GIAC
CERTIFICATIONS

Global Information Assurance Certification Paper

Copyright SANS Institute
Author Retains Full Rights

This paper is taken from the GIAC directory of certified professionals. Reposting is not permited without express written permission.

Interested in learning more?

Check out the list of upcoming events offering
"Security Essentials: Network, Endpoint, and Cloud (Security 401)"
at http://www.giac.org/registration/gsec

© SANS Institute 2005, Author retains full rights.

© SANS Institute 2005 Author retains full rights.

Single-signon with Open Directory
from GNU/Linux

Chad C. Walstrom

20 Jan 2005

Assignment: GSEC Certification Practical Option 1
Version: 1.4c

1 Abstract

This paper examines methods for sharing account and configuration information between networked computers, focusing on the emerging secure standards of Kerberos, LDAP, SSL/TLS, and SASL. It centers on the Apple Open Directory as a platform to provide these services and demonstrates how to inter-operate with a GNU/Linux workstation.

© SANS Institute 2005, Author retains full rights.

© SANS Institute 2005 Author retains full rights.

Contents

© SANS Institute 2005, Author retains full rights.

© SANS Institute 2005 Author retains full rights.

2 Introduction

In today's corporate and educational network environments, dozens of services are deployed over multiple platforms, such as email, file sharing, and website access. Each of these services require account authentication and management. If they are managed separately, as they often are, the time investment and staffing requirements to keep everything updated and synchronized quickly becomes a burden.

This paper will explore some of the possible methods of sharing information between network computers, and focus on emerging standards that incorporate layers of security for authentication, authorization, and access control. Apple's Open Directory incorporates multiple Free and Open Source Software (FOSS) technologies to create a secure network environment over which to share this information: Kerberos and Simple Authentication and Security Layer (SASL) are used for secure authentication, Lightweight Directory Access Protocol version 3 (LDAPv3) is used for information sharing and access control, and Secure Sockets Layer (SSL) and Transport Layer Security (TLS) is used for encrypting protocols. Together, they provide an integrated environment, where multiple computers have the same information, and access to services can be authenticated through one password, a single-signon environment. Apple's Open Directory server simplifies greatly the difficulties associated with implementing these services.

3 Methods of Sharing Information

Managing accounts on a single computer is a relatively simple activity, whether that computer is a PC compatible workstation running some form of Windows, Linux, or BSD variant, an Apple PowerPC running some version of Macintosh OS, or a commercial UNIX server. Each operating system comes with management utilities that make it easy to add, delete, or edit accounts. For the end-user, updating passwords is also a relatively pain-free experience.

When a second or third computer is added to the equation, this task becomes more difficult. Without a way to share account information between computers, you must maintain multiple, separate sets of data. As the number of computers increases on the network, the problem is exacerbated. Humble systems administrators quickly become overwhelmed, and end-users struggle to keep all of their passwords the same so they don't have to remember more than one.

The security implications of this type of behavior are readily apparent. When the number of protocols increases over which a password is sent, the threat of it being captured or "sniffed" is greater. Once a "cracker"[1], a person who breaks into computers or network services -- analogous to safe cracker, has access to a local account on one machine, other machines are soon to follow.

[1] *The Free On-line Dictionary of Computing.* Sep 2003. ([FOLDOC])

© SANS Institute 2005 Author retains full rights.

© SANS Institute 2005, Author retains full rights.

When employee staffing changes occur, locking down access becomes problematic. The time latency between when the employee has left or was fired and the time when all of the accounts have been locked down gives that person a window of opportunity to cause damage. From a security and management perspective, maintaining separate accounts on separate hosts and services just does not scale well.

The rest of this section covers various methods used to share information amongst network computers. Collectively, these technologies are often referred to as Directory Services.

3.1 Copying Files

Among Linux and UNIX systems, a simple method to solve this problem is to push or pull copies of the system files from a central location. Evi Nemeth, et al., describe multiple UNIX utilities to accomplish this in the UNIX System Administration Handbook ([NEMETH]. 513-521). For example, there are four files that contain user and group account information: `/etc/passwd`, `/etc/shadow`, `/etc/group`, and `/etc/gshadow`. One machine acts as the *master* and pushes copies to the *clients* using a network transfer application such as `rcp` the remote copy program, or it's more secure replacement `scp` ([OPENSSH]). More powerful programs `rdist` or `rsync` running over `ssh` can be used to recursively and selectively copy updated files, preserving ownership, permissions, and modification times.

This scheme does work fairly well for a small number of computers, so systems administrators are tempted to push additional configuration files out in the same manner. automount configuration files describe where to automatically mount remote filesystems. Hostname resolution files, such as `/etc/hosts` and `/etc/networks` works well for smaller environments and saves the network administrator the trouble of having to install and maintain a DNS server. Host access files such as `/etc/hosts.allow` and `/etc/hosts.deny` describe which network hosts are allowed access to which services on a given host machine. Administrators can manage these files on one computer and then push them out when changes occur.

Although this method works, it has a few shortcomings. An obvious inconvenience and security risk is the latency experienced between account information changes, i.e. password updates. If an intruder has compromised a password to an account, the window of opportunity for the intruder is equal to the latency between password updates.

Another problem arises when the system administrator of a host needs to have separate accounts apart from those that are pushed out globally. By overwriting the `/etc/passwd` and `/etc/shadow` files, it is impossible for a local systems administrator to selectively update accounts.

In addition to the vulnerability of increasing latency between updates, an interrupted connection during transfer, either accidental or forced, may leave partial copies of the database in place. This could open possible security holes or, in the least, making it inconvenient for the systems administrator to fix the problem. Essentially, this solution

© SANS Institute 2005, Author retains full rights.

© SANS Institute 2005 Author retains full rights.

has scalability problems not easily solved by simple applications.

3.2 Sun Yellow Pages (NIS)

Sun Microsystems developed one answer to the problem of information sharing across the network called SUN Yellow Pages, now known as the Network Information System (NIS) ([NEMETH]. 525). NIS is a Remote Procedure Call (RPC) based system that shares records from data files over the network using a broadcast paradigm. As client computers need information about accounts or host names, they broadcast the request for the individual record on the subnet and wait for an answer from a master or slave *domain server*. If the master server finds the requested record, it broadcast the result over the network where it is parsed by the waiting client.

The master domain server maintains the authoritative copy of the database files, which are referred to as *maps*. As changes are made, the master server must be told to push out copies of the map files to the slave servers. The master is able to host many different maps, essentially anything that is originally in flat file format.

This request and respond method greatly simplified distribution of information, but it also had some shortcomings. Every lookup request by the client and every response by the server is sent over the broadcast address of a subnet, resulting in increased network traffic. One way to reduce the traffic is to employ the some version of a lookup caching daemon. However, such setups systems are back to experiencing a possible latency based on the timeout values on the data being cached.

NIS, as a protocol, is not encrypted. This may be appropriate for a private Local Area Network (LAN), but the Internet is a very dangerous place to be passing unencrypted information back and forth. Anyone can sniff the broadcast traffic on a NIS subnet, including requests for sensitive information such as encrypted passwords. Nemeth, et al. cautions, "If you are concerned about security, you should not use NIS." ([NEMETH]. 525)

3.3 NetInfo

Apple Software, Inc. tried to solve the sharing information problem with their own directory service platform called NetInfo in the earlier versions of Mac OS X and NeXTSTEP. (Stone, Macintosh, and Toporek. 351) In more recent versions of Mac OS X, use of NetInfo is being phased out in favor of LDAP. It is mentioned here because some settings for the Macintosh host may need to be set using NetInfo rather than using UNIX flat files or LDAP. This customization will not be covered by this article.

3.4 Lightweight Directory Access Protocol (LDAP)

The Lightweight Directory Access Protocol (LDAP) is client-server network protocol structured around a hierarchical data tree, a directory. This style of storing and referencing

© SANS Institute 2005 Author retains full rights.

information is highly flexible and customizable. It allows administrators to arrange information in a way that reflects the organizational entities of the company or institution with much greater ease than could be accomplished with flat files. For example, a company may want to organize its user accounts based on which department the employee belongs to, and LDAP will allow the administrator to do so.

In addition to storing account information, LDAP can also store information on NFS automount maps, host names, network names, netgroups (a concept started by NIS), computers, and Kerberos certificates, which will be covered in a later section.

LDAP was designed as a TCP/IP gateway protocol to the more substantial X.500 directory service. ([CARTER]. 6) Although X.500 is still in use today in some academic and corporate settings, its was originally designed upon a separate network infrastructure than ones based on the today's more popular TCP/IP protocol stack. When trying to keep operating costs low, unifying on a single network infrastructure is an important feature to look for in a service protocol. The current version of the LDAP protocol, LDAPv3, has proven itself remarkably flexible and well suited for today's TCP/IP based networks.

Another important feature of LDAPv3 is its strong model of access control. Account holders can be delegated finite responsibilities to change or update information within the hierarchal tree. Referring back to the example above, this would allow the directory administrator to delegate responsibility for maintaining department or entity-specific accounts to other employees. Although this level of customization and data organization is possible, it will not be focused upon in this article.

Like NIS, LDAPv3 servers can be queried using filters to return only the data you're interested in, and because LDAPv3 is read-optimized, it is far quicker. LDAPv3 has been adapted to use three network protocol enhancements to help secure communications between LDAP clients and servers: Secure Sockets Layer([OPENSSL]), SSL, Transport Layer Security([OPENSSL]), TLS, and Simple Authentication and Security Layer ([CMUSASL]), SASL. These enhancements paired with strong "Defense in Depth" policies help ensure the safety and integrity of the session over the insecure Internet.

A popular open source implementation of LDAPv3 is OpenLDAP([OPENLDAP]). OpenLDAP's version 1.0 release was on August 26th, 1998. Since then, it has had five major releases and numerous patch releases, implementing the LDAPv3 protocol in version 2.0 in August 2000. The current version is 2.2.20, released on January 3rd, 2005.

The popularity of the LDAPv3 protocol and OpenLDAP amongst the Free Software and Open Software communities has spurred the development of a plethora of management utilities, command-line, graphical, and web-based. Open source server applications, such as Postfix, an email server, Apache, a popular web server, and Samba, an open source Windows file sharing server all have been adapted to interface with LDAP for information.

Microsoft recognized the potential of inter-operability and based Active Directory off of the LDAP protocol. In turn, Apple adopted OpenLDAP as well for Open Directory. In the interest of inter-operability, Microsoft and Apple have both adopted the use of LDAP directory services to build their information infrastructure, allowing them to speak to the UNIX platforms that have long since seen the advantages of LDAP.

© SANS Institute 2005 Author retains full rights.

4 Authentication and Encryption

The previous section covered various Directory Services technologies used to share information between networked computers. Some of these services are quite insecure, while others lend themselves to be easily encrypted and protected over the network. This section will cover a few such technologies that are emerging as standards amongst the major operating systems.

4.1 Kerberos

In the early 1980's, a research group at the Massachusetts Institute of Technology (MIT) identified a growing need for secure authentication methods over the popular but insecure Internet. ([GARMAN]. 3) The Kerberos[2] protocol addressed these needs by ensuring that passwords are never sent "in the clear" or unencrypted over the network.

To accomplish this, Kerberos employs a centralized server known as a Key Distribution Center, or KDC. The KDC distributes timestamp-based cryptographic keys to both clients and servers. This centralized nature ensures that both the client and the server are positively identified and authenticated, a method called "mutual authentication". The server can verify that the client requesting access to the service is legitimate, and the client can verify that the server is not *spoofed*, or faked.

For a client to establish a connection to an application service, such as IMAP or Telnet, it must authenticate itself through the KDC. If the authentication is successful, the KDC issues the client a time-based session key and a ticket with which to request access to the service, called a Ticket Granting Ticket (TGT). At this point, the client still has not made its intention known to the KDC as to which service it will connect to.

This is where the notion of single-signon is rooted. Once a client session is established to the KDC, the client can request access to any application service registered with the KDC using its TGT. The only time the client needs to re-authenticate is when the session times out.

The next step in the communication handshaking is for the client to request access to a specific service using its TGT. If the session is still live and the TGT is verified, the KDC will offer a *service ticket* to the client. Garman notes in *Kerberos The Definitive Guide* that the KDC "will happily issue a ticket for any service that it knows about."[3] This is an important piece of information to keep in mind. The KDC does not track authorization; rather, it guarantees authentication. It is up to the service provider, the IMAP or Telnet server, to determine who is allowed access.

That is the last step in the communication between client and the application service provider. The client presents the *service ticket* to the server, which verifies the validity of the ticket and offers its own authentication information to the client. In turn, the client

[2]The greek word Kerberos is a reference to a beast in Greek mythology, commonly referred to as Cerberus in English texts, that protects the Gate to the land of the dead, Hades. ([GARMAN], 2)

[3][GARMAN]. 34.

© SANS Institute 2005 Author retains full rights.

verifies the server information with that which it received from the KDC. At this point, the service protocol begins.[4]

4.2 Encrypting Network Sessions

Although Kerberos makes large steps toward ensuring secure authentication through the centralized mutual authentication model, by itself it only covers part of the problem. The network protocols themselves still need to be protected from snooping.

One of the more popular and effective means of securing a protocol is to use public key cryptography, a form of secure key exchange, through an implementation called Secure Sockets Layer (SSL). The complete history of public key cryptography is a fascinating subject that is out of the scope of this document. The strength behind public key cryptography lies in its use of an asymmetric key. Two different groups of people were given credit for the discovery of an asymmetric key, in the mid 1970's. Whitfield Diffie is credited with discovering the concept in 1975, but was unable to be the first to find implement a working algorithm. In 1977, the MIT research group of Ron Rivist, Adi Shamir, and Leonard Adelman, discover the algorithm, which is known as RSH encryption. ([SINGH]. 268, 272)

An asymmetric key consists of two different keys, a private key and a public key. Either key can be used to encrypted a message, but its pair is the only one that can decrypt it. If a message is encrypted with a public key, only the private key can decrypt it. The same is true in the opposite direction. The asymentric key is implemented through a mathematical formula that uses two, very large prime numbers. Although the mathematic formula and one of the prime numbers is known to the public, it is too computationally expensive and time consuming to attempt guessing the private prime number.

SSL([SSL3]) employs this technique to initiate the encryption of a network protocol. Encrypting and decrypting an asymmetric key is computationally expensive, but it works well for encrypting a key for the much faster symmetric key ciphers, such as blowfish, 3DES, and AES. Through SSL, the client and server agree upon a key and cipher to use to encrypt the rest of the session. A service that is SSL enabled generally starts the SSL Handshake upon connection, therefore requiring a separate port than the service would otherwise be found at. For example, an unencrypted web server generally uses TCP port 80, whereas the SSL-encrypted service runs at TCP port 443.

Transport Layer Security (TLS) is an Internet Engineering Task Force ([IETF]) implementation of the SSL protocol, defined in RFC 2246.([RFC2246]) The main functional difference between SSL and TLS is that TLS can be used as an extension to an existing protocol, allowing the service to use the same port as the unencrypted version. For example, a TLS enabled SMTP server would only have to listen to TCP port 25 for both unencrypted and encrypted sessions.

[4]This process is also described in brief in the *Open Directory Administrators Guide*. [ODADMIN] 3:8

© SANS Institute 2005 Author retains full rights.

4.3 Simple Authentication and Security Layer (SASL)

SASL was first implemented at Carnegie Mellon University as a way to add authentication support to connection based protocols such as HTTP and SMTP.([CMUSASL]) Like TLS, the Simple Authentication and Security Layer ([RFC2222]), SASL, is a network protocol extension mechanism for securely exchanging passwords, it is inserted between the protocol initiation and the connection. One unique and flexible aspect of SASL is its ability to specify different authentication mechanisms, including Kerberos, Digest-MD5, CRYPT-MD5, LANMAN, and NTLM, the latter two used for older Windows Network authentication.([IANASASL])

Apple takes advantage of this flexibility by employing the CMU SASL server as a basis for their Password Server. Through the account management interface and password changing interfaces, passwords for protocols that cannot be authenticated through Kerberos can be syncronized.

5 Implementation

Now that a base understanding of the technologies used has been laid, we can talk how to implement single-signon from a Linux or UNIX operating system to a Open Directory Server.

5.1 Open Directory Setup

Setting up a Macintosh OS X Server Open Directory Server is surprisingly simple compared to the instructions for installing your own Kerberos server from scratch, for example. You do not have to download or compile packages or manually edit plain-text files. Apple has worked hard to present its administrators with a simple interface for installation and configuration. ([ODBRIEF])

When you first install the OS X Server software, you are prompted by the installation process whether or not you would like to install the Open Directory Server. You are prompted for the Kerberos realm, which by default is simply a capitalized version of your existing fully qualified domain name.

The OS X software configures itself based on the DNS name you've given the host. You *must* install a legal DNS record for the IP address you plan on assigning the server, one that is resolved by a DNS server. This can be a private DNS name, but it must be assigned and accessible for lookup on the network on which you deploy the server.

5.2 Server Security Considerations

Although its default configuration upon installation for the server is not necessarily vulnerable, it is not very secure either. The Macintosh OS X Server is a very versatile machine and comes with many different network software packages. It is entirely possible to run

© SANS Institute 2005 Author retains full rights.

© SANS Institute 2005, Author retains full rights.

a small private network with just one server hosting all of the necessary daemons: ISC Bind and DHCP daemons, postfix email server, Apache web server, Apple File Sharing, Samba File Sharing, etc. Apple has made this easy to do, with graphical applications to help configure the various servers.

Despite this ease of use, the more services you make available on a single server, the more vectors for attack a cracker or disgruntled employee may have to break in to the system. If you plan on running an Open Directory server and a Password Server, then security must be a priority. The only services you should be running on the machine is the Open Directory server and the Password Server. Use the built-in firewall to control access to these services to the hosts that need it and no one else. Resist the temptation to enable services that make life easier as a system administrator, such as the Remote Desktop Management or Remote Login (SSH).

Princeton University ([PU]) has a good summary and reference document ([OSXSEC]) concerning how to lock down a Macintosh OS X server. Without going into detail, additional steps you should take include the following:

- Formulate a backup and recovery solution
- Formulate a software update plan
- Implement a physical security plan
- Set an Open Firmware password

The Macintosh XServer, Apple's 1U rackmount server, has a simple keyboard and input lock that doesn't offer any real physical security. It simply protects you from the inadvertent keyboard mashing or mouse clicks. Do not rely upon this feature as a security enhancement.

5.3 Setting up SSL for LDAP on Mac OS X 10.3.x

One step that will need to be taken before the LDAP server is exposed to the world is to set up an SSL certificate and edit `slapd_macosxserver.conf`, the prescribed configuration file for LDAP customizations on Macintosh OS X 10.3.x servers. A self-signed certificate will be sufficient, as long as you remember to make the public certificate available for download. The clients connecting to the server will need to retain a local copy for validation.

The Macintosh OS X 10.3.x operating system includes a copy of the OpenSSL command-line utilities with which one can generate the SSL key and certificates:

```
# Make sure all files generated can only be read by you
shell$ umask 077

# Generate the key (unencrypted)
shell$ openssl genrsa -out slapd-key.pem 1024
```

© SANS Institute 2005 Author retains full rights.

The next step is to generate the Certificate Signing Request (CSR). The CSR is submitted to a Certificate Authority (CA), who creates the actual certificate:

```
shell$ openssl req -new -key slapd-key.pem -out slapd-csr.pem
```

The `slapd-csr.pem` file can now be sent to a Certificate Authority (CA), such as Verisign or Thawte, or it can be self-signed. It is perfectly acceptable to do both, send the CSR to a CA for an official certificate and use a self-signed one until the official one arrives:

```
# Generate the self-signed certificate
shell$ openssl req -x509 -key slapd-key.pem -in slapd-csr.pem \
   -out slapd-crt.pem
```

After the certificate has been generated, it and the key will need to be placed where slapd can find it. A good candidate is in the `/etc/openldap` directory. Remember, the permissions to the key file must be strictly for root only:

```
# Install copies to /etc/openldap
shell$ sudo install -o root -g wheel -m 600 \
   slapd-key.pem /etc/openldap
shell$ sudo install -o root -g wheel -m 644 \
   slapd-csr.pem /etc/openldap
shell$ sudo install -o root -g wheel -m 644 \
   slapd-crt.pem /etc/openldap

# Remove local copies
shell$ rm *.pem
```

It is possible to concatenate the certificate and key files together, but it is not necessary. The certificate still needs to be distributed to client machines to validated when they make connections to the OpenLDAP server. Additionally, OpenLDAP provides directives for specifying the certificate, key, and CA certificate. If you are using a self-signed certificate, both the `TLSCertificateFile` and `TLSCACertificateFile` will refer to the same certificate:

```
# /etc/openldap/slapd_macosxserver.conf ...
# TLS Certificate Settings
TLSCertificateFile      /etc/openldap/slapd-crt.pem
TLSCertificateKeyFile   /etc/openldap/slapd-key.pem
TLSCACertificateFile    /etc/openldap/slapd-crt.pem
```

The `/etc/openldap/slapd.conf` file also contains a sample of additional security restrictions that can be placed on the LDAP server. To disable the exchange of clear text passwords, for example, add `disallow bind_simple_unprotected` to the `/etc/openldap/slapd_macosxserv`

© SANS Institute 2005 Author retains full rights.

file. To require certain levels of encryption for updates or binding, use some variant of the `security` directive.[5]

Once the configuration files have been edited, the LDAP server will need to be started with the appropriate command-line arguments to enable SSL. OS 10.3.x does have a graphical user interface (GUI) to enable SSL in the `ServerManager` application.

Essentially the GUI edits the `/etc/hostconfig` file, changing the `LDAPSSL` environment variable to "-YES-". This file is sourced by the `rc.common` file, which in turn is sourced by the startup script for LDAP, located at `/System/Library/StartupItems/LDAP/LDAP`. Similar to the scripts found in the `/etc/init.d/` directory on UNIX-like systems, the scripts found in subdirectories of `/System/Library/StartupItems` can be used to manually start and stop services at the command-line. For example, the LDAP server can be restarted with the sudo[6] command like so:

```
shell$ sudo /System/Library/StartupItems/LDAP/LDAP RestartService
```

The LDAP server will be restarted with SSL enabled, and it will listen on all network interfaces for both the unencrypted TCP port 389 and the SSL-enabled TCP port 636. The next step to locking things down is to add firewall rules with the `ServerManger` to restrict access to exposed interfaces. For example, you may want to block access to 389 from public IP addresses, but allow access to 636.

5.4 Kerberos

The prerequisite to installing the pam_krb5 module is to have a working Kerberos 5 enviroment installed. Most Linux and BSD distributions have their own flavor of package from which they distribute binary modules. Consult your distribution's website or package utilities to find out whether or not Kerberos 5 is available for your architecture and OS environment. The source code for Kerberos 5 can be obtained from the MIT Kerberos Project Home Page([MITKRB]). Also available on the site is detailed instructions on building, installing and configuring the software.

Since the scope of this article is to configure a Linux or BSD machine to act as a client to an Open Directory server, only a simple configuration is needed. The Kerberos session can be configured in the system-wide configuration file `/etc/krb5.conf`.

Upon setup of the Open Directory server, the realm was assigned as the fully qualified domain name for the server. This realm is specified in the configuration file in two places, the `[libdefaults]` section and the `[realms]` section:

```
# /etc/krb5.conf
```

[5]Consult the slapd.conf manpage or the *LDAP System Administration* book. [CARTER]

[6]sudo is a secure replacement for the `su` -- switch-user -- command. sudo switchs the effective uid and gid for a given command without needing to distribute the root password, rather it prompts for the user's own password and consults `/etc/sudoers` to determine if the user has permissions to execute the command.

© SANS Institute 2005 Author retains full rights.

```
[libdefaults]
   default_realm = HOST.DOMAIN.TLD
# ... SNIP ...
[realms]
HOST.DOMAIN.TLD = {
    kdc = HOST.DOMAIN.TLD
   admin_server = HOST.DOMAIN.TLD
}
# ... SNIP ...
```

5.5 Linux Pluggable Authentication Modules

The Linux Pluggable Authentication Module([LNXPAM]), PAM, is a stackable Application Programming Interface (API). It provides a library through which applications can build authentication, authorization, account management, and session management. The stackable nature of PAM enables the use of multiple authentication and management mechanisms to be used for a single application.

Apple doesn't use PAM extensively, but Linux and BSD UNIXs do. It is the method through which single-signon is enabled for Linux and BSD machines against the Open Directory server. All Linux distributions and most BSD distributions come with basic PAM installed, but some additional modules may be required.

One very important note to make about PAM authentication is that it can be quite easy to expose an otherwise secure authentication method over insecure channels. For example, if you were to use the pam_krb5 module to authenticate users for FTP server, you would lose all of the benefits of Kerberos because the FTP protocol is unencrypted. The user's password would be transmitted in plain text, which can easily be sniffed off the network.

PAM's flexibility is in the number of different modules you can use for authentication. Using the above example of hosting an FTP server, one could use a PAM module that consults a separate file to match users to plain-text or hashed passwords. However, this type of setup is far outside the scope of this document.

The bottom line with PAM security is to match the relative security level of the authentication method to the security level of the protocol. pam_krb5 works wonderfully for login consoles or X Window System logins for local workstations[7], because access to consoles or login windows is limited to a physical machine and authentication does not pass through a network interface. pam_krb5 would not be acceptable for authenticating Telnet, FTP, or XDMCP sessions.

[7]X11 Display Managers such as XDM can be configured to allow users to log in remotely. If you plan on allowing this, do NOT enable pam_krb5 authentication for that service. X11 is not an encrypted protocol.

© SANS Institute 2005, Author retains full rights.

© SANS Institute 2005 Author retains full rights.

5.5.1 Installing pam_krb5

pam_krb5 is a PAM module used for Kerberos authentication. The source code can be obtained from its project page [PAMKRB] at the Source Forge[8] site. pam_krb5 comes with a simple Makefile, which you can edit to conform to your operating environment. This PAM module has been available for some time, so it is likely that it is packaged for most PAM-enabled Linux and BSD systems. For example, on the Debian GNU/Linux system, it is packaged as `libpam-krb5`.

The documentation for this module is somewhat developer-specific, but referencing the Linux PAM documentation ([LNXPAM]) will help make the setup details clearer.

5.5.2 Configuring the PAM packages for pam_krb5

Packages that use PAM can be configured in one of two ways. Either configuration directives are placed in the `/etc/pam.conf` file, or they're placed in individual files in `/etc/pam.d` named after the service. Newer distributions are using the `/etc/pam.d` directory as a standard. To add pam_krb5[9] for the login service, the `/etc/pam.d/login` file would look something like the following:

```
# /etc/pam.d/login
# module-type  control-flag  module.so  options
# login service
auth     requisite   pam_securetty.so
auth     requisite   pam_nologin.so
auth     requisite   pam_env.so

# Kerberos authentication
auth     sufficient  pam_krb5.so
#@include common-auth
auth     required    pam_unix.so try_first_pass

# Account settings
account      sufficient      pam_krb5.so
@include common-account

# Session
session  sufficient  pam_krb5.so
@include common-session
session  optional    pam_lastlog.so
session  optional    pam_motd.so
session  optional    pam_mail.so standard noenv
```

[8]http://www.sourceforge.net

© SANS Institute 2005, Author retains full rights.

© SANS Institute 2005 Author retains full rights.

```
# Password management
password sufficient  pam_krb5.so
#@include common-password
password required    pam_unix.so use_first_pass
```

A relatively new feature to PAM is to allow files to be included in /etc/pam.d/SERVICE files by using the @include common-auth. It was commented out above, because the pam_unix.so needs to have an additional option that would not work well in the "common" case. It needs to try the first password attempted by the module above it in the stack, the pam_krb5.so module. If that password fails, it prompts the user for a new one.

Special options are used for pam_unix.so again in the case of the password module-type to allow it to use the password token generated by the pam_krb5 module. If the login package determines that a user needs to change his or her password, it uses the settings from the password module-type to determine where and how to do it.

On a related note, configuring the passwd command-line utility in /etc/pam.d/passwd requires only the last two password module-type lines as seen in /etc/pam.d/login.

The pam_krb5 manpage mentions the configuration change requirement to the login package configuration file /etc/login.defs. The setting CLOSE_SESSIONS must be enabled with the value yes to enable the session credentials to be destroyed at the end of a session:

```
# Enable pam_close_session() calling.
CLOSE_SESSIONS yes
```

If you have problems with pam_krb5, consider adding the debug option to each of the PAM directives. The pam_krb5 manpage has more details on this and other available module options.

There are a few other packages you might want to configure to use pam_krb5, such as the password changing program passwd and an X11 Display Manager like Gnome Display Manager(GDM). The PAM setup steps are identical for /etc/pam.d/gdm as they were for /etc/pam.d/login: add the appropriate entries for the pam_krb5.so module above the pam_unix.so module.

As mentioned above, if you plan on using an X11 Display Manager, shut off remote logins over the XDMCP protocol. The GDM configuration file gdm.conf has XDMCP disabled by default:

```
# /etc/gdm/gdm.conf
# ... SNIP ...
[xdmcp]
# Distributions: Ship with this off.  It is never a safe thing to
```

[9]The pam_krb5 does install with a manpage detailing PAM configuration for the pam_krb5 module-types and options.

© SANS Institute 2005, Author retains full rights.

© SANS Institute 2005 Author retains full rights.

```
# leave out on the net.  Setting up /etc/hosts.allow and
# /etc/hosts.deny to only allow local access is another
# alternative but not the safest.  Firewalling port 177 is the
# safest if you wish to have xdmcp on.  Read the manual for more
# notes on the security of XDMCP.
Enable=false
```

5.6 Name Service Switch

Once PAM has been set up for locally secure applications on the Linux or BSD workstation, the system needs to be configured to grab information from the Open Directory LDAP server, such as user and group ids and names.

The Name Service Switch (NSS), first seen in the C library for Solaris 2 and reimplemented in the GNU libc library [GNULIBC], is a modular database through which applications can request information with standard system calls and receive information from different data sources. For example, a hostname can be requested through the `gethostbyname()` system call. That hostname can be resolved by the /etc/hosts file, through a DNS request, an NIS lookup, or LDAP lookup.

The following section, nss_ldap, will examine how to install and configure nss_ldap module to provide LDAP database lookups to NSS.

5.6.1 nss_ldap

nss_ldap, written by PADL Software [NSSLDAP], provides LDAP lookups capability module for NSS. The source code can be obtained at the nss_ldap Project Page cited above. Like the other software used in this setup, much of it is already packaged by Linux or BSD distributions. In the case of the Debian GNU/Linux distribution, it is packaged as `libnss-ldap`.

If you must compile the package yourself, detailed instructions are found in the `INSTALL` file of the source code. One customization that might be useful is to specify the path to the nss_ldap configuration file using the `--with-ldap-conf-file` configuration option. By default, this file is called `ldap.conf` which conflicts with the configuration file for OpenLDAP's command-line tools. Changing the nss_ldap configuration file name to `nss_ldap.conf` or `libnss-ldap.conf`, as it is in Debian GNU/Linux, removes this potential confusion.

Once the software has been installed, there are two files that must be updated to configure the package for use with the Macintosh Open Directory: the configuration file for nss_ldap and the NSS configuration file, `/etc/nsswitch.conf`.

The nss_ldap configuration file describes to the module which host to query and where to look for information. Microsoft Open Directory stores account information for users under the Distinguished Name (dn) of `cn=users,dc=HOSTNAME,dc=DOMAIN,dc=TLD`. Likewise, group information is stored under `cn=groups,...`:

© SANS Institute 2005 Author retains full rights.

```
# LDAP Configuration -- CBS going to Mac OS X server
#
uri ldaps://192.168.1.1/
ldap_version 3
base dc=HOSTNAME,dc=DOMAIN,dc=TLD

nss_base_passwd      cn=users,dc=HOSTNAME,dc=DOMAIN,dc=TLD?one
nss_base_shadow      cn=users,dc=HOSTNAME,dc=DOMAIN,dc=TLD?one
nss_base_group       cn=groups,dc=HOSTNAME,dc=DOMAIN,dc=TLD?one
```

The configuration file also specifies whether or not to use TLS and SSL, whether to check the authenticity of the CA signature on the certificate, and if so, where to look for the CA public certificate. For a self-signed certificate, a copy of the public key must be made available in the filesystem to reference as the CA certificate:

```
# OpenLDAP SSL mechanism
# start_tls mechanism uses the normal LDAP port, LDAPS typi-
cally 636
ssl start_tls
ssl on

# OpenLDAP SSL options
# Require and verify server certificate (yes/no)
# Default is "no"
tls_checkpeer no

# CA certificates for server certificate verification
# At least one of these are required if tls_checkpeer is "yes"
#tls_cacertfile /etc/ssl/ca.cert
tls_cacertdir /etc/ldap/certs
```

Next, the `/etc/nsswitch.conf` file must be told to use the LDAP module to look up information. This is done by appending the string `ldap` to each entry type of information that LDAP should be queried:

```
# /etc/nsswitch.conf
#
# Example configuration of GNU Name Service Switch functionality.
# If you have the ‘glibc-doc’ and ‘info’ packages installed, try:
# ‘info libc "Name Service Switch"’ for information about this
# file.

passwd:         files ldap
```

© SANS Institute 2005 Author retains full rights.

```
group:          files ldap
shadow:         files ldap

hosts:          files dns
networks:       files

protocols:      db files
services:       db files
ethers:         db files
rpc:            db files

netgroup:       nis [NOTFOUND=return]
```

6 Conclusions

Once all of these pieces are in place, users should be able to log in to the console or GDM screens of a GNU/Linux or BSD workstation or server using their Open Directory account and password. Setting up NFS or SMB mounted home directories was not discussed in this article, but there are plenty of resources on the web that detail these procedures.

Apple Computing, Inc has made a strategic decision in using multiple Free and Open Source Software technologies to build its Open Directory Server. They leverage the proven success of these technologies to inter-operate with a wider audience. Likewise, Microsoft has also adopted the use of Kerberos and LDAP in its Active Directory, closing the gaps between the major Operating Systems vendors.

Using freely available operating systems and software packages, it is possible to inter-operate with these traditionally commercial operating systems.

7 Appendix: References

This document was written in reStructuredText[10] format and translated to LaTeX with Python Docutils[11]. LaTeX does not have support for underlined text. Book, magazine, database, and website titles have therefore been emphasized with italics in the printed document format. References are listed in the order that they are cited in the document, with uncited, yet useful references at the end.

[10] http://docutils.sourceforge.net/rst.html
[11] http://docutils.sourceforge.net

© SANS Institute 2005 Author retains full rights.

References

[FOLDOC] *Free On-line Dictionary of Computing.* 19 Jan 2005. <http://wombat.doc.ic.ac.uk/foldoc/>

[NEMETH] Nemeth, Evi. et al. *UNIX Sysatem Administration Handbook.* Upper Saddle River: Prentice Hall PTR. 2001.

[OPENSSH] *The OpenSSH Project Page.* 19 Jan 2005. <http://www.openssh.org>

[CARTER] Carter, Gerald. *LDAP Sysem Administration.* Sebastopol: O'Reilly & Associates, Inc., 2003.

[OPENSSL] *The OpenSSL Project Page.* 19 Jan 2005. <http://www.openssl.org>

[CMUSASL] *Carnegie Mellon University SASL Project Page.* 11 Dec 2004. <http://asg.web.cmu.edu/sasl>

[OPENLDAP] *OpenLDAP Project Home Page.* 19 Jan 2005. <http://www.openldap.org>

[GARMAN] Garman, Jason. *Kerberos The Definitive Guide.* Sebastopol: O'Reilly & Associates, Inc., 2003.

[ODADMIN] Apple Computer, Inc. *Mac OS X Server Open Directory Administration.* Apple Computer, Inc. 2003 <http://images.apple.com/server/pdfs/Open_Directory.pdf>

[SINGH] Singh, Simon. *The Code Book.* New York: Anchor Books. 2000.

[SSL3] *SSL 3.0 Specification.* 19 Jan 2005. <http://wp.netscape.com/eng/ssl3>

[IETF] *Internet Engineering Task Force Home Page.* 19 Jan 2005. <http://www.ietf.org>

[RFC2246] Dierks, T. et al. "The TLS Protocol Version 1.0" *IETF RFC 2246.* January 1999. 19 Jan 2005. <ftp://ftp.ietf.org/rfc/rfc2246.txt>

[RFC2222] Meyers, J. "Simple Authentication and Security Layer (SASL)", *IETF RFC 2222.* October 1997. 19 Jan 2005. <ftp://ftp.ietf.org/rfc/rfc2222.txt>

[IANASASL] Internet Assigned Numbers Authority. "IANA Assignments for Simple Authentication and Security Layer (SASL) Mechanisms". 11 Dec 2004. <http://www.iana.org/assignments/sasl-mechanisms>

[PU] *Princeton University Home Page.* 12 Jan 2005. <http://www.princeton.edu/>

© SANS Institute 2005, Author retains full rights.

© SANS Institute 2005 Author retains full rights.

[OSXSEC] Princeton University. *OS X Security.* Nov 2004. 12 Jan 2005. <http://www.princeton.edu/~psg/unix/osx/osxsecurity.html>

[MITKRB] *MIT Kerberos Project Home Page.* 19 Jan 2005. <http://web.mit.edu/kerberos/www>

[LNXPAM] *Linux PAM Project Page.* 19 Jan 2005. <http://www.kernel.org/pub/linux/libs/pam>

[PAMKRB] *pam_krb5 Project Page.* 19 Jan 2005. <http://sourceforge.net/projects/pam-krb5>

[GNULIBC] Free Software Foundation, Inc. *The GNU C Library Reference Manual.* Edition v0.10. 2001. 19 Jan 2005. <http://www.gnu.org/software/libc/manual/html_node>

[NSSLDAP] *PADL Software nss_ldap Project Page.* 19 Jan 2005. <http://www.padl.com/OSS/nss_ldap.html>

[PANTHER] Stone, McIntosh, and Chuck Toporek. *Mac OS X Panther in a Nutshell.* Sebastopol: O'Reilly Media, Inc., 2004, 2003.

[OD] *Apple Open Directory Product Page.* 19 Jan 2005. <http://www.apple.com/ca/server/macosx/open_directory.html>

[ODBRIEF] Apple, Inc. "Open Directory Technology Brief". Apple Computer, Inc. 2003. 19 Jan 2005. <http://images.apple.com/ca/server/pdfs/L31755A_OpenDirect2_TB_final.pdf>

[SUNSAG] Sun Microsystems. *System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP).* Santa Clara: Sun Microsystems, 2004. 19 Jan 2005. <http://docs-pdf.sun.com/817-4843/817-4843.pdf>

[ROME] Rome, Jim. "How To Kerberize Your Site". 19 Jan 2005. <http://www.ornl.gov/~jar/HowToKerb.html>

[PAMLDAP] *PADL Software pam_ldap Project Page.* 19 Jan 2005. <http://www.padl.com/OSS/pam_ldap.html>

[OGA] Oga, Alvin "Autofs Automounter HOWTO". 1998. 19 Jan 2005. <http://www.linux-consulting.com/Amd_AutoFS/autofs.html>

© SANS Institute 2005, Author retains full rights.

© SANS Institute 2005 Author retains full rights.