System Administrator DeCal
Final Project

Due Date: May 1, 2007

1 Objective

You will be setting up a basic LAMP server with SSL support in a multi-user environment. You will also be populating the environment with several users.

2 Install the LAMP Server Components

A **LAMP** server is the generic term for a basic setup that includes a web server, database server, and scripting language. Though the meaning of a **LAMP** server changes depending on who you ask, we will take the most standard definition. In this definition, LAMP stands for Linux, Apache, MySQL, PHP.

To setup this server, you will need to install the following elements:

- Apache2
- PHP 5
- MySQL Server 5
- PHP5-MySQL Bindingsa
- Apache2 Mod PHP 5 Module

The most recent version numbers have been included in case you aren't sure which version to use. You can compile these elements from source code, but you will likely find it much easier to install them using the package manager. (Note that you can search the package repository in `aptitude` by typing '/'. This search term can include regular expressions). You will also need to change the port that the webserver listens on to 2XX80, where XX is your decal number. If you installed Apache from the debian package, this should be in `/etc/apache2/ports.conf`. If you installed apache from source, you will need to read the documentation.

3 Setting Up Web Analyzer

It is often useful to see what your webserver is doing. You could always read the logs yourself and find out, but its much easier to use an analyzer. The **Webalizer** is a fast, free web server log file analysis program. It produces highly detailed, easily configurable usage reports in HTML format, for viewing with a standard web browser. Install Webalizer on the main web site (not a user website). This can be downloaded and compiled from `http://www.mrunix.net/webalizer/`, or downloaded as a package from the debain repository. There are a lot of compile dependencies and a lot of post install configuration to be done with this, so I would suggest you use the package manager (unless you've got some free time).

4 Setting Up Per-User Websites

The basic LAMP setup will create a main webserver, but we want to allow individual users to be able to host their own sites (like `ocf.berkeley.edu/~aoaks`). This is accomplished in Apache by using the `userdir` module. If you installed Apache using the debian package (i.e. with aptitude or apt-get), you will find the `/usr/sbin/a2enmod` program to be quite useful. If you installed Apache from source you will need to read the Apache documentation (which is quite good) to see how to manually enable the module.

5 Setting Up MySQL

The default MySQL is setup is sufficient for most applications, but there are some default settings we want to change. First, start up MySQL by running:

```
i21:/# mysql -u root
```

This will connect you to the MySQL server as the MySQL root user. The first thing to do is set a password for the MySQL root user account. To do this, run (from the MySQL prompt):

```
mysql> SET PASSWORD FOR root@localhost=PASSWORD('new_password');
```

We also want to remove the default users from MySQL, so run:

```
mysql> use mysql;
mysql> delete from user where not (host="localhost" and user="root");
mysql> flush privileges;
```

6 Creating User Accounts

Now we want to create some accounts for users that would access the server. This will include:

- Creating a system account
- Creating a mysql user account and database
- Creating a web directory

6.1 System Account

The system account for user 'aaron' can be created easily enough using the `adduser` command.

6.2 Web Directory

Most of the work in creating a per-user website is done by Apache's userdir module, but the user still needs to create the web directory in his home directory. The user web directory is specified in the configuration for userdir and is usually `public_html`. Create this directory in the user's home directory and make sure it has the correct (user's) owner and group set before moving on.

6.3 MySQL Account

Creating the MySQL database and user account can be done quickly from the MySQL prompt. Login as the the MySQL root user with:

```
i21:/# mysql -u root -p
```

which should prompt you for the MySQL root password. Once in, you need to create a database for the user. The database name need not be the same as the system account. It is here for the sake of consistency. You can create the database with:

```
mysql> CREATE DATABASE aaron;
```

Then you need to create a user account and give it permissions on that database. Note that this account name need not be the same as the system account. It is here for the sake of consistency. This can be done quickly with:

```
mysql> GRANT ALL ON aaron.* TO 'aaron'@'localhost' IDENTIFIED BY 'some_password';
```

7 Installing Personal Web App

Pick a user that you have created and change to that user. This can be accomplished quickly with (and doesn't require a password if you are root):

```
i21:~# su - aaron
```

We'll use Wordpress as an example web app because everybody loves to blog. Download the most recent source code from the Wordpress website. Extract the contents and move them into the web directory. When you first go to the website, it will ask you to configure Wordpress, which will only require the MySQL user data you already know.

Once you have configured Wordpress, submit a sample blog post.

8 Setting Up Secure Apache

Using the normal webserver setup is fine for serving most data, but if you are going to be exchanging confidential information between the client and server (i.e. credit cards numbers, usernames/passwords, etc) you should be using a secure connection.

8.1 Generating A Certificate

Generating an SSL certificate from scratch will give you something to protect data transfer between the client and server. However, since it is not signed by a "trusted" source, it will generate warnings. Importing a paid and "trusted" certificate will remove the warnings, but you shouldn't be doing anything that would require a trusted certificate on these servers anyway.

Generating a self-signed certificate can be done using the "`make-ssl-cert`" command. You will first want to create a directory to store your certificate. I suggest `/etc/apache2/ssl/` . The `make-ssl-cert` command takes takes a configuration template, prompts for some information about the certificate, then generates the certificate in the specified location. To generate the certificate, run:

```
21:~# make-ssl-cert  /usr/share/ssl-cert/ssleay.cnf /etc/apache2/ssl/apache.pem
```

and answer the prompts.

8.2 Enabling SSL

Now that you have a certificate, you will need to enable SSL connections on the apache server. To do this, you will need to enable the "`SSL`" apache module and add a second `Listen` directive in `/etc/apache2/ports.conf` for a port to listen for secure requests on (I suggest 2XX81).

8.3 Configuring an SSL Host

Now that you have a certificate and Apache is configured to load and accept SSL connections, the last step is to have a virtual host setup to handle these requests. The debian configuration system uses the modular sites-available/sites-enables system. The sites available for use have configuration files stored in `/etc/apache2/sites-available/`. The sites are then enabled by placing symlinks to the configuration files in `/etc/apache2/sites-enabled/`. When apache starts up, it will load all the sites in `sites-enabled`. You can see an example of this with the "default" site that is installed.

To create an SSL host, copy the configuration file for the default site in `sites-available` to something like "ssl". To enable SSL on this host, you will need to make a few changes to the configuration file. You will need to add a port to the "`NameVirtualHost`" and "`<VirtualHost>` directives like so:

```
NameVirtualHost *:2XX81
<VirtualHost *:2XX81>
```

You will also need to add the following lines somewhere in the `<VirtualHost>` block:

```
SSLEngine on
SSLCertificateFile /etc/apache2/ssl/apache.pem
```

8.4 Enabling the SSL Site

Now that the SSL site has been configured, it need to be enabled. Create a symbolic link in the `sites-enabled` directory the the SSL configuration in `sites-available`:

```
i21:~# ln -s /etc/apache2/sites-available/ssl /etc/apache2/sites-enabled/001-ssl
```

With the site now enable, restart the Apache server with:

```
i21:~# /etc/init.d/apache2 restart
```

Now test it out. Go to `http://tempest.ocf.berkeley.edu:2XX80` and see if you get the normal site. Now go to `https://tempest.ocf.berkeley.edu:2XX81` and see if you get a secure site. If you do, then congrats.

9 Installing Secure Webmail

Since logging into a webmail system will involve the exchange of usernames and password between the client and the webserver, this should be accessed on the secure site. For this example, we will use RoundCube Mail. This is a cool new webmail interface, but it is still in beta, so I make no claims about its reliability and don't suggest it for production use. That said, its still pretty cool and its really easy to setup. Ideally, you would have this connect to your own mail server's IMAP and SMTP servers, but we will use CalMail here because they are already set up.

As a challenge, this section will be unassisted. Download the latest stable release from the RoundCube website and install it (fine, one hint: read INSTALL in the download). When configuring the IMAP and SMTP settings, use
`ssl://calmail.berkeley.edu` as the server for both and default ports `993` and `465` for IMAP and SMTP respectively. When connecting to this webmail client, be sure to use your SSL secured server since you will be entering a username/password.

10 (Optional, but useful) No-IP Dynamic DNS

Everything you've done here can be done on a server you set up at home. What keeps many from running a server at home is that their dynamic IP address makes it difficult to connect to the server remotely. How are you going to run a webserver if the address keeps changing? One solution is to buy a static IP. Unfortunately, we aren't all that wealthy, so an alternative is to use a dynamic dns provider, like No-IP. You can sign up for a free hostname, and with the help of a server daemon, keep that hostname updated with your server's frequently changing IP.

10.1 Getting an Account

Getting a No-IP account couldn't be simpler. Go to no-ip.com and sign up for a free account. No-IP offers several paid plans with some interesting features, but for now you should be fine with a "No-IP FREE" account.

10.2 Adding A Host

Once you have an account and are logged in, you will need to add a host. A single No-IP account can be used to manage several hostnames, so you can add more later if you find the need to.

Under "Your No-IP", go "Hosts/Redirects", "Add". Here you will specify the hostname information for your host. Pick a hostname and a domain from the list of free domains. Choose "DNS Host (A)" as the host type. (In case you forgot, A type records are DNS records that map a name to an IP). For testing purposes, set the IP to `192.58.221.243`. This is the IP of OCF's webserver. The rest you can leave default/blank. Once you create the host, try pointing your browser at your domain name. (Note that it may take a few minutes for DNS to update.) If you get the OCF homepage, congrats, your host is set up.

10.3 Downloading the Updater

Most people don't have a static IP address, so you will want to download their update client. This will run on the server and periodically check your IP, updating your No-IP host when it changes. Get the Linux client from the No-IP Downloads section.

10.4 Setting Up the Daemon

After you download and untar the source, you will need to build the daemon. You may have notices that there is no `configure` script. This is a very simple daemon, so only a `make` and `make install` are needed. You will be configuring the the daemon during the install step. It will prompt you for your No-IP login info, along with which hosts you want to update with this daemon.

10.5 Adding to Startup

You will probably want this daemon to run at system startup. This way you don't have to worry about restarting the daemon every time you need to reboot. The download package comes with a debian startup script already written, all you have to do is install it. It should be in untarred directory as `debian.no-ip.org.sh`. Move this file to `/etc/init.d/noip2.sh`. The `/etc/init.d` directory contains all of the system startup scripts, as you will see if you look a the other files in that directory. It is a shell script, so you will also want to make it executable (look at the other files in `/etc/init.d` if you are unsure about what the permissions should be). The last step is actually adding it to system startup. This can be done easily in debian by running "`update-rc.d noip2.sh defaults`".

10.6 Starting the Daemon

Once the startup script is installed, its an easy matter to start it up. Just run "`/etc/init.d/no-ip2.sh start`" to start it. Assuming you get no errors, and giving sufficient time for DNS to update, the next time you access your No-IP host you should the the decal homepage. If you do, then congrats, your updater is working property. Note that to get to your website, you will still need to specify the ports for your vserver, because No-IP is only providing a link to your IP address.

It may seem like kind of a waste to do this on the decal server because it already has a hostname and a static IP address, but if you set up a server at home, you probably don't have either and this will be quite useful. Note that if you do set up a server at home, you will probably need to set up port forwarding correctly on your router before you can get incoming traffic, but that is usually a pretty simple task (though router dependent, so I won't go into it here).

11 Submission

Submit working links (that means include http(s) and the port number) to:

- Your main webserver site
- Your Webalizer page
- Your user blog
- Your SSL site
- Your webmail install
- Your No-IP domain (if you did it and don't mind telling me)

Include the following at the top of your e-mail:

- Your Name (as it shows in your university registration)
- Your SID
- Your INST account name (cs198-f*)
- Your tempest vserver identifier (iXX)