Apache based WebDAV with LDAP and SSL HOWTO

Saqib Ali

saqib@seagate.com

Revision History

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...1
1.1. Copyright and License...1
1.2. What is WebDAV?...1
1.3. What is mod_dav?...1
1.4. Why use WebDAV?...1
1.5. How does WebDAV work?...2
1.6. What do we need?...2
1.7. Assumptions...2
1.8. Opinions and Suggestions...3

2. Installing WebDAV services...4
2.1. Pre−Requirements...4
2.1.1. OpenLDAP lib files installation...4
2.1.2. OpenSSL Engine...5
2.2. Pre−configuring Apache...5
2.3. Configuring and Installing mod_dav...5
2.4. Installing and configuring mod_auth_ldap...6
2.5. Installing and configuring mod_ssl...6
2.6. Configuring and Installing Apache...6

3. Configuring and Setting up the WebDAV services...8
3.1. Modifications to the /usr/local/apache/conf/httpd.conf...8
3.2. Creating a directory for DAVLockDB...8
3.3. Enabling DAV...9
3.4. Create a Directory called DAVtest...9
3.5. Restart Apache...10

4. Implementing and using SSL to secure WebDAV traffic...11
4.1. Introduction to SSL...11
4.1.1. Quick and Dirty on Encryption algorithms used in SSL...11
4.2. Homegrown Certificates...12
4.3. Then why do I need a CA?...12
4.4. How to create CSR...12
4.5. Removing passphrase from the RSA Private Key...12
4.6. Trusted Certificate Authorities...13

1. Introduction

The Objective of this document is to setup a Apache based WebDAV server that can authenticate against a LDAP server. You will learn how to install, configure, run, maintain and fine−tune a Apache−based WebDAV 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.

WebDAV Support and Discussion Forum

This document is not a HOWTO on setting up LDAP server. For setting up a LDAP server please visit http://www.linuxdoc.org/HOWTO/LDAP−HOWTO−1.html

---

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:

Properties:

Name−space management:

1.3. What is mod_dav?

Please read http://www.webdav.org/mod_dav/faq/

1.4. Why use WebDAV?

Using WebDAV technology people throughout the world can be working on the same document. With the locking mechanism that DAV provides, the person who is editing the resource will lock it for editing. Others can read it but not modify it. When the person who is editing the document, is done editing, he/she will release the lock, and then others can modify the documents as well.

Authentication to gain access to resources is available in HTTP. This HOWTO discusses using LDAP for authentication. Permissions using groups and users in LDAP directory can be setup to control read/write access to various resources(documents) on the WebDAV server, so that you dont need to share passwords to work on the same documents

If you are worried about people trying to sniff your traffic, and gain access to confidential data, then worry not – you can always use HTTP with SSL ver3. This is the best encryption available. This documents details on how to implement WebDAV server which uses SSL encryption for transferring data.

1.5. How does WebDAV work?

text goes here...

1.6. What do we need?

The tools needed to achieve this objective are:

- Apache Web Server
- LDAP Module for Apache
- WebDAV Module for Apache
- LDAP lib file

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

1.7. 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.

1.8. 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. Installing WebDAV services

Our game plan is to first install OpenLDAP package. OpenLDAP is not required for Apache_LDAP_DAV to work, but we will need the installed OpenLDAP lib files to compile mod_ldap. And then we will compile Apache with mod_ldap and mod_dav.

Please download the following packages.

- Apache Web Server
- OpenLDAP (for LDAP lib files)
- mod_dav (Apache module for WebDAV)
- mod_ldap (LDAP module for LDAP)
- OpenSSL (SSL enginer for creating/managing certificates)
- mod_SSL (SSL Module for Apache)

2.1. Pre−Requirements

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.

2.1.1. OpenLDAP lib files installation

Become root by using the su command:

```
$ su
```

Now change to the directory where you placed the OpenLDAP (tar) source file, and extract the content. (I use a temp directory, `/tmp/download`):

```
# 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
```

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 into appropriate places:

```
# make install
```

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

2.1.2. 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
```

2.2. 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
```

2.3. 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.

2.4. 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
```

2.5. 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
```

2.6. 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

3. Configuring and Setting up the WebDAV services

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

3.1. 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.

3.2. 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/
```

3.3. 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 "username_and_passwd_required"
AuthType Basic
<Limit PUT POST DELETE PROPFIND PROPPATCH MKCOL COPY MOVE LOCK UNLOCK>
Require valid-user
</Limit>
LDAP_Server your.ldap.server.com
LDAP_Port 007
Base_DN "o=ROOT"

UID_Attr uid
</Directory>
```

3.4. 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/
```

3.5. 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.

If you have any questions, suggestions, recommendations, kudos, comments, or complaints, please email them to me (saqib@seagate.com).

Note: Please free to contact me if you encounter any problems installing Apache or any of the modules, or if you would like upgrade your existing Apache server to support WebDAV.

WebDAV Support and Discussion Forum

4. 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.

There are many administration tasks related with implementing and maintaining SSL encryption for a Web Site. You can either use commercial certicates, or generate your own. In the following sections we will briefly go over a few simple things that are required to get the SSL website up and running, and maintain it. This tutorial is by no means a full in−depth overview of the SSL.

4.1. 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.

This HOWTO discusses about the HTTP over SSL or simply HTTPS. For more information about POP/IMAP/LDAP over SSL, please visit http://www.stunnel.org. Or if you are looking for a commerical products look into IPlanet for LDAP and Netscape for IMAP and POP.

4.1.1. Quick and Dirty on 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.

4.2. Homegrown Certificates

The certicate that we created while installing Apache, is what I call a Homegrown Certificate. We used the makefile provided by mod_ssl to create this custom Certificate. We used the command:

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

These homegrown Certificates are usually used for testing purposes − to check if the Apache with mod_ssl installation worked.

4.3. Then why do I need 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 certificates that you created yourself will cause a warning message to appear on the user's browser.

If you are implementing a commercial website, I would highly recommend using a Certification Authority for getting your certificates signed. The warning message usually make the customers go away.

If you are on a "intranet", you can also implement your own CA, but it is a complicated task, and requires a lot of day−to−day management.

4.4. How to create 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. add text here......

4.5. 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
```

4.6. Trusted Certificate Authorities

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

1. Verisign
2. Thawte