Journal of Universal Computer Science, vol. 15, no. 15 (2009), 2999-3018
submitted: 1/2/09, accepted: 29/8/09, appeared: 1/9/09 © J.UCS

A User Controlled Approach for Securing Sensitive Information in Directory Services

William Claycomb
(Sandia National Laboratories
Albuquerque, New Mexico, USA
wrclayc@sandia.gov)

Dongwan Shin
(Secure Computing Laboratory
Computer Science Department, New Mexico Tech University
Socorro, New Mexico, USA
doshin@nmt.edu)

Abstract: Enterprise directory services are commonly used in enterprise systems to store object information relating to employees, computers, contacts, etc. These stores can act as information providers or sources for authentication and access control decisions, and could potentially contain sensitive information. An insider attack, particularly if carried out using administrative privileges, could compromise large amounts of directory information. We present two solutions for protecting directory services information from insider attacks. The first is a centralized approach utilizing a customized virtual directory server. The second is a distributed approach using existing key management infrastructure and a new component called a Personal Virtual Directory Service. We explain how these solutions interact with existing directory services and client applications. We also show how impact to existing users, client applications, and directory services are minimized, and how we prevent insider attacks from revealing protected data. We compare and contrast both solutions, including potential tradeoffs, administrative overhead, and enterprise systems impact. Additionally, our solution is supported by implementation results showing the impact to client performance and directory storage capacity.

Key Words: Directory, Security and Protection
Category: SD H.2.7, K.6.5

1 Introduction

Enterprise directory services (EDS) are commonly used in enterprise systems to store information pertaining various directory objects, such as users, computers, or contacts. EDS are used to share information with others, such as address books, or as authoritative sources for authentication and access control. In most cases, this dual role is combined in the same directory service instance.

Generally, organizations seek to establish interoperability and seamless communication between heterogeneous systems in enterprise systems, and directory services enable them to do so. Using standard communication protocols, such as

Lightweight Directory Access Protocol (LDAP), EDS may provide authentication services or information to multiple requestors, independent of platform or implementation. This is particularly advantageous to organizations seeking to consolidate multiple authoritative information sources into a single repository.

However, this move towards centralized information services has serious drawbacks. In particular, as the amount of information stored increases, the potential for storing sensitive information increases. This is especially true in instances where certain pieces of information are not necessarily sensitive when stored separately, but become sensitive when combined. Consider attributes such as department number and security level. Knowing which people are in a certain department is not necessarily sensitive. Likewise, knowing which people hold a certain security level is not necessarily sensitive, particularly if many people hold the same level. However, if combined, then a very specific set of people can be identified - those in a particular department holding a higher security level. That information could be used to specifically target those individuals for context-aware, or spear phishing [Jakobsson 2005] attacks, where individuals are targeted and the attack appears to come from a legitimate sender, such as a colleague.

Another potential hazard when consolidating multiple directory services into a single EDS is the inclusion of certain information not meant to be shared among larger sets of users. Such tightly controlled information could include attributes considered to be personally identifiable information (PII) such as employee numbers, or other confidential information such as bank account numbers. In these cases, the intended users are a small subset, such as the human resources or payroll offices only.

Protecting this information is critical, and most directory services solutions provide methods for limiting access. However, such measures can usually be circumvented by system administrators, or those with elevated privileges. These users may obtain access to sensitive directory information in more than one way. For instance, they might override existing access control methods, or they may impersonate an authorized user to gain access to the information. Another method would be to simply copy the entire directory information store to attempt extraction of sensitive information.

We propose two methods for protecting sensitive directory services information from all users, including system administrators, using encryption. Furthermore, we base our solutions on existing infrastructure commonly used in enterprise systems. Recognizing that heterogeneous organizations require different solutions, we propose both centralized and distributed approaches to this problem. The first solution builds on technology called *virtual directories* to customize a solution for protecting sensitive information. This solution, called *Enhanced Virtual Directory Services*, is based on existing user passwords, and prevents dedicated administrator attacks while allowing users to delegate access control to authorized users. Because virtual directories are centralized by nature, this solution maintains many of the advantages of a centralized approach. The second approach involves the introduction of a new type of virtual directory service, called a *personal virtual directory service (PVDS)*, and presents a more distributed solution. The PVDS interfaces with a key management system (KMS) and handles encryption and decryption of sensitive information at the client level.

To support our proposed solutions, we carefully analyze the advantages and disadvantages of both approaches, comparing and contrasting their features such as administrative overhead, client application impact, and enterprise systems impact. Additionally, we show how our solutions' impact to existing directory services is minimal, in terms of directory size and performance. Finally, we demonstrate how our solutions mitigate various attacks on virtual directories, including the insider attack, where the attacker uses domain administrator privileges to attack a directory service.

The remainder of this paper is organized as follows. Section 2 presents related work and previous approaches, followed by Section 3, which details our approaches. In Section 4, we discuss the advantages of our solutions, compare and contrast them, list various attack models, and show implementation results. Section 5 concludes the paper with suggestions for future work.

2 Background

The threat of unauthorized access of sensitive data by employees or other authorized users, known as "dedicated insiders", is well documented [Kowalski et al. 2008,Keeney et al. 2005,Shaw et al. 1998]. In January 2008, the U.S. Secret Service and CERT issued a report titled "Insider Threat Study: Illicit Cyber Activity in the Government Sector" [Kowalski et al. 2008]. This study outlines a multi-year project, started in 2002, that explores the activity and threats posed by insiders. Among the key findings of this study 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

2.1 Previous Approaches

The structure of EDS is hierarchical, with each leaf representing an *object*. Objects are described by individual *attributes*, such as name, title, password, etc. Solutions for protecting directory services are generally implementation-specific, and rely largely on per-attribute access control lists (ACLs). Other directory services instances generalize this approach by using the concept of *confidential* attributes [Microsoft Corporation 2007], but the underlying implementation is still ACL-based. The use of encryption in directory services is very limited, with specific implementations employing encryption for every instance of certain attributes across the entire directory [Red Hat, Inc. 2005]. However, this approach uses a single server-based key for encrypting all attributes.

A user-centric approach to protecting directory attributes is described in [Claycomb et al. 2007]. This method is not dependent on a particular directory implementation. Rather, it utilizes user-based public/private keys to allow users control of encrypted attributes related to their own directory information. This solution describes different methods for using public/private keys to ensure either data authenticity alone, or data authenticity combined with confidentiality. Specific solutions are proposed for scalability and usability purposes. However, key control is maintained by the user, which raises issues of maintainability and ease-of-use.

2.2 Virtual Directories

"A *virtual directory* functions as an abstraction layer between applications and data repositories." [Radiant Logic, Inc. 2008] In contrast to *metadirectories*, virtual directories do not maintain data in a standalone data source. Rather, virtual directories reference various data sources and present a consolidated view to the end user. This has the advantage of not requiring data synchronization - the data presented is always real-time, directly from the source. Most virtual directory implementations have the additional capability of acquiring data from sources other than directories, such as databases, and presenting this information to end users via LDAP [Radiant Logic, Inc. 2009].

3 Our Approach

We propose two solutions for protecting sensitive information in directory services. The first is a centralized approach, which adds various components to standard virtual directory services. These components interact with client applications to handle encryption, decryption, and delegation of access to sensitive information. We refer to this solution as *Enhanced Virtual Directory Services* (EVDS). The second solution is a distributed approach, which performs many of the same functions, but is handled at the client machine level, and interfaces directly with key management infrastructure to enable protection and delegation of directory information. We call this approach a *Personal Virtual Directory Service* (PVDS).

3.1 EVDS - A Centralized Approach

The centralized approach to protecting sensitive information in directory services is to encrypt that information using user-controlled keys and to provide access to that data using user-controlled delegation. This user-centric approach follows current trends in computer security and privacy, but should not interfere with more traditional approaches to access control. Our approach also maintains usability with existing client applications and source directories. To better understand the overall picture of our solution, it is first important to understand various key components.

3.1.1 Data Encryption

Encrypting sensitive information to protect it from misuse is hardly a new concept. In the simplest application, towards protecting information in directory services, an end user would simply encrypt sensitive information and then store the encrypted data in a directory.[1] To share information, the user would share the encryption/decryption key with another user, who would obtain the encrypted form from the directory and decrypt it locally.

However, the EVDS approach presents several usability and security problems. First of all, the confidentiality of the data relies entirely on the shared key. If a malicious user were to obtain this key, or if an authorized user were to share it with an unauthorized party, the information could be compromised. Data confidentiality could be provided by using an asymmetric encryption algorithm, such as RSA, but this still does not protect the data from unauthorized access.

Secondly, the EVDS approach requires the user to perform encryption and decryption before and after retrieving the information from the directory. At best, this could be accomplished by a custom application, which interfaced directly with the client LDAP application. At worst, existing client LDAP applications would need to be rewritten to incorporate encryption and decryption. This is an undesirable situation for which a simple solution exists: add a third party, between the client and server, to handle encrypting and decrypting the data.

[1] Note that encryption here is orthogonal to that of secure LDAP (or LDAPs). The former is for data protection in data stores while the latter is for network communication protection. It is assumed that LDAPs is supported for better security in our approach.

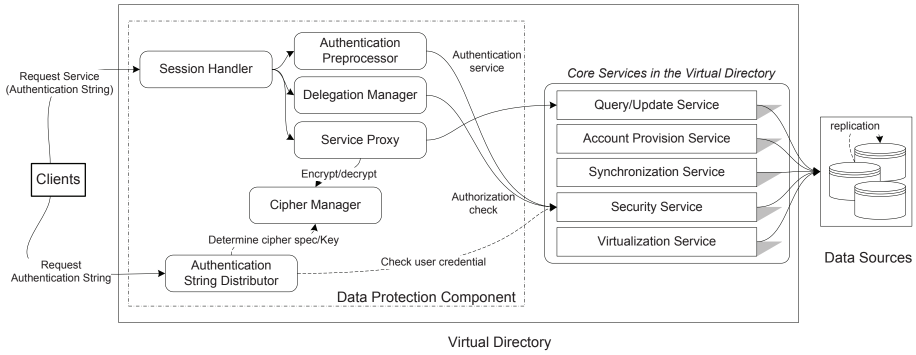

Figure 1: EVDS: System Architecture for Protecting Sensitive Information in Directory Service using Virtual Directory

This third party component could be a custom component written specifically for the purpose of handling encryption and decryption of information between the client and directory. However, we find it much more useful to leverage the existing technology of virtual directories to provide the third party component to the model. The benefits of doing so are numerous, and will be discussed in detail later.

3.1.2 System Architecture

If we consider a virtual directory as the container of the third party component - the one responsible for encrypting and decrypting data - we must consider several key aspects, namely: how does the virtual directory obtain key information from the client, how does the virtual directory perform pass-through authentication to destination directories, and how does the virtual directory manipulate the data in the destination directory? The EVDS system architecture, shown in Figure 1, which enhances standard virtual directory services to include these features, is proposed to address those questions.

3.1.2.1 Obtaining client key information

When LDAP communications occur between a client and server, several standard pieces of information are transmitted. These components are generally configured by the client application, and can be changed by the end user. They are: username, password, and destination server name and port. We leverage these components to pass encryption information to the server as follows. The destination server and port are replaced with the destination server name and port of the virtual directory. This configures the client to communicate with the virtual directory, instead of the original destination directory. Note that the original destination directory is transparent to the client through virtualization, which is one of the core services in virtual directories, as shown in Figure 1. The password remains the same as the original password used to authenticate to the original destination directory.[2] We replace the username component with a string which is the concatenation of the following: the client username, ID_c, the hash of the original user password, $H(pwd_c)$, and a symmetric key between the client and virtual directory, K_{cv}. The last two components are encrypted using a secret key known only to the virtual directory server, K_v. The addition of these last components requires additional setup, performed by the Authentication String Distributor shown in Figure 1 with access to the virtual directory key, K_v, and is also discussed in detail later. The resulting string is called an *authentication string (AS)*:

$$ID_c|\{K_{cv}|H(pwd_c)\}_{K_v}$$

3.1.2.2 Performing pass-through authentication

We do not ignore traditional authentication and access control methods with the EVDS solution. Unless configured for anonymous authentication (also called *anonymous bind*), the destination LDAP server will expect a client to authenticate prior to data retrieval. Some virtual directory implementations allow a static username and password to be used for every transaction, but this defeats the purpose of fine-grained access control. Rather, we will pass the original client username and password, obtained from the AS and password provided by the client, to perform an initial bind prior to data retrieval. If this bind is not successful, then no data transmission occurs between the virtual directory and the client.

3.1.2.3 Storing the data

Once the user has successfully authenticated to the destination directory, we use the transformation capabilities of the Authentication Preprocessor module in the EVDS architecture to extract the user's symmetric key, K_{cv}, and password hash, $H(pwd_c)$. The password hash is used as an additional measure of security against an attack where a malicious administrator may change the user's password and, using the original authentication string, masquerade as the user. While this step may seem redundant, it is necessary because of the nature of LDAP clients. Many LDAP clients allow users to cache login information, including the username. An attacker would need to have no knowledge of the client secret key, K_{cv}, if he used a cached authentication string and a newly-reset password. However, if the

[2] Pass-through authentication is more commonly practiced than single-sign-on, in virtual directories.

Client	Secure comm.	Virtual Directory Server	Secure comm.	Dest. server
Auth. string + pwd'_c	$\longrightarrow$	Authenticate client using $ID_c \vert \{K_{cv} \vert H\{pwd_c\}\}_{K_v}$		
		Verify $H\{pwd_c\} = H\{pwd'_c\}$		
Recv. auth.	$\longleftrightarrow$	If match, continue authentication	$\longleftrightarrow$	Authenticate client, using ID_c and pwd'_c
Request	$\longrightarrow$	Receive request		
		$\{data \vert H\{pwd_O\} \vert ACL\}_{K_{cv}}$	$\longleftarrow$	$\{data \vert H\{pwd_O\} \vert ACL\}_{K_{cv}}$
		Decrypt using K_{cv} provided by client, and check $H\{pwd_c\} = H\{pwd_O\}$ or $H\{pwd_c\} \in ACL_R$		
	$\longleftarrow$	If match, return to client		

Figure 2: EVDS - Reading an encrypted attribute

client were configured to prompt for a password every time, while still retaining a cached authentication string, the user's password hash could be checked against the password hash encrypted by the virtual directory's secret key in the AS. In this instance, a changed user password would cause a failure, because its hash would not match the original hash in the AS.

Once verified, the user's secret key, K_{cv}, is used to perform encryption or decryption of data stored in the directory. The protocol for reading an encrypted attribute is shown in Figure 2, and the protocol for writing an encrypted attribute is shown in Figure 3.

3.1.3 Collaboration and Delegation

One of the key components to our approach is the capability of the user to delegate access to attributes, enabling collaboration with other users. We modify a traditional Access Control List (ACL) model, by identifying the access control entry principal by password hash. If another user is delegated permission to access a particular attribute, the corresponding password hash, $H\{pwd_c\}$, must exist (read and/or write) in the ACL attached to the attribute when stored in

Client	Secure comm.	Virtual Directory Server	Secure comm.	Dest. server
Auth. string + pwd'_c	$\rightarrow$	Authenticate client using $ID_c\|\{K_{cv}\|H\{pwd_c\}\}_{K_v}$		
		Verify $H\{pwd_c\} = H\{pwd'_c\}$		
Recv. auth.	$\leftrightarrow$	If match, continue authentication	$\leftrightarrow$	Authenticate client, using ID_c and pwd'_c
$data'$	$\rightarrow$	Receive $data'$		
		$\{data\|H\{pwd_O\}\|ACL\}_{K_{cv}}$	$\leftarrow$	$\{data\|H\{pwd_O\}\|ACL\}_{K_{cv}}$
		Check $H\{pwd_c\} = H\{pwd_O\}$ or $H\{pwd_c\} \in ACL_W$		
		If match, encrypt using K_{cv} , then save	$\rightarrow$	$\{data\|H\{pwd_O\}\|ACL\}_{K_{cv}}$
	$\leftarrow$	Acknowledge success		

Figure 3: EVDS - Writing an encrypted attribute

the destination directory. Alternatively, if the attribute owner attempts to access the attribute, identified by $H\{pwd_o\}$, full access is granted. The ACL is managed by the virtual directory server, and again would require additional interaction by the attribute owner to manage. This is supported by the Delegation Manager in our system architecture.

3.2 PVDS - A Distributed Approach

Next, we propose an approach to protecting sensitive directory services information using encryption which does not rely on user-protected shared keys or passwords. We build on the EVDS solution by addressing various usability challenges and security concerns. Additionally, we simplify the model by eliminating the middle component, the VDS, and replace it with a novel approach to virtual directory technology, which we call a *personal virtual directory service (PVDS)*.

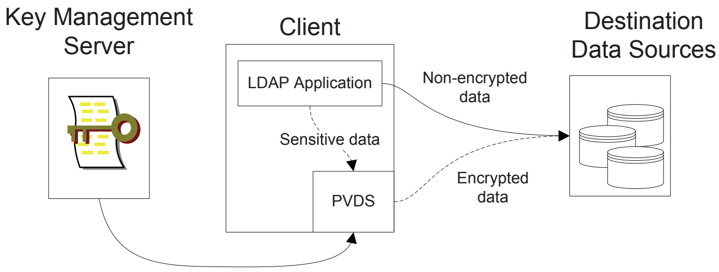

Figure 4: System architecture for a client system using a personal virtual directory service

3.2.1 Personal Virtual Directory Service

Not only does the VDS component of previous works require additional configuration and administration, but it serves as a target for attacks. If the server hosting the VDS is compromised, then all protected information it processes could be revealed to an attacker. We propose moving the virtual directory concept from a centralized configuration to a more distributed configuration. This is accomplished by running what is essentially a simplified VDS on each client machine - the PVDS.

The purpose of the PVDS is to handle communication between an LDAP client application and the destination directory. It is only used in cases where sensitive information needs to be processed - not all LDAP communications between the client and directory services need to be protected, however. In cases where sensitive information is not processed, standard communication should not be interrupted. This basic architecture is shown in Figure 4.

The PVDS is configured at a client level, instead of a centrally managed VDS. This has the potential to increase administrative overhead, as distributed applications tend to do. However, we will show that the function of the PVDS is largely self-contained, requiring no additional configuration once deployed. The necessary configuration to allow protected information to be processed occurs at the client level only, as with previous approaches. Therefore, the overall administrative overhead is actually reduced using our new model by removing the VDS component.

3.2.2 Using Existing Key Management Infrastructure

Instead of using user-controlled and user-protected keys, our solution makes use of exiting key management infrastructure to provide encryption and decryption information to the PVDS. This shifts the burden of key protection from systems not designed to protect sensitive information (directory services, virtual directories, and users) to systems specifically engineered to withstand attacks on keying information. Once a user authenticates to the KMS, the PVDS handles retrieving keying information and using it for data protection, as well as delegation.

3.2.3 Client Modification

Recall that EVDS *authentication strings*, controlled by the VDS, must be changed when any configuration modification occurs. Our approach simplifies this considerably. The only client modifications necessary to enable secure information protection are to replace the destination directory information with the local path to the PVDS, and to concatenate the username with the actual destination directory information. That is, the information contained in the username configuration of the LDAP client would be $ID_c \| dest_{LDAP}$, where ID_c is the username of the client and $dest_{LDAP}$ is the network address and port of the destination LDAP directory instance. No centrally managed authentication string is required.

3.2.4 Delegating Access

Because we use an existing key management infrastructure, delegation of access is fairly straightforward. Data owners may delegate permission to read and/or write protected data. Granting access, modifying access, and revoking access are controlled via an interface with the PVDS called a *Delegation Manager*. The delegation manager communicates directly with the KMS to obtain verified delegatee key information. Data protection information is encrypted using verified public keys of delegatees, ensuring that only intended parties have access to encryption and decryption keys.

3.2.5 Protecting the Data

Sensitive data is encrypted in the directory using a symmetric key S, which is chosen randomly and managed by a component of the PVDS called the *Cipher Manager*. A unique S is used for each attribute encrypted. In contrast to previous approaches, S is never known by the user, nor is it stored by the PVDS on the client machine. It only exists in the destination directory, encrypted by the public key of the owner, K_o, and any delegate users, K_u.

By adding the capability to delegate read/write access, we necessitate additional protection information. A public/private key pair $\{K_{rw}, K'_{rw}\}$ is generated for every protected attribute. K'_{rw} is used by authorized users to digitally

Key mgmt. svc.	Secure comm.	Client	Secure comm.	Dest. server
		Retrieve $\langle data \rangle =$ $\{\{data\}_S\}_{K'_{rw}} \Vert \{K_{rw}\}_{K'_o} \Vert ID_o \Vert \{S, K'_{rw}\}_{K_o} \Vert \{S, K'_{rw}\}_{K_{u1}} \Vert \ldots$	$\leftarrow$	$\langle data \rangle$
		Decrypt $\{S, K'_{rw}\}$ using K'_{u1}		
K_o	$\rightarrow$	Retrieve K_o		
		Using K_o, validate K_{rw} from $\{K_{rw}\}_{K'_o}$		
		Use K_{rw} to validate $\{data\}_S$		
		Use S to decrypt $data$		

Figure 5: PVDS - Reading an encrypted attribute

sign the encrypted data, $\{data\}_S$. Only users delegated write permission have access to K'_{rw}, which is encrypted (along with S) using the delegatee's public key K_u. K_{rw} is included in the data stored with the attribute, is signed by the data owner, and is used during all read operations to verify the authenticity of $\{data\}_S$. Using the data owner's public key, K_o (available from the KMS), the authenticity of K_{rw} can be verified.

An example of the combined form of all data stored in the directory for a protected attribute is shown as follows:

$$\{\{data\}_S\}_{K'_{rw}} \| \{K_{rw}\}_{K'_o} \| ID_o \| \{S, K'_{rw}\}_{K_o} \| \{S, K'_{rw}\}_{K_{u1}} \| \{S, \varnothing\}_{K_{u2}} \| \ldots$$

In this example, we know that User 1 has read and write access because K'_{rw} is included with S in the information protected by User 1's public key K_{u1}. Similarly, we know that User 2 only has read access, because only S is encrypted with K_{u2}. Additional users' access control information would also be included.

3.2.5.1 Reading Protected Information

The process for reading protected information, shown in Figure 5, begins with the PVDS X retrieving protected data from the LDAP server. The PVDS checks to see if client X can decrypt the protected information by detecting if S has been encrypted using K_X, in which case the PVDS decrypts S using K'_X. At this point, the data can be decrypted using S, but we need to verify it is authentic first. To do so, the PVDS retrieves K_o from the KMS using ID_o to uniquely

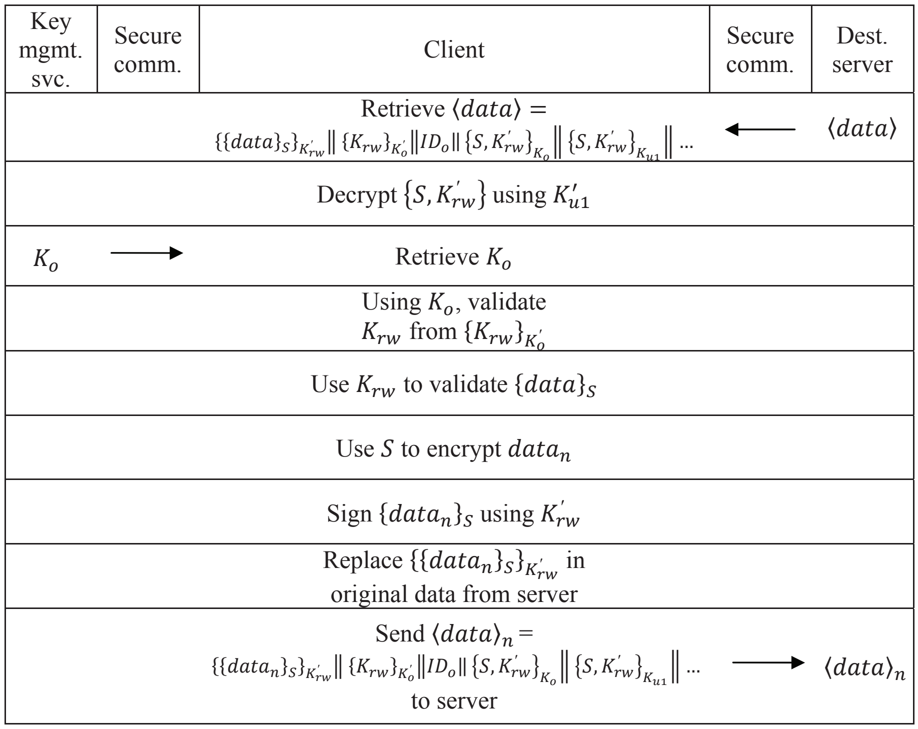

Figure 6: PVDS - Writing an encrypted attribute

identify the owner. With K_o, the signature on K_{rw} can be verified, and K_{rw} can be used to verify the signature on the data. Once verified, the PVDS passes the plaintext data to the local application.

3.2.5.2 Writing Protected Information

The process for writing protected information, shown in Figure 6, is similar to the process for reading. However, the PVDS must also verify X has access to write the data, specifically, does X know K'_{rw}? Any client with S can actually encrypt new information. However, only those clients with K'_{rw} can sign it. If K'_{rw} is found encrypted with K_X, it is decrypted and used to sign the modified data.

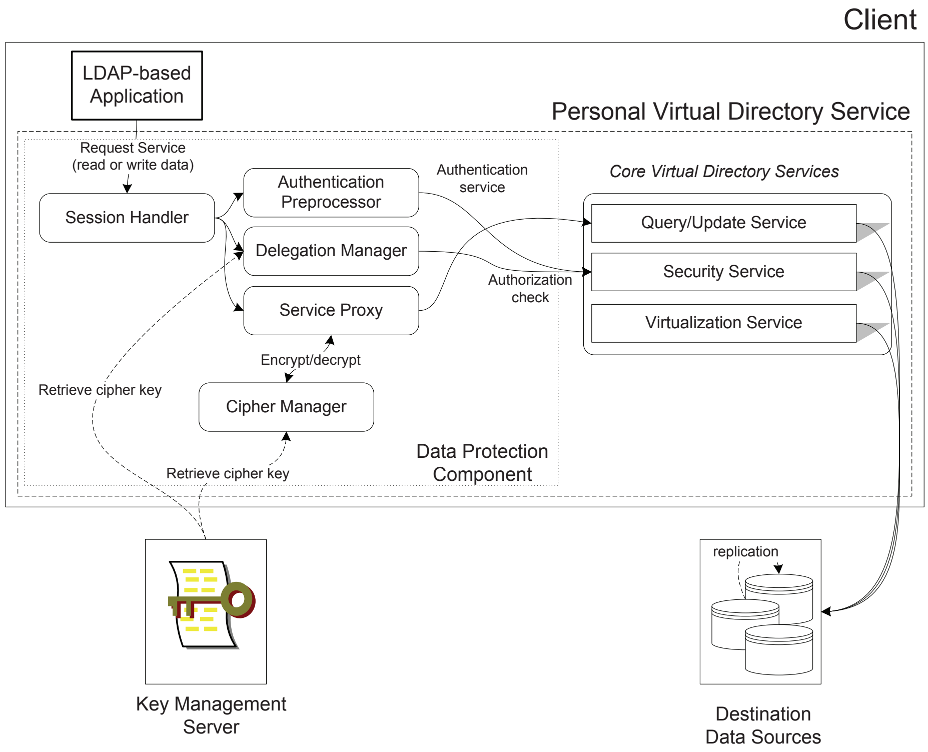

Figure 7: Personal Virtual Directory Service Framework

3.2.6 Putting It All Together

The interaction with the KMS when protecting sensitive directory services information requires various core components of virtual directories, as well as the creation of several new components to handle encryption, decryption, and delegation, as described above. We show the PVDS framework to describe the interaction between these components in Figure 7. As the core components of data protection have not changed from within the EVDS to the PVDS, please refer to § 3.1.2 for an explanation of these components.

4 Discussion

Although designed to serve the same purpose, and using similar technology, the two solutions proposed here have different impacts on enterprise systems. Among the affected areas include administrative overhead, client application impact, and enterprise systems impact. The impact to existing directory services should also be considered, in terms of directory size and performance. We will briefly compare and contrast both solutions here.

4.1 Enterprise Impact

Any solution involving clients and applications in an enterprise systems environment must address the administrative overhead involved with deployment and support. EVDS does not require additional client applications be installed, but does require LDAP client reconfiguration, to include the authentication string. Additionally, there is significant overhead involved with maintaining this approach, as any change to user passwords requires a change to the authentication string, and any change in passwords among the delegated users will result in the need to modify directory information. It is conceivable this could be automated to some extent, but doing so raises the possibility of additional administrator attacks. Changing password information in authentication strings and in protected information should require knowledge of both the new and old passwords.

Table 1: Time to retrieve directory services objects

Configuration	Time to retrieve object (ms)	Time to retrieve 1000 objects (ms)
Standard dir. services	3	534
Standard VDS	9	1090
EVDS	10	2440
PVDS	5	1510

4.1.1 Performance

While directory services interaction (creating, retrieving, modifying, deleting objects) is typically very fast and efficient, cryptographic operations, such as hashing, encrypting and decrypting information, and signing data is relatively slower and more computationally expensive. Most of the time in a complete LDAP request is consumed finding and returning the information from the directory store. For instance, the average time to retrieve a single user object (about 25 attributes) is 3 ms when connecting directly to the directory. When using a virtual directory, this time is about 9 ms. However, retrieving 1000 user object (25,000 attributes) only takes 534 ms for standard directory services, and 1090 ms for virtual directory services.

Adding data protection components slows this performance, understandably. Whether the cryptographic operations happen at the virtual directory level, as in EVDS, or at the client leve (PVDS), the effect to performance is similar. Table 1 shows the comparison of average times to retrieve user objects, with all information decrypted, using standard directory services (including virtual directory services) versus using our data protection solutions.

4.1.2 Administrative overhead

The PVDS approach we propose would involve similar administrative overhead in terms of client application reconfiguration. However, it would also require PVDS deployment to any client needing to access protected information. Many organizations have automated software management tools to assist this process. However, additional client components also require additional support for troubleshooting and maintenance. In contrast to the EVDS solution, however, no modification needs to be made at the directory or virtual directory level. For both solutions, no modifications are necessary to any data source, unless reconfiguration is necessary to allow the protected data to be stored in binary form.

4.1.3 Directory Services Impact

Measuring the impact to existing directory services is similar in both cases, because they both use directory services simply as storage for encrypted sensitive information. Therefore, the only impact we consider for directory services is the increase in storage required for the directory service on the host machine. We based our implementation on existing directory services instances, and used three protected attributes out of 55 total populated attributes per user.[3]. We tested each implementation on directory instances of Microsoft Active Directory Administration Mode SP1 [Microsoft Corporation 2009b] with 10,000 user objects. Table 2 shows the impact to directory service size for each solution.

4.2 Attacks

Attacks on directory services and virtual directories can take several forms. These range from simply stealing information from a directory, either using existing security loopholes or by gaining administrative access to the host machine and taking data directly from the source, to more complicated attacks exist involving

[3] In actual directory services where protected data is included, the actual number is probably 1-2 attributes per user

Table 2: Directory size on disk (MB)

Configuration	Beginning size (MB)	Final size (MB)
No data protection (no encryption)	6.3	61.5
EVDS	6.3	69.6
PVDS	6.3	100.4

directory services [Chadwick 2004] as well as virtual directory services [Claycomb & Shin 2009]. Both solutions presented here mitigate various attacks, including the dedicated insider attack, and also provide solutions to the specific attacks on directories and virtual directories.

4.2.1 Threats against Directory Services

Threats against directory services, specifically Microsoft Active Directory [Microsoft Corporation 2009a], are described in [Chadwick 2004], and include: spoofing, tampering, and information disclosure. Spoofing involves either masquerading as the requesting client or the source directory. This is prevented by our solution simply because in order to spoof a legitimate client, the attacker would need to possess the client's secret key. Masquerading as the source directory is also prevented, as the attacker could neither provide incorrect data (correctly encrypted, of course) to a requesting client, nor could the attacker gain any knowledge of data being sent to the directory, as it is already in encrypted form.

Tampering is the unauthorized modification of directory data, either within the directory itself or in transit. Again, because the information in these solutions is encrypted, tampering at the directory level is prevented. [Chadwick 2004] suggests using secure communication channels, such as LDAPS, to prevent this attack in general, and we reiterate this suggestion as an important concept of our overall architecture as well.

A user accessing data without proper authorization is known as information disclosure. In many cases, this is carried out by an insider, perhaps even with administrative privileges to the directory or the machine hosting the directory data. As previously mentioned, this type of attack could range from exploiting security loopholes, such as anonymous LDAP queries, to gaining administrative access to the source machine and stealing data directly from the directory itself. Because our solutions store directory data in encrypted form, and require a user-controlled key to decrypt, information disclose to a dedicated insider is prevented.

4.2.2 Attacks on Virtual Directories

Although threats to directory services have been outlined, little work has been done to address attacks specific to virtual directory services. The first attempt to address this subject appears in [Claycomb & Shin 2009], where four specific attack models against virtual directory services are outlined. These include authentication attacks, cache attacks, transformation attacks, and network disruption attacks.

An authentication attack involves the attacker stealing stored credentials from a virtual directory server, and using them to access unauthorized information in a source data repository, such as a directory or database. Fortunately, even if an attacker could gain access to the data source, our solutions prevent information disclosure by storing data in encrypted form. The attacker would still need the key to retrieve any data.

Many virtual directory services offer a solution for high-availability data needs by utilizing a local data cache of frequently accessed data. A cache attack occurs when an attacker gains access to this local cache, and either steals or modifies data contained within. To address this, an important architectural decision must be made. Specifically, sensitive information must never be allowed to exist in the cache. If a cache is used for sensitive user information, it must store that information in encrypted form, and decrypt it prior to delivery to the user.

Transformation attacks are very similar to cache attacks. This attack is of particular concern, because the EVDS approach we present uses the transformation capability of virtual directories to encrypt and decrypt sensitive data. Unfortunately, unless a virtual directory server is properly protected against this attack, EVDS is vulnerable. Solutions to this attack are detailed in [Claycomb & Shin 2009]. Fortunately, our PVDS approach is not susceptible to this attack, because any information that passes through a virtual directory server in this case will already be encrypted.

5 Conclusion

We have demonstrated two methods for controlling sensitive user information in directory services. The first, EVDS, is a centralized approach which adds components to standard virtual directory services, handling functionality such as data transformation, encryption/decryption, and delegation management at the virtual directory level. In this approach, users protect encryption keys and authentication is based on knowledge of an original user password. The second approach is a distributed one, PVDS, which moves the protection components to a local service on the client machine. In this approach, users interface with existing key management infrastructure to handle delegation of access to sensitive data. We have described implementation details and results, and discussed how these solutions are resistant to insider attacks on sensitive directory services information. Additionally, we showed how our solutions prevent attacks on directory services and virtual directory services. Future work will include a more comprehensive implementation and user testing scenario. Also, an analysis of attacks against key management services would be helpful in preventing data loss from the solutions we propose. Finally, increasing performance times should be addressed. We believe that protection of sensitive directory information is a critical task facing enterprise system administrators, and we hope that our solution provides a solid step forward in the efforts to secure this data.

Acknowledgements

This work was partially supported at the Secure Computing Laboratory at New Mexico Tech by the grant from the National Science Foundation (NSF-IIS-0916875).

References

[Chadwick 2004] Chadwick, DW.: "Threat Modelling for Active Directory."; Eighth IFIP TC-6 TC-11 Conference on Communications and Multimedia Security (CMS 2004), (September 2004), 203–212.

[Claycomb & Shin 2009] Claycomb, William, & Shin, Dongwan: "Threat Modeling for Virtual Directory Services."; Proceedings of the 43rd IEEE International Carnahan Conference on Security Technology, (October 2009).

[Claycomb et al. 2007] Claycomb, William, Shin, Dongwan, & Hareland, Della: "Towards Privacy in Enterprise Directory Services: A User-Centric Approach to Attribute Management."; Proceedings of the 41th IEEE International Carnahan Conference on Security Technology, (October 2007).

[Jakobsson 2005] Jakobsson, Markus: "Modeling and Preventing Phishing Attacks."; Phishing Panel at Financial Cryptography, (February 2005).

[Keeney et al. 2005] Keeney, Michelle, Capelli, Dawn, Kowalski, Eileen, Moore, Andrew, Shimeall, Timothy, & Rogers, Stephanie: "Insider Threat Study: Computer System Sabotage in Critical Infrastructure Sectors"; Tech. rept. U.S. Secret Service and CERT/SEI, (May 2005)

[Kowalski et al. 2008] Kowalski, Eileen, Cappelli, Dawn, Conway, Tara, Willke, Bradford, Keverline, Susan, Moore, Andrew, & Williams, Megan: "Insider Threat Study: Illicit Cyber Activity in the Government Sector." Tech. rept. U.S. Secret Service and CERT, (January 2008)

[Microsoft Corporation 2007] Microsoft Corporation: How to mark an attribute as confidential in Windows Server 2003 Service Pack 1; `http://support.microsoft.com/kb/922836`

[Microsoft Corporation 2009a] Microsoft Corporation: "Windows Server 2008 Active Directory"; `http://www.microsoft.com/windowsserver2008/en/us/active-directory.aspx`

[Microsoft Corporation 2009b] Microsoft Corporation: "Windows Server 2003 Active Directory Application Mode"; `http://www.microsoft.com/windowsserver2003/adam/default.mspx`

[Radiant Logic, Inc. 2008] Radiant Logic, Inc.: "Using Virtualization to Leverage your Investment in Active Directory"; Tech. rept. Radiant Logic, Inc., http://www.radiantlogic.com/main/pdf/white_paper_activedirectory.pdf

[Radiant Logic, Inc. 2009] Radiant Logic, Inc.: "RadiantOne VDS"; http://www.radiantlogic.com/main/

[Red Hat, Inc. 2005] Red Hat, Inc.: "Fedora Directory Server"; http://directory.fedoraproject.org/

[Shaw et al. 1998] Shaw, Eric, Ruby, Keving, & Post, Jerrold: "The Insider Threat to Information Systems."; Security Awareness Bulletin, **2**(98).