# Global Information Assurance Certification Paper

# Secure use of LDAP for Naming Services with Solaris

**GSEC Gold Certification**

Author: Raymond Scott
Advisor: Joey Niem

March 15, 2007

## Introduction

This paper will discuss some security considerations when using Lightweight Directory Access Protocol (LDAP) as a naming service for Solaris systems, that is, as a networked storage location for the information usually stored in local files, such as account and group information, automount maps etc. It will specifically discuss changes to the configuration of the Sun ONE Directory Server 5.2 product, and changes to the Solaris client configuration to help avoid some security vulnerabilities. Note that all examples use fake data.

## Overview

LDAP is a protocol for communications between LDAP servers and LDAP clients. LDAP servers store data in hierarchical "directories" (databases) which are accessed by LDAP clients. LDAP is lightweight because it is a smaller and easier to use protocol, derived from the X.500 Directory Access Protocol (DAP), defined in the OSI network protocol stack (Tech-Faq, 2006).

LDAP is frequently being used as a replacement for older naming services like NIS and NIS+ due to some of the limitations of those products (Frisch, 2002). While LDAP can be a good replacement, it is important to understand how it works and configure it to avoid security vulnerabilities and minimize performance problems.

## Preventing Privilege Escalation

Sun ONE Directory Server 5.2 can be configured to serve as a naming service by running the supplied idsconfig script installed in the /usr/lib/ldap directory (Haines and Bialaski, 2004). This script will create the object classes, containers, client profiles, and Access Control Instructions (ACI) needed

to support Solaris Native LDAP. LDAP uses ACIs, to control access to information inside the directory. ACIs allow the administrator to grant access by user and/or group, IP address, domain name, authentication method, time of day, etc. (Faust, 2001)

This is a good starting point, but you may want to consider adding additional ACIs to improve security.

### *Restrict write privileges*

Besides the LDAP directory administrators, there are usually some additional individuals who maintain the data stored within a 'domain' or OU (Organizational Unit). These are usually system administrators. They are responsible for maintaining the data for the servers that they support in their domain. This can be done by creating a static group and giving the members of that group the privileges needed to maintain the data; see the Sun Java System Directory Server 5 2005Q1 Administration Guide for information on how to create groups.

You will need to add ACIs on each container to give these individuals the needed privileges to maintain their data. For one thing, they will need to be able to create and update the uidnumber attribute. To help eliminate the possibility of someone changing their own or someone else's uidnumber to 0 (thereby giving that user root privileges) (Sun Microsystems, 2003), create a 'root' account entry in the ou=people branch that has uidnumber 0. Then, add uidnumber to the uniqueness plug-in for that OU, and add an ACI like the following to prevent anyone from changing root's uidnumber:

```
dn: uid=root,ou=People,ou=eng
aci: (targetattr = "uidNumber") (version 3.0;acl "Deny any modification
to root entry or uidNumber";deny (selfwrite,write,delete,add)(userdn =
"ldap:///anyone");)
```

This will prevent anyone from assigning themselves the uidnumber 0, giving themselves root privileges wherever they can login.

Individual users should not be able to update their own entry's uidnumber. If the directory software is installed with the 'typical' option, the ACI created will give the user who binds to the directory write permission to their own entry (Bialaski, 2000). That ACI should be changed to prevent this:

```
dn: ou=People,ou=eng
changetype: modify
replace: aci
aci: (targetattr != "cn || uid || uidNumber || gidNumber ||
gidNumber || homeDirectory || shadowLastChange || shadowMin ||
shadowWarning || shadowInactive || shadowExpire || shadowFlag ||
memberUid") (version 3.0; acl "Allow self entry modification";
allow (write) userdn = "ldap:///self"; )
```

If users need to be able to change their passwords from the command line, they will need write permission to the userpassword attribute, but if password maintenance is done via a web interface, then a 'webadmin' account can be used to update passwords, and users would not need write privileges to userpassword either.

### *Directory Audit logs*

If you have a large organization, it is likely that you will have a directory support team who administers the actual directory server, acting as the LDAP DBAs. It should be understood that the LDAP directory administrators, the people with the 'cn=Directory Manager' password, must be highly trusted individuals. They, in essence, have write permission to every password and shadow file on every machine that uses LDAP as a naming service. Directory server audit logs should be copied to a central repository to maintain a clear audit trail.

## Preventing password exposure

Solaris systems configured to use LDAP as a naming service should be running the latest LDAP patches. There are numerous changes being made to the LDAP software in Solaris (the Solaris 8 LDAP patch is at version 65 at the time of writing; an average of about 2 patches per month for the last two years).

LDAP Clients should use an LDAP profile that protects the user's password on the network, avoids the overhead of unneeded encryption and prevents the display of password hashs.

### *A Review of Client Versions*

There are two Solaris LDAP naming service clients available: (Haines and Bialaski, 2004)

- Native Solaris (Phase 1)
- Secured LDAP (Phase 2)

#### Native Solaris (Phase 1)

The Native Solaris (Phase 1) profile uses a proxyagent account (Haines and Bialaski, 2004) which introduces some security concerns. Native Solaris profiles are easily identified by the version "1.0" in the list output of the ldapclient command.

The main problem with the Native Solaris profile is that it does not support the use of Transport Layer Security (TLS) for an encrypted session between the client and the directory server, and it uses the proxyagent account, which allows for some undesirable capabilities, which are discussed below.

In order to use the phase 1 type profile, the directory server must have some ACIs that give read permissions to the proxyagent account.

These ACIs give anyone the right to read any data from the directory except for the userpassword attribute; and gives the proxyagent the right to read, search, or compare any attribute including userpassword.

### Secured LDAP (Phase 2)

The Secured LDAP (Phase 2) profile will show version "2.0" in the listing from the ldapclient command. In a secured LDAP profile, the authentication method can be just 'simple', so TLS would not be used, so it functions like the Native Solaris profile.

The below examples will use a phase 2 profile that uses 'simple' with proxyagent (effectively, the same as a phase 1 profile) to show what happens with a Native Solaris (phase 1) profile.

With a credential level of 'proxy' and with TLS not being used, the password hash of the person logging in is passed over the network as is, as seen in this snoop capture from port 389 during an SSH login:

```
LDAP:  ----- Lightweight Directory Access Protocol Header -----
LDAP:    *[LDAPMessage]
LDAP:       [Message ID]
LDAP:      Operation *[APPL 4: Search ResEntry]
LDAP:          [Object Name]
LDAP:              uid=joeuser, ou=People, ou=eng
LDAP:           *[Partial Attributes]
LDAP:            *[Attribute]
LDAP:                [Type]
LDAP:                     uid
LDAP:               *[Vals]
LDAP:                     [Value]
LDAP:                          joeuser
LDAP:             *[Attribute]
LDAP:                [Type]
LDAP:                     userpassword
LDAP:               *[Vals]
LDAP:                     [Value]
LDAP:                          {CRYPT}dMcgsaDYQ6/y.Q
LDAP:
```

The other issue with using the proxyagent account with authentication method set to ‘simple’, is that the proxyagent’s password is sent to the directory server in a bind request in the clear (Sun Microsystems, 2004). Using the profile as discussed above, a snoop capture on port 389 of an SSH login shows the proxyagent password being passed:

```
TCP:  Destination port = 389 (LDAP)
LDAP:  ----- Lightweight Directory Access Protocol Header -----
LDAP:    *[LDAPMessage]
LDAP:       [Message ID]
LDAP:      Operation *[APPL 0: Bind Request]
LDAP:          [Version]
LDAP:          [Object Name]
LDAP:              cn=proxyagent,ou=profile,ou=eng
LDAP:         Authentication: Simple  [0]
LDAP:              pr0xyagen1pw
LDAP:
LDAP:  ----- LDAP:   -----
```

By sniffing the proxyagent account and password off the network, an attacker does not even need an account within the domain; she can just use the proxyagent account with an ldapsearch command to capture all the password hash values for a later run through a password cracking program.

## *A partial solution*

Enabling TLS with this profile will encrypt ALL traffic between the directory and client (Nieminen, 2006). It is important that TLS always be used whether you are using the proxyagent account or not. A drawback to using TLS with proxyagent is that now you have the overhead of encrypting everything, in addition to the extra work generated on the directory since the proxyagent binds multiple times for each query.

Even when using proxyagent with TLS, it still leaves a significant problem. When a client is using a proxyagent

account type profile, it allows any user who can logon to the box to list ALL the accounts in the domain, (not just the ones who can logon to that box), AND their password hash values.

```
/# ldaplist -l passwd | more
dn: uid=joeuser,ou=people,ou=eng
        userPassword: {crypt}WMOcvRAShwv2U
        loginShell: /usr/bin/csh
        gecos: Joe T User,2247788
        gidNumber: 334
        objectClass: top
        objectClass: account
        objectClass: posixAccount
        objectClass: shadowAccount
        uidNumber: 41072
        homeDirectory: /home/joeuser
        cn:
        uid: joeuser

dn: uid=imauser,ou=people,ou=eng
        loginShell: /bin/ksh
        gecos: Ima User,235546
        gidNumber: 300
        objectClass: top
        objectClass: account
        objectClass: posixAccount
        objectClass: shadowAccount
        uidNumber: 12203
        homeDirectory: /home/imauser
        cn:
        uid: imauser
        userPassword: {crypt}Opx5.GGjHYvk
[…]
```

It is possible to prevent the display of the users' passwords by configuring the client system to use pam_ldap in the /etc/pam.conf file and removing the ACI that allows proxyagent to read userpassword (you would first have to update the /etc/pam.conf on all your clients that use a proxyagent type profile) (Sunsolve Doc# 73327, 2005). The profile will still show a credential level of 'proxy' but it is not doing anything useful, only generating unneeded network traffic.

It is better to use a profile without proxyagent at all as discussed below.

## *A better solution - pam_ldap*

Normally, when using the pam_ldap module for authentication, the user password and uid are passed to the directory in the clear, in an attempt to bind. If the bind is successful, login can proceed. Of course, we do not want the password going over the network in the clear, so TLS must also be used. We do not want the overhead of encryption for ALL traffic, so we use a credential level of 'anonymous' for everything else. The resulting profile looks like this:

```
/# ldapclient list
NS_LDAP_FILE_VERSION= 2.0
NS_LDAP_SERVERS= 10.1.1.32, 10.1.1.64
NS_LDAP_SEARCH_BASEDN= ou=eng
NS_LDAP_AUTH= none
NS_LDAP_SEARCH_REF= TRUE
NS_LDAP_SEARCH_SCOPE= one
NS_LDAP_SEARCH_TIME= 30
NS_LDAP_SERVER_PREF= 10.1.1.32, 10.1.1.64
NS_LDAP_PROFILE= starbase
NS_LDAP_CREDENTIAL_LEVEL= anonymous
NS_LDAP_BIND_TIME= 30
NS_LDAP_SERVICE_AUTH_METHOD= pam_ldap:tls:simple
```

Here, the authentication method is 'none' and credential level is 'anonymous' meaning that for most of our LDAP traffic, we are accessing the directory with an anonymous bind.

The service authentication method is 'pam_ldap:tls:simple' meaning that for the pam_ldap service, we will use TLS for encrypted traffic, and simple authentication (uid and password in an attempt to bind). It can be argued that this is still not optimal, since the user's password is still sent over the network, even though it is through an encrypted session. While it is possible to use Kerberos for authentication, this comes with additional overhead and may not perform well under load.

### Compatibility considerations

Most recently developed software seems to be PAM aware and will work correctly with the pam_ldap module. However, you may encounter issues with older versions that do not support PAM. Under Solaris 8, we had to recompile our internally provisioned OpenSSH, sudo, and WU-FTP packages to enable PAM support.

## Limiting access to the Client

Usually you do not want every user in the domain to be able to logon to every client machine; you want some way to specify who can login to a specific client. One way to do that is by using netgroups with 'compat' in the /etc/nsswitch.conf file. Be aware that there is a performance issue with the use of compat mode and LDAP below a certain patch level. Depending on how many netgroups you have, using compat can generate a huge amount of network traffic with searches against the directory, causing slow logins and slow responses to ls –l, ps –ef, and other commands. For Solaris 8, the 108993-63 patch corrects this problem. For Solaris 9 the fix is in 112960-40. We have not seen the fix for Solaris 10 yet.

An alternative to using netgroups is to use the pam_netgroup module, which is available at:

http://opensolaris.org/os/community/security/projects/pam

This module allows you to put the authorized users and or netgroups in /etc/users.allow without incurring the performance hit with using compat (Moffat, 1999).

## Password Aging Problems

While you can enable password aging on the LDAP directory (Sun Microsystems, 2003), users may or may not receive the "password is expiring" message when logging onto a client machine, and the user may still be able to login even after the password expiration date has passed. It all depends on how the client system is configured, what versions of software are running, and how the user is logging in. For example, SSH with public/private key authentication may still allow the user to login even if the password has expired. Therefore, you cannot count on the directory server's password expiration to prevent someone from logging on to a client.

One solution is to implement password aging by creating two password policies on the directory, one where the password expires (for example, called password-expire) and another where the password does not expire (password-noexpire). Sun ONE Directory 5.2 allows a policy to be assigned to each individual entry, so the administrator can choose which accounts will expire and which will not by setting the passwordpolicysubentry attribute to point to the desired policy using the console or via a script. Once the policy for an account has been set, the password must be updated by either changing it, or updating it with the same value, to 'start the clock'. The directory will then add the passwordexpirationtime attribute.

Then, a program can be run on the LDAP master each day that looks for passwords expiring in the near future, say 21, 14, 7, or 1 days. The program can send an email to the account owner telling them to change their password.

If the account owner does not change their password before the expiration date, the same program can lock the account by

changing the user's password to a '*LK*' type value, and setting the loginshell attribute to a value like '/bin/false'. This will prevent that account from being used for logging in. Although, that user's cron jobs will continue to run.

## Profile Management

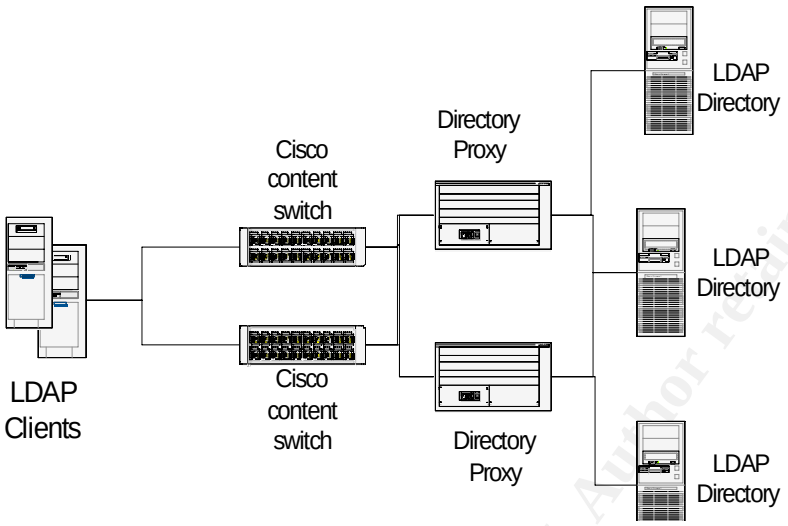
In my organization, we have settled on a standard of creating a separate profile for each host. This gives us very granular control over which LDAP server a particular client is using. It is possible to use one profile for many clients, but this means that a change to that profile will affect all those clients at once.

We have had cases where a developer on a single host creates a program with a bug that generates thousands of LDAP connections a minute, to the point where the directory is unable to respond; basically, a denial of service attack. We can modify that host's profile on the directory (we do not need root access to the client) and have it use a test LDAP server while we determine who or what is causing the problem. At the ldapcache_mgr refresh time, the client will pull down the modified profile and begin using it.

## Architecting for Availability

Losing connectivity to your LDAP directories can be very disruptive. Logins fail, cron jobs will not run, and automounts will not mount. The profile can have multiple LDAP servers listed, but depending on your patch level, the 'failover' may take 10 minutes or more. In addition, as mentioned above, a single client can generate enough traffic to render a directory unable to respond. Should the client then failover to another directory, it would just take that directory down too.

To help make the directory environment more robust, we adopted the architecture proposed as "An Internal High Availability Configuration" in the Sun ONE Directory Proxy Server documentation (Sun Microsystems, 2005), although we did add a second content switch:




The content switches are setup as active/passive. One switch is normally in use, while the other is a failover standby. The Sun ONE Directory Proxy Server has the ability to limit the number of simultaneous connections from a single IP address. This helps to prevent the denial of service type attacks (intentional or by accident). We are currently using a limit of 50 simultaneous connections from a single IP address. The proxy server is also a dynamic load balancer that is LDAP aware. This gives us the ability to take down a back end directory for maintenance with no disruption of service. Having two proxy servers also allows us to take one down for maintenance with no disruptions.

This architecture provides high availability and allows non-disruptive maintenance to be performed easily. In addition,

capacity can be added quickly by adding a new back end directory behind the proxy server.

## Summary

To improve security when using LDAP as a naming service on Solaris, there several things that should be considered:

Sun frequently issues fixes to their LDAP software that includes security fixes. The Solaris 8 LDAP patch is up to version 65 (at the time of writing). It is important to stay current on patches to ensure you have the latest fixes installed.

With default access privileges, it is possible for someone to give themselves or others escalated privileges. Use the uniqueness plug-in and a related ACI to prevent someone from giving themselves root privileges via their LDAP entry.

Remember that those administrators who are given create and write permissions on the directory essentially have write permission to every password and shadow file on every box configured to use LDAP as a naming service for the domains they administer. Maintain a clear audit trail by storing directory audit logs on a central log server.

Do not use the old phase 1 profiles on clients; it passes passwords in the clear over the network. Use the Secure LDAP profile (phase 2) configured to use TLS.

Do not use the proxyagent account with a profile set for a credential level of ‘proxy’. While the network traffic can be encrypted, it still allows users to display the encrypted passwords of all the accounts via the ldaplist command similar to the ‘ypcat passwd’ command on NIS.

Use the pam_ldap module (Sun Microsystems, 2002). This allows the elimination of the proxyagent account. Be sure and

use TLS with pam_ldap to encrypt the ID and password being sent to the directory.

Make use of password policies within Sun Directory Server 5.2 that can be assigned to individual accounts. While you cannot count on the directory's expiration of an account to prevent someone from logging in, it will calculate the expiration dates for you. By allowing the directory to maintain the expiration date for you, you can 'lock' the account by changing the password and setting the login shell to /bin/false.

Consider creating a profile for each client to give you granular control over which LDAP directory a client will use. Since the profiles are stored on the directory, it will allow the LDAP administrator to cause a client to use a different LDAP directory without needing root access to the client.

Consider building your LDAP server environment to be redundant and robust by using content switches and directory proxy servers in front of your LDAP directories. This will provide load balancing and the ability to prevent denial of service attacks. The administrators will also have the ability to perform maintenance on the proxy and directory servers without a disruption in service.

## References

Bialaski, T (2000,Dec). Directory Server Security. Retrieved December 10, 2006, from Sun Blueprints Web site: http://www.sun.com/blueprints/1200/ldap-security.pdf

Faust, S (2001). TISC Insight, Volume 3, Issue 18. Retrieved December 22, 2006, from The Internet Security Conference Newsletter Web site: http://www.severus.org/sacha/docperso/intro_to_ldap_tisc.htm

Frisch, A (2002, Oct). Top Five Open Source Packages for System Administrators. Retrieved November 30, 2006, from O'Reilly ONLamp.com Web site: http://www.onlamp.com/pub/a/onlamp/2002/10/17/essentialsysadmin.html?page=1

Haines, M. & Bialaski, T. (2004). LDAP in the Solaris Operating Environment. Santa Clara, CA

Moffat, D (1999). Pluggable Authentication Module. Retrieved December 14, 2006, from Opensolaris.org Web site: http://opensolaris.org/os/community/security/projects/pam/

Nieminen, M (2006, July). What is Transport Layer Security?. Retrieved December 29, 2006, from SearchSecurity.com Web site: http://searchsecurity.techtarget.com/sDefinition/0,,sid14_gci557332,00.html

Sun Microsystems (2005, April). How to Hide Native LDAP userPassword attribute in ldaplist output. Retrieved November 30, 2006, from Sunsolve.sun.com Web site: http://sunsolve.sun.com/search/document.do?assetkey=1-9-73327-1 (Sunsolve account required for access)

Sun Microsystems, (2003). Retrieved December 12, 2006, from Understanding Solaris 9 Operating Environment Directory Services: Article is provided courtesy of Prentice Hall PTR. Web site: http://www.informit.com/articles/article.asp?p=31550&seqNum=3&rl=1

Sun Microsystems, (2004). System Administration Guide: Naming and Directory Services (DNS, NIS, and LDAP) . Retrieved December 12, 2006, from docs.sun.com Web site: http://docsun.cites.uiuc.edu/sun_docs/C/solaris_9/SUNWaadm/SYSADV5/p54.html

Sun Microsystems, (2002). Retrieved December 12, 2006, from docs.sun.com Web site: http://docs.sun.com/app/docs/doc/816-5221/6mbcm38sr?a=view

Sun Microsystems, (2003). System Administration Guide: Security Services. Retrieved December 14, 2006, from docs.sun.com Web site: http://docsun.cites.uiuc.edu/sun_docs/C/solaris_9/SUNWaadm/SYSADV6/p5.html

Sun Microsystems, (2005) Directory Proxy Server 5.2 Administration Guide. Retrieved December 14, 2006, from docs.sun.com Web site: http://docsun.cites.uiuc.edu/sun_docs/C/solaris_9/SUNWaadm/SYSADV6/p5.html

Tech-Faq, (2006). Retrieved December 22, 2006, from What is LDAP: Lightweight Directory Access Protocol? Web site: http://www.tech-faq.com/ldap-lightweight-directory-access-protocol.shtml