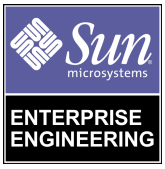
Directory Server Security

By Tom Bialaski - Enterprise Engineering

Sun BluePrints™ OnLine - December 2000



http://www.sun.com/blueprints

Sun Microsystems, Inc.
901 San Antonio Road
Palo Alto, CA 94303 USA
650 960-1300 fax 650 969-9131

Part No.: 806-7055-10
Revision 01, December 2000

Please
Recycle

Directory Server Security

Several security considerations must be taken into account when configuring the iPlanet™ Directory Server to support Solaris™ Operating Environment (Solaris OE) native Lightweight Directory Access Protocol (LDAP) clients. To protect naming service data from intruders, users must be authenticated before gaining access, and once authenticated must be authorized to perform operations on the naming service data. Since the LDAP security model for user authentication and authorization is different than the NIS model that administrators are familiar with, this article provides an overview of what the LDAP security model consists of and what security changes need to be made to accommodate the Solaris OE naming service requirements.

Additional information on this subject can be found in the Sun BluePrints™ book titled *Solaris™ and LDAP Naming Services* (ISBN: 0-030678-9), which will be published in December 2000 and available through www.sun.com/books, amazon.com, fatbrain.com, and Barnes & Noble bookstores.

LDAP Security Model

There are two aspects of directory security that affect how native LDAP clients interact with the directory server: authentication and authorization. Authentication is the process of identity validation, that is, proving a user's identity. Authorization is the granting of access rights to authenticated users. In the NIS naming service, clients bind to an NIS server without the need for authentication and all clients are granted the same permissions. All NIS maps are readable by all clients and the password shadow map is writable by users through the `rpc.yppasswdd` process which runs on NIS servers.

Compared to NIS, the iPlanet Directory Server provides a much more flexible security model. Clients can either be authenticated by one of several methods or connect to the directory server anonymously. Permissions, or access rights, can be set for the entire directory content, directory entries, or even data items within an entry.

LDAP Authentication

Clients authenticate to an LDAP server by attempting a *bind* operation. A connection between the client and the server is established *if* the bind is successful. As part of the bind request, the client chooses which authentication method it wants to use and supplies the credentials required by that method. If a method is not specified, credentials are not sent and the client is bound as an anonymous user. The available methods in iPlanet Directory Server 4.12 are simple name/passwords and digital certificates using secure socket layer (SSL).

The simple authentication method is the easiest to implement and the only method currently supported when the native LDAP client is serviced by the iPlanet Directory Server is used as the server. This method requires a distinguished name (DN) -and a password associated with the entry identified by the DN. The password is stored on the directory server in one of three formats: 1) clear text, 2) SHA-1, or 3) UNIX® crypt. When the server receives the password, either in clear text or in one of the hashed formats, a comparison is made. If a match is successful, then the authentication succeeds.

Unlike the Solaris OE, LDAP does not require a specific user account format. Any directory entry which contains the `userPassword` attribute can be specified as the bind DN. One exception to this is the root DN which has the common name (`cn`) of `Directory Manager`. This entry resides outside the directory information tree (DIT) and has privileges similar to what `root` has in the Solaris OE. Users who bind to the directory with this DN have unlimited access rights and can establish rights for other users.

LDAP Authorization

Authorization to directory objects is based on an access control list (ACL) which is maintained in the DIT. The ACL is composed of a series of access control instructions (ACIs). The ACI is an attribute that can be assigned to any directory object. Each ACI defines a set of access rules and multiple ACIs can be assigned to the same object. To define access rights for the entire directory, an ACI is set for the directory suffix of the DIT.

Since there can be, and usually are, multiple ACIs per DIT, rules are established to determine which ACIs take precedence. ACIs set on objects which appear lower in the DIT, take precedence over ACIs on objects above them. For example, you may

grant read permission for everyone at the top of the DIT, but restrict read access for subtrees that contain sensitive data. Access can be specified as **allow** or **deny**. If there is a conflict between ACIs, **deny** always takes precedence.

Note – The ACI is not defined in the LDAP standard and is only used in the iPlanet Directory Server implementation. Other LDAP servers have their own mechanism for assigning access rights.

The ACI Structure

Since ACIs are so fundamental to securing directory data, it is helpful to understand them in detail. During the configuration of the iPlanet Directory Server to support native LDAP clients, several ACIs are created or modified. While access rights on certain objects must be set correctly for native LDAP to work, other access rights are optional and are only recommended. You should have sufficient knowledge of how ACIs work to customize access rights in accordance with your company's security policy.

ACI Format

ACIs are in this general format:

<*target*> <*permission*> <*bind rules*>

- <*target*>—Specifies the entry to which the ACI is targeted. This is usually a subtree and/or an attribute that is targeted. In addition, an LDAP search filter can be specified.
- <*permissions*>—Defines what the access rights are. These include:
  - Read
  - Write
  - Search
  - Compare
  - Selfwrite
  - Add
  - Delete
- <*bind rules*>—Indicates which bind situation the ACI applies to.

The target is usually expressed as the path to the directory entry it applies to followed by the attribute(s) it applies to. Wildcards can be used to specify several or all attributes within an entry. Multiple attributes can be specified to target more than one specific attribute. If you wish to exclude certain attributes, the not (!) operator can be used.

Permissions are expressed by either allowing or denying access. Seven types of access are defined. Use these to restrict whether the data is used in searches and compares or whether values are modified or deleted. The purpose of one type listed above, called Selfwrite, may not be obvious. If Selfwrite is set, then users can add/delete themselves from groups.

The bind rules define which user(s) or bind DN, the ACI applies to along with what type of access is acceptable. Users can be placed in groups with permissions set for members of the groups. In addition to restricting or allowing access to user and groups, you can specify other binding criteria. For example, you could only allow connections from clients with particular IP addresses or restrict access to certain times of the day.

Use bind rules to set permissions for all users who successfully bind to the directory. They can also set permissions for users who do not supply a bind DN, which is also called an anonymous bind. It is important to note that these are two *different* situations although the keywords used, `all` and anyone, sound similar. If allowing anonymous binding is desired, the anyone keyword is required.

How Native LDAP Clients Bind

Before the native LDAP client can access naming service data in the iPlanet Directory Server, it must first perform a bind operation. Since this occurs during the boot sequence, a user is not logged in at the time. Therefore, the client must have a

bind DN and password stored locally to use the data. The client obtains this information from client profiles that are stored on the directory server. The following is a sample profile in LDAP Data Interchange Format (LDIF) format.

```
dn: cn=default,ou=profile,dc=blueprints,dc=com
        SolarisBindDN:
cn=proxyagent,ou=people,dc=blueprints,dc=com
        SolarisBindPassword: {NS1}c53708877bc6
        SolarisLDAPServers: 129.148.181.130
        SolarisSearchBaseDN: dc=blueprints,dc=com
        SolarisAuthMethod: NS_LDAP_AUTH_SIMPLE
        SolarisTransportSecurity: NS_LDAP_SEC_NONE
        SolarisSearchReferral: NS_LDAP_FOLLOWREF
        SolarisSearchScope: NS_LDAP_SCOPE_ONELEVEL
        SolarisSearchTimeLimit: 30
        SolarisCacheTTL: 43200
        cn: default
        ObjectClass: top
        ObjectClass: SolarisNamingProfile
```

The sample profile LDIF is generated by the `ldap_gen_profile` command. Of interest in the output is the DN:
`cn=proxyagent,ou=people,dc=blueprints,dc=com`

This is the DN the client will use to bind with. Associated with the bind DN is a password stored in the `SolarisBindPassword` attribute. This is stored in a hashed format and sent over the network to the iPlanet Directory Server along with the DN for the entry `cn=proxyagent`. This entry could be named anything, but the name `proxyagent` gives it some meaning. The entry is set up using the following LDIF:

```
dn: cn=proxyagent,ou=people,dc=blueprints,dc=com
cn: proxyagent
userpassword: secret
objectclass: top
objectclass: person
```

In the previous LDIF example, the objectclass `person` is used to create the entry, but any objectclass which contains the `userpassword` attribute will work. The profile information for binding is stored in the `/var/ldap/ldap_client_cred` file as shown below:

```
blueprints# more ldap_client_cred
#
# Do not edit this file manually; your changes will be lost.Please
use ldapclient (1M) instead.
#
NS_LDAP_BINDDN= cn=proxyagent,ou=people,dc=blueprints,dc=com
NS_LDAP_BINDPASSWD= {NS1}c53708877bc6
```

The Proxy Agent ACI

Since the native LDAP client is binding as `cn=proxyagent`, this DN needs to be able to access user passwords so that Solaris OE users can be authenticated. To enable this, an ACI must be created similar to the following LDIF.

```
dn: dc=blueprints,dc=com
changetype: modify
add: aci
aci: (target="ldap:///dc= blueprints,dc=com")
(targetattr="userPassword")
 (version 3.0; acl "password read"; allow (compare,read,search)
 userdn = "ldap:///
cn=proxyagent,ou=people,dc=blueprints,dc=com"; )
```

This ACI will be added to the top entry (suffix) of the directory server. It is important to note that compare, read, and search permissions must be granted.

Self Entry Modification

If the iPlanet Directory Server is installed using the **Typical** option, an ACI is automatically created that grants write permission to the entry which the user binds to the directory with. Leaving this ACI unchanged creates a security hole since the

user would be able to change the `uidNumber` attribute in the entry to 0. This effectively grants root permissions to that user. To prevent this, the default ACI is changed as reflected in the following LDIF.

```
dn: dc=blueprints,dc=com
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"; )
```

The `userPassword` attribute is excluded in the previous LDIF example, because users must still be able to change their passwords.

VLV Control Access

Another modification needed to the default ACI settings is to allow anonymous access to the Virtual List View (VLV) control object. As discussed in the *Directory Server Indexing* article , VLVs are used to create browsing indexes that are frequently used by the native LDAP client when accessing data. By default, an ACI is set to allow all users who successfully bind to the directory permission to access the VLV control object. However, during client initialization, an anonymous bind is required since a bind DN has not been established.

To allow anonymous binding, the `userdn` field needs to be changed from `all` to `anyone` as shown below.

```
dn: oid=2.16.840.1.113730.3.4.9,cn=features,cn=config
changetype: modify
replace: aci
aci: (targetattr != "aci")(version 3.0; acl "VLV Request
Control"; allow( read ,
 search, compare ) userdn = "ldap:///anyone";)
```

Authenticating Solaris OE Users

After the native LDAP client binds to the directory server, account information stored there is used to authenticate users attempting to log in. There are two options for authentication: 1) Traditional UNIX system-based and 2) LDAP-based. Which authentication method is used depends on the client's Pluggable Authentication Module (PAM) configuration. The PAM UNIX (`pam_unix`) module has been enhanced so that passwords stored in UNIX crypt format are retrieved from an LDAP server instead of an NIS or NIS+ server. A new PAM module, PAM LDAP (`pam_ldap`), is used to perform the authentication on the LDAP server.

While the PAM LDAP module is capable of performing authentication using sophisticated methods such as Challenge Response Authentication Mechanism, Message digest 5 (CRAM-MD5), the iPlanet Directory Server version 4.12 does not support this method. Simple username/password pairs are the only method currently supported through thePAM UNIX module. However, the use of this method is discouraged since passwords are transmitted in clear text over the network. Instead, use the enhanced PAM UNIX module since only the crypt format is sent over the network.

Conclusion

The LDAP security model is measurably different than any previous naming service supported by the Solaris OE. The iPlanet Directory Server provides a flexible mechanism for controlling access to directory data, but this mechanism can be quite complex and requires becoming well-acquainted with it. When configuring the iPlanet Directory Server to support native LDAP clients, modifications to the default ACIs is required. Understanding what these modifications do and why they are necessary will help you diagnose directory access problems you may encounter in the future.

Author's Bio: Tom Bialaski

Tom Bialaski is currently a Staff Engineer with the Enterprise Engineering group at Sun Microsystems, and is the author of "Solaris Guide for Windows NT Administrators." and co-athor of "Solaris and LDAP Naming Services". Tom has nearly 20 years of experience with the UNIX operating system and has been a Sun Engineer since 1984.