Apache based WebDAV Server with LDAP and SSL

Saqib Ali

saqib@seagate.com

Revision History

Revision v3.4	2002−06−29	Revised by: sa
Added the section "How to generate a CSR"		
Revision v3.3	2002−04−14	Revised by: sa
Add the section of DAV server management.		
Revision v3.2	2002−04−13	Revised by: sa
Added the Litmus (WebDAV compatility tester) sub−section.		
Revision v3.1	2002−04−11	Revised by: sa
Updated the introduction section		
Revision v3.0	2002−04−09	Revised by: sa
Added "Implementing and using SSL to secure WebDAV traffic" section		
Revision v2.1	2002−03−24	Revised by: sa
Refined the WebDAV Introduction and SSL section.		
Revision v2.0	2002−03−20	Revised by: sa
Added the SSL section		
Revision v1.1	2001−11−29	Revised by: sa
Initial public release.		
Revision v1.0	2001−11−01	Revised by: sa
Initial public release.		

.This document is an HOWTO on installing a Apache based WebDAV server with LDAP for authentication and SSL encryption.

Table of Contents

1. Introduction

The Objective of this document is to setup a Apache based WebDAV server that can authenticate against a LDAP server. This document will provide the basic groundwork for setting up a WebDAV server. It will also provide information on fine tuning and maintaining the server.

Note: If you encounter any problems installing Apache or any of the modules please feel free to contact me @ saqib@seagate.com

---

1.1. Copyright and License

1.2. What is WebDAV?

WebDAV stands for Web enabled Distributed Authoring and Versioning. It provides a collaborative environment for users to edit/manage files on web−servers. Technically DAV is an extension to the http protocol.

Here is a brief description of the extensions provided by DAV:

Overwrite Protection: Lock and Unlock mechanism to prevent the "lost update problem". DAV protocol support both shared and exclusive locks.

Properties: Metadata (title, subject, creater, etc)

Name−space management: Copy, Rename, Move and Deletion of files

Access Control:Limit access to various resources. Currently DAV assumes access control is already in place, and does not provide strong authentication mechanism.

Versioning:Revision control for the documents. Versioning is not implemented yet.

1.3. What do we need?

The tools needed to achieve this objective are:

- C Compiler e.g. GCC
- Apache Web Server
- LDAP Module for Apache
- WebDAV Module for Apache
- LDAP lib file

- SSL engine
- Mod SSL Libraries

NOTE: All of these packages are free and are available for download on the net.

We will compile all the above mentioned packages to produce binaries for the Web Server. The WebDAV and LDAP module will be compiled statically into Apache. Static compilation results in faster execution times. Apache was chosen, because it is the most versatile web server ever to exist. Market Shares for Top Web Servers shows that Apache has the largest web server market share.

Apache is also being used by:

- RackSpace − WebHosting provider
- CIHost − WebHosting provider
- SlashDot

1.4. Assumptions

This document assumes that you have the following already installed on your system.

1. gzip or gunzip − available from http://www.gnu.org
2. gcc and GNU make − available from http://www.gnu.org

The document also assumes there is a LDAP server installed elsewhere which will be used for the authentication.

1.5. Opinions and Suggestions

If you have any questions about the information available on this document, please contact me on the following email address: saqib@seagate.com

If you have comments and/or suggestions, please let me know as well!

---

2. Requirements

You'll have to download and compile several packages. This HOWTO will explain the compilation process, but you should be fimiliar with installing from source code.

2.1. Basics

You will need a machine running Solaris and GNU CC compiler. This compiler is available from http://www.sunfreeware.com. If you need any help installing the compiler please email me. If your OS doesnt already have gzip, you will need that as well. You can d/l gzip from http://www.sunfreeware.com as well.

Also create a directory /tmp/downloads . We will use this directory to store the downloaded source code.

2.2. Apache 1.3.x

Apache is the HTTP server, it will be used to provide the WebDAV services. Please download the Apache 1.3.x source code from http://www.apache.org/dist/httpd/.

2.3. OpenSSL

You will need to download the OpenSSL from http://www.openssl.org/source/ . Please download the latest version. OpenSSL installation will be used for SSL libraries for compiling mod_ssl with Apache, and for managing SSL certificates on the WebServer. Please download the OpenSSL source code gzipped file into /tmp/downloads

2.4. OpenLDAP

Download the OpenLDAP source code from http://www.openldap.org/software/download/. We will OpenLDAP for the LDAP lib files. You may also use IPlanet LDAP lib files. However GNU packages are recommended.

2.5. mod_dav

mod_dav will be used to enable DAV support in Apache. Download the source code for mod_dav from http://www.webdav.org/mod_dav/#how.

To find out more about mod_dav please visit http://www.webdav.org/mod_dav/faq/#00−00

2.6. mod_auth_ldap

mod_auth_ldap will be used for compiling LDAP support into Apache. Please download mod_auth_ldap from http://www.muquit.com/muquit/software/mod_auth_ldap/mod_auth_ldap.html

2.7. mod_ssl

mod_ssl will be used to enable SSL support in Apache, please download the mod_ssl source code from http://www.modssl.com/source/

3. Installing WebDAV services

Next is to first install pre−requisites (OpenSSL and OpenLDAP), and then Configure Apache with all the modules

3.1. Ground Work

To compile the WebDAV service with LDAP authencation capability, we will need to have the LDAP library files installed on the machine. The LDAP library files will be used to compile the LDAP module for Apache. Best way to get the LDAP library files is to download the OpenLDAP sourcecode from http://www.openldap.org and compile it to produce the required library files. You may use any other LDAP like IPlanet as well, but I recommend an OpenSource solution.

3.1.1. OpenLDAP lib files installation

Become root by using the su command:

```
$ su
```

Now change to the directory (`/tmp/downloads`) where you placed the OpenLDAP (tar) source file, and extract the content:

```
# cd /tmp/download
# gzip -d openldap-stable-xxxxxxx.tar.gz
# tar -xvf openldap-stable-xxxxxxx.tar
# cd openldap-x.x.xx
```

Now you can run "**configure**" for the openldap package. "**configure**" has many command line options. Type "**configure --help**" to see all options.

For this WebServer we dont really need the LDAP deamon, assuming there is a LDAP server running elsewhere. We just need the LDAP lib files. Since we will not be compiling the LDAP deamon, we will have to specify '`--disable-slapd`' as a command line option to '**configure**':

```
# ./configure --disable-slapd
```

`--disable-slapd` will tell the configure to not install the deamon. After you are done with configuring, you can make the dependencies for the openldap package:

```
# make depend
```

After making the dependencies the openldap package needs to be compiled. Use the **make** command:

```
# make
```

If everything goes OK, you will end up with compiled version of openldap in the current directory. Then you will need to install the compiled binaries and LDAP lib file into appropriate places:

```
# make install
```

Now you should have the compiled LDAP lib files required for the mod_ldap in the correct directory structure.

3.1.2. OpenSSL Engine

OpenSSL is required to create and manage SSL certificates on the webserver. The installion is also necessary for the lib files that will be used by the SSL module for apache.

Now change to the directory where you placed the OpenSSL source code files

```
# cd /tmp/download
# gzip -d openssl.x.x.gz
# tar -xvf openssl.x.x
# cd openssl.x.x
# make
# make test
# make install
```

3.2. Pre−configuring Apache

mod_dav requires that you have Apache pre−configured so that it knows where where everything is. Change back to the directory where you have the source files:

```
  # cd /tmp/download
  # gzip -d apache_1.x.x.tar.gz
  # tar -xvf apache_1.x.x.tar
  # cd apache_1.x.x
  # ./configure --prefix=/usr/local/apache
```

3.3. Configuring and Installing mod_dav

As mentioned above mod_dav will be statically linked with the Apache installation. Start by extracting mod_dav files:

```
# cd /tmp/download
# gzip -d mod_dav-1.x.x.tar.gz
# tar -xvf mod_dav-1.x.x.tar
```

Change to the NEW directory which was created during the extract:

 # cd mod_dav−1.x.x

Now configure the mod_dav package for static linking to Apache:

 # ./configure −−with−apache= /tmp/download/apache_1.x.x

Compile and install the files:

 # make

 # make install

mod_dav will have been partially compiled and placed into the Apache tree during the make install step.

3.4. Installing and configuring mod_auth_ldap

Change back to the temp download directory, and extract the mod_auth_ldap files:

```
# cd /tmp/download
# gzip -d mod _auth_ldap.tar.gz
# tar -xvf mod_auth_ldap.tar
```

Now install the modauthldap files to the Apache source tree:

```
# cd apache_x.x.x
# mv ../modauthldap ./src/modules/ldap
```

3.5. Installing and configuring mod_ssl

```
# cd /tmp/download
# gzip -d mod_ssl-2.x.x.tar.gz
# tar -xvf  mod_ssl-2.x.tar
# ./configure --with-apache=../apache_1.3.x.x
```

3.6. Configuring and Installing Apache

Finally we have reached the destination. But not yet.....

```
"The Journey is the Destination" (Jerry Garica of Grateful Dead)
```

Now we are ready to compile and install Apache with WebDAV and LDAP authentication for DAV.

Change back to the temp download directory:

 # cd /tmp/download

Change to the Apache tree directory:

 # cd apache-x.x.x

Now set the variable SSL_BASE to the OpenSSL lib files. On tcsh it will be as following:

```
# setenv SSL_BASE /tmp/download/openssl-0.9.x
```

This will the compiler where to find the SSL LIB files.

And now configure apache for the compilation with mod_dav, mod_auth_ldap, and mod_ssl:

```
 # ./configure --prefix=/usr/local/apache \
    --enable-module=ssl \
    --activate-module=src/modules/ldap/mod_auth_ldap.c \
    --activate-module=src/modules/dav/libdav.a \
    --enable-shared=ssl

    [...you can add more options here...]
```

--enable-shared is an optional, it tells the configure to compile SSL as dynamic module. Depending on the services that you will be providing, you may or may not need dynamic compilation.

Now compile the Apache and install it into the appropriate place:

 # make

Now create the SSL certification on the web server

 # make certificate TYPE=custom

Follow through the instructions, and you will have a certificate in no time. Remember CommonName is your FQDN (Fully Qualified Domain Name) e.g. dav.yourcompany.com

For details on creating and managing the SSL certificates, please read the section titled "Creating and Managing SSL certificates".

Now install Apache into its own directory

 # make install

---

4. Configuring and Setting up the WebDAV services

Now for the easy part. In this section we will WebDAV enable a directory under Apache root.

4.1. Modifications to the `/usr/local/apache/conf/httpd.conf`

Please verify that the following Apache directive appears in the `/usr/local/apache/conf/httpd.conf` :

```
  Addmodule mod_dav.c
```

If it does not please add it. This directive informs Apache about DAV capability. The directive must be placed outside any container.

Next we must specify where Apache should store the DAVLockDB file. DAVLockDB is a lock database for the WebDAV. This directory should be writable by the httpd process.

I store the DAVLock file under `/usr/local/apache/var`. I use this directory for other purposes as well. Please add the following line to your `/usr/local/apache/conf/httpd.conf` to specify that the DAVLockDB file will be under `/usr/local/apache/var` :

```
  DAVLockDB      /usr/local/apache/var/DAVLock
```

The directive must be placed outside any container.

4.2. Creating a directory for DAVLockDB

As mentioned above a directory must be created for DAVLockDB that can be written by the web server process. Usually web server process runs under the user '*nobody*' . Please verify this for your system using the command:

```
ps –ef | grep httpd
```

Under `/usr/local/apache` create the directory and set the permissions on it using the following commands:

```
# cd /usr/local/apache
  # mkdir var
  # chmod –R 755 var/
  # chown –R nobody var/
  # chgrp –R nobody var/
```

4.3. Enabling DAV

Enabling DAV is a trivial task. To enable DAV for a directory under Apache root, just add the following directive in the container for that particular directory:

```
DAV On
```

This directive will enable DAV for the directory and its sub−directories.

The following is a sample configuration that will enable WebDAV and LDAP authentication on `/usr/local/apache/htdocs/DAVtest`. Place this in the `/usr/local/apache/conf/httpd.conf` file.

```
<Directory /usr/local/apache/htdocs/DAVtest>
Dav On
#Options Indexes FollowSymLinks

AllowOverride None
order allow,deny
allow from all
AuthName "LDAP_userid_password_required"
AuthType Basic
<Limit GET PUT POST DELETE PROPFIND PROPPATCH MKCOL COPY MOVE LOCK UNLOCK>
Require valid−user
</Limit>
LDAP_Server ldap.server.com
LDAP_Port 389
Base_DN "o=ROOT"

UID_Attr uid
</Directory>
```

4.4. Create a Directory called DAVtest

As mentioned in a earlier section, all DAV directories have to be writable by the WebServer process. In this example we assume WebServer is running under username '*nobody*'. This is usually the case. To check httpd is running under what user, please use:

```
# ps −ef | grep httpd
```

Create a test directory called '`DAVtest`' under `/usr/local/apache/htdocs` :

 # mkdir /usr/local/apache/htdocs/DAVtest

Change the permissions on the directory to make it is read−writable by the httpd process. Assuming the httpd is running under username '*nobody*', use the following commands:

```
# cd /usr/local/apache/htdocs
# chmod −R 755 DAVtest/
# chown −R nobody DAVtest/
# chgrp −R nobody DAVtest/
```

4.5. Restart Apache

Finally you must run the configuration test routine that comes with Apache to verify the syntax in `httpd.conf`:

 # /usr/local/apache/bin/apachectl configtest

If you get error messages please verify that you followed all of the above mentioned steps correctly. If you can not figure out the error message feel free to email me with the error message (saqib@seagate.com).

If the configtest is successful start the apache web−server:

 # /usr/local/apache/bin/apachectl restart

Now you have WebDAV enabled Apache Server with LDAP authentication and SSL encryption.

4.6. WebDAV server protocol compliance testing

It is very important that the WebDAV that we just implemented be fully complaint with the WebDAV−2 protocol. If it is not fully compatible, the client side WebDAV applications will not function properly.

To test the complaince we will use a tool called Litmus. Litmus is a WebDAV server protocol compliance test suite, which aims to test whether a server is compliant with the WebDAV protocol as specified in RFC2518.

Please download the Litmus source code from http://www.webdav.org/neon/litmus/ and place it in the /tmp/downloads directory.

Then use gzip and tar to extract the files:

```
# cd /tmp/downloads
# gzip −d litmus−0.6.x.tar.gz
# tar −xvf litmus−0.6.x.tar
# cd litmus−0.6.x
```

Compiling and installing Litmus is easy:

```
# ./configure
# make
# make install
```

make install will install the Litmus binary files under `/usr/local/bin` and the help files under `/usr/local/man`

To the test the complaince of the WebDAV server that you just installed, please use the following command

```
# /usr/local/bin/litmus http://you.dav.server/DAVtest userid passwd
```

5. WebDAV server management

In this section we will discuss about the various management task − e.g. using LDAP for access control, and working with DAV method on Apache

Most of the configuration changes for the DAV will have to done using the `httpd.conf` file. This file is located at `/usr/local/apache/conf/httpd.conf`

`httpd.conf` is a text based configuration file that Apache uses. It can b editted using any text editor − I preffer using vi. Please make backup copy of this file, before changing it.

After making changes to the `httpd.conf` the Apache server has to be restarted using the **/usr/local/apache/bin/apachectl restart** command. However before restarting you test for the validity of the `httpd.conf` by using the **/usr/local/apache/bin/apachectl configtest** comand.

5.1. Restricting access to DAV shares

In the previous section when we created the DAVtest share, we used the LDAP for authentication purposes. However anyone who can authenticates using their LDAP useri/passwd will be able to access that folder.

Using the **require** directive in the httpd.conf file, we can limit access to certain individuals or groups of individuals.

If we look at the DAVtest configuration from the previosu section:

```
<Directory /usr/local/apache/htdocs/DAVtest>
Dav On
#Options Indexes FollowSymLinks

AllowOverride None
order allow,deny
allow from all
AuthName "LDAP_userid_password_required"
AuthType Basic
<Limit GET PUT POST DELETE PROPFIND PROPPATCH MKCOL COPY MOVE LOCK UNLOCK>
Require valid-user
</Limit>
LDAP_Server ldap.server.com
LDAP_Port 389
Base_DN "o=ROOT"

UID_Attr uid
</Directory>
```

We see that the **require** is set to **valid−user**. Which means any valid authenticated user has access to this folder.

5.1.1. Restricting access based on Individual UID(s)

LDAP UID can be used to restrict access to DAV folder.

require valid−user directive can be changed to **require user 334455 445566**

This will restrict access to individuals with UID 334455 and 445566. Anyone else will not be able to access this folder.

5.1.2. Restricting access based on groups of individuals.

require can also be used to restrict access to groups of individuals. This can be either done using LDAP groups or LDAP filters. The filter must be valid LDAP filter syntax.

5.2. Restricting write access to DAV shares

It maybe be required that the editting for the resources on the DAV shares be restricted to certain individual, however anyone can view the resources. This can be easily done using the **<Limit>** tags in the httpd.conf file

```
<Directory /usr/local/apache/htdocs/DAVtest>
Dav On
#Options Indexes FollowSymLinks

AllowOverride None
order allow,deny
allow from all
AuthName "LDAP_userid_password_required"
AuthType Basic
<Limit GET PUT POST DELETE PROPFIND PROPPATCH MKCOL COPY MOVE LOCK UNLOCK>
Require valid−user
</Limit>
LDAP_Server ldap.server.com
LDAP_Port 389
Base_DN "o=ROOT"

UID_Attr uid
</Directory>
```

You restrict write access to certain individuals by changing the **<limit>** to

```
<Limit PUT POST DELETE PROPPATCH MKCOL COPY MOVE LOCK UNLOCK>
Require 334455
</Limit>
```

Basically we are limiting the PUT POST DELETE PROPPATH MKCOL COPY MOVE LOCK and UNLOCK to an individual who has the UID of 334455. Everone else will be able to use the methods GET and PROPFIND on the resources, but not any other method.

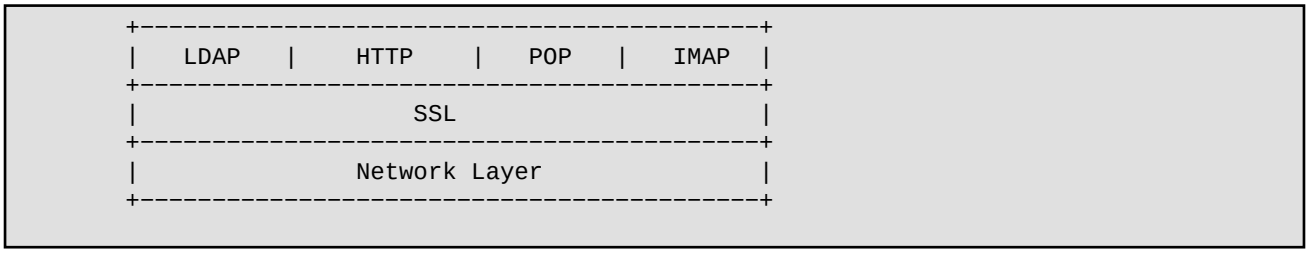
6. Implementing and using SSL to secure WebDAV traffic

Security of the data stored on a file server is very important these days. Compromised data can cost thousands of dollars to company. In the last section, we compiled LDAP authentication module into the Apache build to provide a Authentication mechanism. However HTTP traffic is very insecure, and all data is transferred in clear text − meaning, the LDAP authentication (userid/passwd) will be transmitted as clear text as well. This create a problem. Anyone can sniff these userid/passwd and gain access to DAV store. To prevent this we have to encrypt HTTP traffic, essentially HTTP + SSL or HTTPS. Anything transferred over HTTPS is encrypted, so the LDAP userid/passwd can not be sniffed. HTTPS runs on port 443. The resulting build from the last section's compilation process will have Apache to listen to both port 80 (normal HTTP) and 443 (HTTPS). If you are just going to use this server for DAV, then I will highly suggest that you close port 80. In this section of the HOWTO I will provide some information regarding SSL and maintaining SSL on a WebDAV server. However this is a not limited to a DAV server, it can be used on any web server.

6.1. Introduction to SSL

SSL (Secure Socket Layer) is a protocol layer that exists between the Network Layer and Application layer. As the name suggest SSL provides a mechanism for encrypting all kinds of traffic − LDAP, POP, IMAP and most importantly HTTP.

The following is a over−simplified structure of the layers involved in SSL.



6.1.1. Encryption algorithms used in SSL

There are 2 kinds of encryption algorithms used in SSL.

Public−Private Key Crytography − Initiating SSL connection: This algorithm is used for initiating the SSL session. In this algorithm, the encryption must be performed using the Public Key, and the decryption can only be performed using the Private Key. The Web−server holds the private Key, and sends the Public key to the client. The public key is sent to the client in a certificate.

1. The client request content from the Web Server using HTTPS.
2. The web server responds with a Certificate which includes the server's public key.
3. The client check to see if the certificate has expired.
4. Then the client checks if the Certificate Authority that signed the certificate, is a trusted authority listed in the browser. This explains why we need to get a certificate from a a trusted CA.

5. The client then checks to see if the Domain Name of the web server matches the Comman Name (CN) on the certificate?
6. If everything is successful the SSL connection is initiated.

Symmetric Cryptography − Actuall transmission of data: After the SSL connection has been established, Symmetric cryptography is used to encrypting data. Public−Private Key cryptography is CPU cycle intensive, so Symmetric cryptography is used. In symmetric cryptography the data can be encrypted and decrypted using the same key. The Key for symmetric cryptography was exchange in the initiation process.

6.2. Test Certificates

While compiling Apache we created a test certificate. We used the makefile provided by mod_ssl to create this custom Certificate. We used the command:

```
# make certificate TYPE=custom
```

This certificate can be used for testing purposes.

6.3. Certificates for Production use

For production use you will need a certificate from a CA. CA or Certificate Authorities are certificate vendors, who are listed as a Trusted CA in user's browser client. As mentioned in the Encryption Algorithms section, if the CA is not listed as a trusted authority, your user will get a warning message when trying to connect to a secure location.

Similarly the test certificates will also cause a warning message to appear on the user's browser.

6.4. How to generate a CSR

CSR or Certificate Signing Request must be sent to the trusted CA for signing. This section discusses howto create a CSR, and send it to the CA of your choice.

```
cd /usr/local/apache/conf/
/usr/local/ssl/bin/openssl req −new −nodes −keyout private.key −out public.csr
```

At this point you will be asked several about your server location, to generat the Certificate Signing Request

Note: You Common Name is the DNS name of your webserver e.g. dav.server.com . If you put in anything else, it will NOT work. Remember the passwd that you use, for future reference.

Once the process is complete, you will have private.key and a public.csr . At this pointe the public.key is not encrypted. To encrypt"

```
mv private.key private.key.unecrpyted
/usr/local/ssl/bin/openssl rsa −in private.key.unecrpyted −des3 −out private.key
```

6.5. Removing passphrase from the RSA Private Key

RSA Private Key stored on the webserver is usually encrypted, and you need a passphrase to parse the file. That is why you are prompted for a passphrase when start Apache with modssl:

```
# apachectl startssl
Apache/1.3.23 mod_ssl/2.8.6 (Pass Phrase Dialog)
Some of your private key files are encrypted for security reasons.
In order to read them you have to provide us with the pass phrases.
Server your.server.dom:443 (RSA)
Enter pass phrase:
```

Encrypting the RSA Private Key is very important. If somebody gets hold of the you "Unencrypted RSA Private Key" he/she can easily impersonate your webserver. If the Key is encrypted, the hacker can not do anything without the passphrase.

However encrypting the Key can sometimes be nuisance, since you will be prompted for a passphrase everytime you start the web−server. Specially if you are using rc scripts to start the webserver at boot time, the prompt for passphrase creates problems.

You can get rid of the passphrase prompt easily by decrypting the Key. However make sure that no one can hold of this Key. I would recommend Hardening and Securing guidelines be followed before decrypting the Key on the webserver.

To decrypt the Key:

First make a copy of the encrypted key

```
# cp server.key server.key.cryp
```

Then re−write the key with encryption. You will be prompted for the original encrypted Key passphrase

```
# /usr/local/ssl/bin/openssl rsa −in server.key.cryp −out server.key
read RSA key
Enter PEM pass phrase:
writing RSA key
```

One way to secure the decrypted Private Key is to make readable only by the root:

```
# chmod 400 server.key
```

6.6. Trusted Certificate Authorities

The following is list of Certificate Authorities that are trusted by the various browsers:

1. Verisign
2. Thawte