How to secure WebDAV with SSL and Two-Factor Authentication

By Nick Owen
Published: 2007-04-16 16:20

How to secure WebDAV with SSL and Two-Factor Authentication

Benefits of Using WebDAV

Web-based distributed authoring and versioning or WebDAV is the protocol that is creating the "read-write" web. It is facilitating collaboration in many ways across the internet, replacing proprietary protocols (FrontPage, e.g.) or superseding less functional open protocols ((FTP, SFTP). The primary drivers for the adoption of WebDAV include:

- WebDAV resources can be set up like local drives allowing you to work with remote files as if they were on your machine.
- It allows for "locking" of files so multiple users can work with a file at the same time, but only one at a time can make changes.
- It is more efficient than FTP or SFTP. You can pipeline multiple transfers through a single TCP connection.
- It is an extension of HTTP and uses the same ports - 80 or 443, avoiding potential firewall issues.
- Support for WebDAV is available across multiple platforms creating a cross-platform solution.

However, as with all things Internet-related, security is an issue, particularly if you are dealing with confidential information. Yet companies always need to share information and work with outside personnel. It can be very tricky to collaborate with third-parties, yet still be able to authenticate users. How do you know that users aren't sharing a password? Yet, you don't want to provide a hardware token to a non-employee particularly for a short project, as you will more than likely never get it back.

A scenario where combining two-factor authentication with WebDAV might be for a public company that collaborates with an outside PR firm for financial releases. Knowledge of pending merger announcements or financial results is highly confidential corporate information and using two-factor authentication greatly reduces the risks of sharing static passwords. Many firms might also need to replace less-secure FTP services due to new compliance regulations.

For our purposes, a key benefit is that we can use the security tools available to protect HTTP services to protect WebDAV. In this how-to, we will create a secure WebDAV resource using Apache, Radius, SSL and two-factor authentication from WiKID Systems to set up secured remote drives on Windows, Mac and Linux machines. DAV Commands

It is a good idea to get familiar with WebDAV. Here are the new methods WebDAV adds to HTTP 1.1, according to Wikipedia:
- PROPFIND: Used to retrieve properties, persisted as XML, from a resource. It is also overloaded to allow one to retrieve the collection structure (a.k.a. directory hierarchy) of a remote system.

- PROPPATCH: Used to change and delete multiple properties on a resource in a single atomic act.
- MKCOL: Used to create collections (a.k.a. directory).
- COPY: Used to copy a resource from one URI to another.
- MOVE: Used to move a resource from one URI to another.
- LOCK: Used to put a lock on a resource. WebDAV supports both shared and exclusive locks.
- UNLOCK: To remove a lock from a resource.

Configuring the Server

For any service to be secure, the underlying server also needs to be secure. Be sure that your server is locked down tight and always up-to-date. Please note that I used Apache version 2.2.2-1.2 installed via RPM on a Fedora Core 5 System. I was not able to get mod_auth working on httpd-2.2.3-5 and have heard the same of 2.4. Create WebDAV directories

Create some directories for your user's files. You may even want to create individual directories:

```
mkdir /var/www/webdav/
mkdir /var/www/webdav/USERNAME
chown -R root:apache /var/www/webdav
chmod -R 750 /var/www/webdav
```
Note that the Apache user is the owner of these directories. **Configuring WiKID**

The WiKID Strong Authentication System is a commercial/open source two-factor authentication solution. Unlike most offerings which use shared-secrets, it uses public key cryptography to securely transmit PINs and one-time passcodes between the server and software tokens. With WiKID the two-factors are possession of the private key and knowledge of the PIN. Because we are testing across Windows, Mac and Linux, the screenshots here are of the open-source J2SE WiKID token. The token client uses port 80, so again, there are no firewall concerns. More information on WiKID's technology can be found [here](http://www.howtoforge.com/).

Here's how it will work, when the user wants access to the WebDAV resource, they will be prompted for a username and password. The user generates the one-time passcode on their WiKID token and enters it into the password prompt. Apache will route the username and one-time password to the WiKID server via pam_auth_xradius. If the username and one-time password match what WiKID expects, the server will tell Apache to grant access. First, we add Apache to the WiKID Strong Authentication Server as a network client, then add radius to Apache. I assume you already have a WiKID domain and users setup - more information on how to install and configure WiKID can be found [here](http://www.howtoforge.com/).

Start by adding a new Radius network client to the WiKID server for your web server:

- Log into WiKID server web interface (http://yourwikidserver/WiKIDAdim).
- Select **Network Clients** tab.
- Click on **Create New Network Client**
- Fill in the requested information.

- For the IP Address, use the web server IP address
- For Protocol, select Radius
- Hit the Add button, and on the next page, enter a shared secret
- Do not enter anything into the Return Attribute box
- From the terminal or via ssh, run 'stop' and then 'start' to load the network client into the built-in WiKID radius server

That is it for the WiKID server. Now on to Apache.

**Configure Apache**

If you are using a binary version of Apache, you will only need to load the modules in httpd.conf. If you are compiling yourself, use the --with-dav option
at compile time, and add support for the file system backend with the --enable-dav-fs option. Obviously, it is critical that Apache be configured for SSL to encrypt the data in transfer. If you do not have SSL keys, you will need to generate them:

```
openssl genrsa -out webdav.mydomain.com.key 1024
```

```
openssl req -new -key webdav.mydomain.com.key -out webdav.mydomain.com.csr
```

```
```

```
cp webdav.mydomain.com.key /etc/httpd/conf/
```

```
cp webdav.mydomain.crt /etc/httpd/conf/
```

Now, you can edit your httpd.conf file. First, validate that the DAV modules are loaded:

```
LoadModule dav_module modules/mod_dav.so
LoadModule dav_fs_module modules/mod_dav_fs.so
```

Load the mod_auth_xradius modules and set the time-out, in this case 1 hour:

```
LoadModule auth_xradius_module modules/mod_auth_xradius.so
AuthXRadiusCache dbm conf/authxcache
AuthXRadiusCacheTimeout 3600
```
And, create a locking database for WebDAV:

```http
<LocationMatch ^/lockdb>
  DAVLockDB /var/lib/dav/lockdb
</LocationMatch>
```

Make sure that Apache can handle various WebDAV clients:

```http
BrowserMatch "Microsoft Data Access Internet Publishing Provider" redirect-carefully
BrowserMatch "MS FrontPage" redirect-carefully
BrowserMatch "^WebDrive" redirect-carefully
BrowserMatch "^WebDAVFS/1.[0123]" redirect-carefully
BrowserMatch "^gnome-vfs/1.0" redirect-carefully
BrowserMatch "^XML Spy" redirect-carefully
BrowserMatch "^Dreamweaver-WebDAV-SCM1" redirect-carefully
```

Then, add a virtual host for SSL:

```http
NameVirtualHost *:443
<VirtualHost webdav.mydomain.com:443="">
  ServerName webdav.mydomain.com
  DocumentRoot /var/www/html
  SSLEngine on
  SSLProxyEngine on
  SSLCertificateFile conf/webdav.mydomain.com.crt
  SSLCertificateKeyFile conf/webdav.mydomain.com.key
  Alias /webdav/ '/var/www/webdav/
</VirtualHost>
```

http://www.howtoforge.com/
How to secure WebDAV with SSL and Two-Factor Authentication

```
DAV on
<Limit GET PROPFIND POST OPTIONS MKCOL PUT DELETE LOCK UNLOCK COPY MOVE PROPPATCH>
  AuthTypeSe Basic
  AuthName "You must be authenticated by WiKID to Enter!"
  AuthXRadiusAddServer "wikid_server_ip:1812" "webdav_shared_secret"
  AuthXRadiusTimeout 7
  AuthXRadiusRetries 2
  require valid-user
</Limit>
</Directory>
</VirtualHost>
```

A couple of notes: Depending on your setup, you might also need to add the location of your SSL cert and key to your ssl.conf file. I am using an alias here because I have the document root set to publish public facing webpages. I also am using the Limit command as defined in the Apache Documentation:

The purpose of the Limit directive is to restrict the effect of the access controls to the nominated HTTP methods. For all other methods, the access restrictions that are enclosed in the Limit bracket will have no effect. The following example applies the access control only to the methods POST, PUT, and DELETE, leaving all other methods unprotected.

You can set the AuthXRadiusCacheTimeout for whatever time you think is appropriate depending upon your needs and what you think the client environment is like. The more open the access, the more frequently your users should have to re-authenticate. Making it too short, however can cause problems with large file transfers, so if you are looking for a secure replacement to FTP, think about typical file sizes and transfer times.

Each sub-directory listed in the httpd.conf file inherits the security of it's parent, so if you want to further restrict access you can. For example, this virtual host add subdirectories that are limited to specific users:

```
NameVirtualHost *:80
<VirtualHost webdav.mydomain.com:443=
```
You could also have a top-level directory that was read-only and sub-directories that were write-able by certain users.

**N.B.:** I was getting a 412 Error when trying to use Cadaver on linux. Checking the audit.log showed that mod-security was blocking it. I had to comment out this line in mod-security:

```
# SecFilterSelective HTTP_Content-Type "!(application/x-www-form-urlencoded|multipart/form-data)"
```
Setting up clients Linux

On Linux, I used a package called Cadaver, which could not have been easier:

```
cadaver https://webdav.mydomain.com/webdav/
```

You will be prompted for your username and password, and then you have commandline access. Type 'help' to see a list of commands.

Setting up WebDAV on Windows was somewhat confusing as there seem to be a few ways to do it. However, there was only one way that seemed to consistently work. First you must enable basic authentication in the Windows registry. I opened a command prompt and typed regedit. I browsed to:

```
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\WebClientParameters
```

And right-clicked on Parameters, Add New, Selected DWORD and created "UseBasicAuth". Set the value to 1 to enable basic authentication. You can change it to 0 to turn it off again. Basic authentication sends the password in the clear so it is disabled, but we are using SSL and one-time passcodes, so it is not an issue here. I was able to add drag-and-drop access by adding the WebDAV address as a Network Place.

Start by clicking on the My Network Places icon on your desktop - Select Add Network Place:
This starts the wizard:
Select the bottom option, "Choose another network location":
Enter the URL of your WebDAV folder:
When prompted to enter your username and the WiKID one-time passcode:
Start the WiKID token and select the Domain associated with the WebDAV resource:
Enter the PIN:
And you will get back the one-time passcode. The OTP is time-bounded, but the time can be set on the WiKID server to whatever you want:
You will be asked to give the location a name:
That should be it, click Finish:
The location should open and you should be able to drag and drop a file from Explorer into this location. I was unable to map the WebDAV location to a driver letter. The directory listing entered an infinite loop. **MacIntosh OSX**
Setting up a WebDAV location on the Mac was a bit easier. Start in the Finder, selecting Go, Connect to Server. A dialog box will open.

Enter your username and password:
And that is it. You can drag and drop files to this location. **Linux**

Getting WebDAV to work on Linux is also simple. The trick is to use webdavs as the protocol. In your Konquerer, for example, you can enter: "webdavs://webdav.mydomain.com/webdav/". Additionally, if you would like to not enter the username each time, you can enter "webdavs://username@webdav.mydomain.com/webdav/". You can then bookmark that URL:
How to secure WebDAV with SSL and Two-Factor Authentication

In Gnome, the process is the same, but the URL would be "davs://username@webdav.mydomain.com/webdav/"

Conclusion

Combining WebDAV and Apache provides a great deal of flexibility. With flexibility often comes insecurity. However, by locking down your server, encrypting data in transit with SSL and using two-factor authentication, you can create a system that offers maximum ease of use without sacrificing security. Apache controls access and provides encryption in transit via SSL; Mod_auth_Xradius provides the credential caching and WiKID strongly authenticates the user.
Links of Interest

- WebDAV
- Apache
- WiKID Systems