_{pd}}$
	$\longleftarrow$	Acknowledge success		

Fig. 6. Writing encrypted data to a directory using an active LDAP proxy

Client	Secure comm.	LDAP proxy	Secure comm.	Dest. server
Auth. string + P'_c	$\longrightarrow$	Authenticate client using $\{U_c \mid D_s \mid S_c \mid \{P_c\}_H\}_{S_{pa}}$		
		Check $\{P'_C\}_H = \{P_C\}_H$		
Recv. auth.	$\longleftrightarrow$	If match, continue authentication	$\longleftrightarrow$	Authenticate client
Request	$\longrightarrow$	Receive request		
		$\{\{data \mid \{P_O\}_H \mid ACL\}_{S_c}\}_{S_{pd}}$	$\longleftarrow$	$\{\{data \mid \{P_O\}_H \mid ACL\}_{S_c}\}_{S_{pd}}$
		Check $\{P_C\}_H = \{P_O\}_H$ or $\{P_C\}_H \in ACL_R$		
	$\longleftarrow$	If match, decrypt using S_c provided by client		

Fig. 7. Reading encrypted data from a directory using an active LDAP proxy

5. Analysis and Discussion

The protocol for encrypting directory attributes described in this paper has several key aspects that make it a desirable approach to protecting sensitive directory information. We will consider the data protection this solution provides, discuss how it meets our design goals, and discuss potential attack scenarios. Additionally, we will compare the two configurations and discuss the advantages and disadvantages of each.

5.1 Data Protection

The primary contribution of this solution is directly related to our primary motivation – directory information is protected from unauthorized access. Data is stored in an encrypted form in the directory, encrypted using a key known only to the LDAP proxy server, S_{pd}[2]. This ensures that encrypted data can only be read by the LDAP proxy server itself. To provide an additional measure of security, the data is protected with a secret key, S_c. This not only restricts data access to users with knowledge of the secret key, S_c, but also ensures that a compromised LDAP proxy server key cannot be used to read protected information.

An additional measure of protection is provided when using the active LDAP proxy configuration. Not only is the data protected as previously described, but access to that data is further controlled by the LDAP proxy server. Only a user that provides the same password used to bootstrap the authentication string is allowed to access the data. This prevents a dedicated administrator from spoofing an authorized user by changing the user password to some known value. While doing so would allow the user to be authenticated at the destination LDAP server, it would not satisfy the requirements of the LDAP proxy, which matches the password provided with the password hash in the authentication string. The attacker would need to obtain a new authentication string from the LDAP proxy server, which would require knowledge of the old password to change. So even if S_c were compromised, the user's original password would also need to be compromised to access data protected by an active LDAP proxy configuration.

Therefore, protecting S_c is of critical importance. To keep the actual key safe, a user may choose to store it on removable media, or in some other encrypted form. However, the key must be presented to the LDAP proxy server to allow for data encryption and decryption. This is why the encrypted authentication strings are applied: to protect S_c. For a passive LDAP proxy configuration, the authentication string is encrypted first using the client's private key, K'_c, which helps to prevent an attacker from creating his own authentication string with the original client's username and password, because only the client would be able to correctly encode this first part of the authentication string[3]. This encrypted value is then further encrypted, using the LDAP proxy server's public key, K_p, which ensures that only the LDAP proxy server is able to decrypt the string. Once decrypted, it can confirm the message was origi-

[2] We assume the LDAP proxy server and its keys are administered securely, and have not been compromised by a dedicated administrator.

[3] As before, we assume the client's public/private key has not been compromised by an attacker.

nally encrypted by the client, by using the client's public key, K_c, obtained from an enterprise certificate authority (CA).

5.2 Addressing Design Goals

Several design goals were presented in Section 3.2. We will now discuss how this solution meets those goals.

5.2.1 Adhere to Existing LDAP Standards

When presenting a new communication protocol, which enhances an existing protocol, it is critical that the new solution does not severely limit the functionality provided by the original. Our solution provides complete support for LDAP functionality on unencrypted data by simply acting as a data proxy, passing client requests directly to the destination LDAP server, and relaying responses back to the client. However, with encrypted attributes, we must carefully consider whether or not existing LDAP functions can operate as expected. LDAPv3 [17] provides support for the following operations: StartTLS, Bind, Unbind, Search, Modify, Add, Delete, ModifyDN, Compare, Abandon, Unsolicited, and Extended.

Briefly we will describe those operations that are easy for the LDAP proxy to relay between the client and destination LDAP server, and that do not require further explanation. *Unbind* simply closes an LDAP connection. *ModifyDN* is a way to change the unique identifier of an object in a directory. Since the unique identifier of an object should not be duplicated, it should be stored in unencrypted form, and thus would not be handled by the LDAP proxy. *Abandon* allows a client to request that an uncompleted operation be abandoned by the server. *Unsolicited* is a method for the LDAP server to send the client an unsolicited message, usually pertaining to the condition of the server. *Extended* operations allow the LDAP protocol to be extended to support additional functionality. These would need to be handled on a per-case basis, but we will generally assume that such messages could be easily relayed by the LDAP proxy.

The *StartTLS* operation allows the client and server to establish a TLS channel prior to any LDAP communication. Since the client in our solution intends to communicate directly with the LDAP proxy, this operation will be supported. The LDAP proxy will establish its own secure channel with the destination LDAP server. The security of this channel depends on the capabilities of the destination server, which may or may not include TLS or SSL.

Bind is the authentication mechanism of LDAP. Client credentials, such as unique identifier (username) and password are passed to an LDAP server for authentication. Sometimes this is done using a "simple bind," in which the username and password are sent in clear-text between the client and server. This is a common practice among directory services, and is protected by the underlying transport security of SSL, TLS, etc. [18]. Other methods of authentication include SASL [19]. In each instance, the authentication mechanism can be carried out via proxy. Essentially, this is a desired "man-in-the-middle" attack, where the LDAP proxy is the "man-in-the-middle." Usually, this attack is prevented by using a secure channel, and actual "man-in-the-middle" attacks on this protocol are prevented by the use of secure channels between the client and the LDAP proxy, and between the LDAP proxy and the destination LDAP server.

Search operations find directory entries specified by parameters sent by the client. The response is a list of the objects that match the search parameters, including certain attributes. The limitation of our solution with search operations is that encrypted values could not be included in the search string, also called a search *filter*. If this were allowed, the LDAP proxy would have to decrypt all values of a particular attribute for all objects to check for a match. Not only would this be time-consuming, but the requesting user may not have access to other objects' attributes. However, encrypted attributes could be returned in the response to the search request. The LDAP proxy would handle decrypting these values if permitted.

Compare operations can be easily handled by the LDAP proxy server. A compare operation is a request to the LDAP server to check if a certain assertion is true for a particular directory entry. If permitted, according to credential supplied with the authentication string, the LDAP proxy would handle this comparison, after decrypting the necessary values. The result would be returned to the client as expected.

For *Modify*, *Add*, and *Delete* operations, our proposed protocol describes the process and decisions involved at the LDAP proxy server. When adding or modifying an encrypted attribute, a passive LDAP proxy would simply write the encrypted value to the directory. An active LDAP proxy would first confirm the user had adequate permission before writing the value. Deleting an attribute using a passive LDAP proxy would be completed as expected, but with an active LDAP proxy, access controls could be implemented to specify which users had permission to perform delete operations.

5.2.2 No Additional Client or Server Software

The topology of our solution places the LDAP proxy server as a communication link between the client and destination LDAP server. The only changes necessary to the client are to replace the existing username string with a modified authentication string, which is either user-generated (passive configuration), or generated by the LDAP proxy (active configuration). The destination LDAP server requires no changes or additional software, unless specific configurations exist to limit communication from certain hosts, in which case, the LDAP proxy server would need to be added to that list.

5.2.3 Interoperable with Existing Clients and Servers

All communication between the client and LDAP proxy server occurs over standard LDAP ports, using standard LDAP operations. Likewise, all communication between the LDAP proxy server and destination LDAP server occur using standard LDAP communication, under the context of the client connected to the LDAP proxy server. One limitation to note is that some clients may not allow very long strings to be input as the username for bind operations. We find this to be unlikely, as distinguished names are often required to be specified as the client username, and these can be quite long. The new authentication string would be the length of the original username, plus the destination server name and port. This would be followed by S_c, which, as an example, would be 32 bytes long if a 256 bit key is used. Next, the hash of the user's password, which would be 20 bytes if a SHA-1 hash were used. Assuming the server name and port length was around 20 characters, the original length of the username string would be extended by around 72 characters. Because this value is then encoded using the LDAP proxy symmetric key, S_{pa}, the actual length of the string will be a multiple of the block size of the cipher used to encrypt the data. Symmetric encryption was selected over asymmetric encryption in this case (i.e. encrypting with K'_p and then K_p) due to performance issues. While the asymmetric cipher would result in a smaller ciphertext, the performance gain of doing a single symmetric cipher outweighs the additional space used to store the result. For block sizes of 256 bits, the output would be in multiples of 32 bytes, which is probably not going to make any difference to a client application able to store long username values anyway. However, this may cause some problems for clients with severely limited username fields.

5.2.4 Share Access to Encrypted Information with Other Users

Sharing access to protected information is an important consideration for some users. Normally, access is controlled via ACLs, but standard ACLs are subject to attack. This solution provides a method for users to share access to encrypted information with other users, while maintaining the security of the data.

For passive LDAP proxy configurations, all the user needs to share is S_c. Another user can use S_c to create an authentication string of their own, and use that string to request access to the data via the LDAP proxy. For an active configuration, not only does S_c need to be shared, but additional action must be completed by the data owner (who specified in the data stored by the LDAP proxy server in the destination directory.) The owner must obtain a hash of the additional user's password $\{P_x\}_H$ and apply that to the access control structure maintained by the LDAP proxy. The additional user must also interact with the LDAP proxy to obtain an encrypted authentication string.

5.2.5 Easy to Use

The previous sections detail how the solution works with existing LDAP clients and servers without detailed configuration changes. The only other interaction required by the user would be when using an active configuration. In this case, the user would need to use the LDAP proxy server to obtain an authentication string and specify access controls. A well-designed web interface would make this interaction easy for users, proving them with the means to specify which attributes should be marked as encrypted, which other users should have access, and so on. Additionally, a web form could be used to accept user input to generate the authentication string. Some difficulty could arise when users as asked to enter $\{P_x\}_H$, the hash of another user's password. This could be alleviated by the implementation of another simple interface the other user could use to create a password hash, given the password as input. As before, we assume these interactions are protected by secure communication protocols (i.e. HTTPS), and that the LDAP proxy server has not been compromised.

5.2.6 Manage Encrypted and Unencrypted Attributes without Changing Client or Server Configuration

We have discussed how a small configuration change is required on the client to enable interaction with the LDAP proxy. The intention of this design goal is to allow the client to process both encrypted and unencrypted data without having to further change the authentication string specified in the bind operation. The solution to this challenge is to allow the LDAP proxy server to determine if an attribute in the destination directory is encrypted. If so, the attribute is processed using credentials supplied by the user in the authentication string. If not, the LDAP server processes the request using the same credentials supplied, but does not need to perform any encryption, decryption, or access control checking. It should be noted that while this solution will allow any client to secure data in any LDAP server, a unique authentication string will be need to be generated for each destination server.

5.3 Attack Scenarios

The primary goal of this solution is to prevent unauthorized access of protected directory information. This does not include preventing an attacker from replacing existing data with incorrect values, another insider threat. We will not address that issue, but instead will focus on the methods by which a dedicated administrative attacker could try to compromise protected data.

In [20], several threats are presented for Active Directory. These threats can generally be extended to model threats to other LDAP directories as well, and we will address two of these threats in particular, *Spoofing* and *Elevation of Privileges*. Spoofing is the attempt by the attacker to act as either a specific client or intended server, without the knowledge of the other party. Elevation of Privileges is the unauthorized modification of access controls on directory objects to allow an attacker access to the data. Both of these threats are likely methods by which an attacker could attempt to compromise the solution presented here. Other threats are possible, but many are mitigated by the use of a secure channel, or by encrypting object data, which our solution provides by default.

5.3.1 Spoofing

There are several different ways an attacker could spoof the client. Without compromising the LDAP proxy, however, the use of PKI will prevent the LDAP proxy from being spoofed as the server. To spoof the client, the attacker would need to gain access to different pieces of information, depending on whether a passive or active LDAP proxy configuration was used. If using the passive configuration, the attacker would need to obtain S_c to be able to read protected data. Additionally, if the attacker wanted to truly spoof the client, that is, make it appear as if the client had accessed the data, the attacker would need to either compromise the user password, or reset the password to one known to the attacker, and compromise or reset the user's public key, K_c. With that knowledge, the attacker would then create an authentication string using the original username, destination server and port, and the compromised S_c, and encrypt it using the compromised or reset public key, K_c. That would then be provided with the compromised or reset password to successfully connect, bind, and access the protected data from the directory server.

When considering an active configuration, however, the attacker must compromise more than just S_c. To mount a successful attack, the attacker must also compromise the original password used by the client to encrypt the data. Without that component, a duplicate authentication string could not be generated using the LDAP proxy, and the attacker could not match the $\{P_c\}_H$ value in the authentication string with the correct P_c as the password.

Another possibility for a spoofing attack would be to compromise the authentication string itself. Depending on the configuration of the client's LDAP application, the authentication string may be stored in an easily accessible location. Compromising the entire string would allow the attacker to access protected data in a passive configuration (assuming the attacker could successfully authenticate to the destination LDAP server), but would not allow the attacker access to protected data in an active configuration. In this case, as before, the attacker would also need to know the correct password, P_c.

5.3.2 Elevation of Privileges

The second possible threat to our solution is by a dedicated administrator threat that targets the data in the directory itself, rather than the client application. The normal approach to this attack is to modify ACL settings on protected data, access the data, and replace ACL settings without being detected. Diligent auditing of such events can help mitigate this threat, but a dedicated administrator may have access to alter monitoring logs.

Our solution does not rely on traditional ACLs to protect the data in the directory. Rather, we rely on encryption using keys that the administrator does not have access to. By protecting the keys used by the LDAP proxy (S_{pd} and S_{pa}), the PKI keys used by both the LDAP proxy and the client (K_p, K'_p, K_c, and K'_c), and the client's secret key (S_c), we keep an attacker from compromising the data encrypted in the directory. This may seem like a difficult task, but PKI keys are generally well protected by the CA, and secure administration techniques, such as key hiding or obfuscation, can keep S_p secret from even the server administrator. The weakest link in preventing this type of attack is the security of S_c. Even if this key were compromised, and the attacker had the other components necessary to decrypt the data, only the information encrypted by S_c would be vulnerable to disclosure.

5.4 Comparison

Comparing the two methods, active and passive LDAP proxy, it is clear that the active configuration provides more protection against dedicated administrator attacks. However, this protection comes at the expense of performance. Validating that a client's authentication string contains the $\{P_c\}_H$ value that matches the provided password takes time, as does matching requests for access against ACLs that contain other clients' $\{P_x\}_H$ values. Additionally, more user interaction is required with the active configuration, both to set up encrypted attributes, and to manage access to the data. Each time the user's password changes (or any authorized user's password changes), updates must be made to the ACL maintained on that data by the LDAP proxy. Therefore, for faster response time, with security that rests on the secrecy of S_c alone, the passive configuration is superior. For better security, where additional information must be compromised for an attack to succeed, but at lower response times, with greater user interaction necessary, the active configuration would be the best choice.

6. Conclusion and Future Work

We have described a protocol for encrypting directory services information. This protocol provides data protection without relying on additional software or extensive reconfiguration to either the client or server components. Rather, it adds an LDAP proxy server to handle encryption and decryption of data, based on information contained in an authentication string provided by the client. The configuration provides basic protection in a passive mode, and more comprehensive protection in an active mode, but at the expense of response time and increased user interaction. Future work includes exploring ways to prevent attackers from replacing existing data with incorrect data. Additional work could include implementing an LDAP proxy server and measuring the performance and impact against a real-world LDAP configuration. Possible measurements include response time of both the client and server, additional storage requirements in the destination LDAP server, and interoperability with a variety of existing client and server applications.

References

[1]. Shin, D., Ahn, G., Shenoy, P.: Ensuring Information Assurance in Federated Identity Management. In 23rd IEEE International Performance Computing and Communications Conference (IPCCC). Phoenix, Arizona (2004)

[2].Windows Server 2003 Active Directory,
http://www.microsoft.com/windowsserver2003/technologies/directory/activedirectory/default.mspx

[3]. IBM Tivoli Directory Server,
http://www-306.ibm.com/software/tivoli/products/directory-server/

[4]. Mac OS X Server Open Directory,
http://www.apple.com/server/macosx/opendirectory.html

[5]. Novell eDirectory, http://www.novell.com/products/edirectory/

[6]. Open LDAP, http://www.openldap.org/

[7]. Fedora Directory Server, http://directory.fedoraproject.org/

[8]. Sun Java System Directory Server, http://www.sun.com/software/products/directory_srvr/home_directory.xml

[9]. Carter, G.: LDAP System Administration. O'Reilly, USA (2003)

[10]. How to mark an attribute as confidential in Windows Server 2003 Service Pack 1, http://support.microsoft.com/kb/922836

[11]. Claycomb, W., Shin, D., Hareland, D.: Towards Privacy in Enterprise Directory Services: A User-Centric Approach to Attribute Management. In: 41th IEEE International Carnahan Conference on Security Technology, Ottawa, Canada (2007)

[12]. Berger, A.: Privacy Protection for Public Directory Services. In: Computer Networks and ISDN Systems vol. 30, pp. 1521—1529. Elsevier Science Publishers B. V (1998)

[13]. Stokes, E., Milman, I.: Method for Securing Sensitive Data in a LDAP Directory Service Utilizing a Client and/or Server Control. US Patent No. 6339827, Jan. 15, 2002.

[14]. Kowalski, E., Cappelli, D., Conway, T., Willke, B., Keverline, S., Moore, A., Williams, M.: Insider Threat Study: Illicit Cyber Activity in the Government Sector. U.S. Secret Service and CERT/SEI, Jan. 2008.

[15]. Keeney, M., Capelli, D., Kowalski, E., Moore, A., Shimeall, T., Rogers, S.: Insider Threat Study: Computer System Sabotage in Critical Infrastructure Sectors. U.S. Secret Service and CERT/SEI, May 2005.

[16]. Shaw, E., Ruby, K., Post, J.: The Insider Threat to Information Systems. In: Security Awareness Bulletin, No 2-98. Department of Defense Security Institute, Sept. 1998.

[17]. J. Sermersheim, Ed.: Lightweight Directory Access Protocol (LDAP): The Protocol. IETF RFC 4511, June 2006.

[18]. ADSI Does a Simple Bind When You Specify ADS_USE_SSL, http://support.microsoft.com/kb/321315

[19]. A. Milnikov, Ed.: Simple Authentication and Security Layer (SASL). IETF RFC 2222, June 2006.

[20]. Chadwick, D.: Threat Modeling for Active Directory. In: Communications and Multi-media Security, IFIP, vol. 175, pp. 173—182. Springer, Boston (2005)