Apache and...

Virtual Hosts ---- aliases

mod_rewrite ---- htaccess

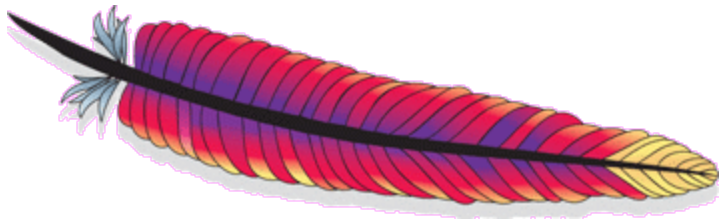
What is Apache?

Very good overview here:

http://en.wikipedia.org/wiki/Apache_web_server

The Apache web site is an excellent source of information as well:

http://www.apache.org/



Quick Facts

- Initially released in 1995
- Used on around 250 million web sites
- Approximately 65% of all sites worldwide use Apache
- Runs on Unix, Linux, FreeBSD, Solaris, Netware, Mac OS X, Windows, OS/2 and more.
- Licensed under the Apache License. Incompatible with GPL version 2, compatible with version 3.
- Originally designed by Robert McCool who was involved with the original web server, NCSA's HTTPd.
- Named “Apache” either because it involved many patches to the original NCSA server, or after the American Indian Apache tribe.

What is a Virtual Host?

There are two types:

- Name-based
- IP-based

We will be configuring named-based virtual hosts.

This allows a single IP address to serve many web sites from a single server. This is possible because the web client sends the name of the site it wishes to connect to as part of its initial connection request.

Issues

- Originally with HTTP/1.0 headers the hostname was not required to be included. Some browsers, notably Internet Explorer did not include the site name. This caused name-based hosting to fail.
- HTTP/1.1 released in 1999 requires the hostname to be part of the header. So, this is no longer an issue.
- SSL fails with name-based hosting as the hostname is not part of the initial TLS/SSL handshake – thus you cannot match the correct certificate to use for each site. For some ssl virtual hosting tricks see:

http://wiki.apache.org/httpd/NameBasedSSLVHosts

IP-based Hosting

- This requires a separate IP address for each hostname on a web server.
- IP-based hosting works with current SSL implementations.
- IP-based hosting (can) works even if DNS has failed.
- However, requires an IP address for each site. This may not be possible and requires more effort to implement.

Configuration Considerations: Apache

- Directory naming conventions. Decide upon one from the start:
  - /usr/local/www/?? (FreeBSD)
  - /var/www/?? (Linux)
- What to do about default actions? We'll give an example in our exercises.
- Must deal with directory permissions in more detail.

Questions?

?

Other Popular Apache Items

Three include:

- aliases
- mod_rewrite
- htaccess

Aliases

Allows you to specify a web directory name that maps to a separate directory *outside* the file structure of a web site.

For example:

Your site is `http://www.example.com/`

The site resides in `/usr/local/www/share/default/`, but you want the files in `/usr/local/www/books/` to be available at `http://www.example.com/books/`

How would you do this?

Aliases continued

In the file `httpd.conf`...

```
Alias /books /usr/local/www/books
```

But, you must set Directory permissions as well. For instance:

```
<Directory “/usr/local/www/books”>
      Options Indexes FollowSymLinks
      AllowOverride None
      Order allow,deny
      Allow from all
</Directory>
```

Remember, case counts in Apache configuration files!

mod_rewrite

Allows you to redirect requests from a page, or a pattern of pages to another page, or another pattern of pages.

- Extremely powerful
- Uses regular expression language
- Can save you if you move a large number of pages

In order to use `mod_rewrite` the rewrite module must be part of your Apache install and it must be loaded:

```
/etc/apache2/mods-enabled/rewrite.load
# a2enmod rewrite
# server apache2 restart
```

mod_rewrite continued

Here is some sample code where `mod_rewrite` is actually used (from `sites/000-default`):

```
# turn on the use of the mod_rewrite module
        RewriteEngine on

# trac logins must be secure
        RewriteCond %{SERVER_PORT} !443
        RewriteCond %{REQUEST_URI} ^/trac
        RewriteRule ^(.*)$ https://nsrc.org$1 [R=301]
```

Forces all users viewing wiki pages to do so using https (SSL), including logins. *Very commonly used.*

[R=301] ➔ Apache Redirect Response

htaccess

1. Use mod_rewrite to force https for any directory
2. Use htaccess to require a password.

Create a file “.htaccess” in the directory you wish to protect. In that file you might have something like this:

```
AuthName ”My Personal Photos"
AuthType Basic
AuthUserFile /home/user/public_html/photos/.htpasswd
require user sebastian
```

Note the file “`.htpasswd`” above. This is where you store user/password information. You do this by running and using the `htpasswd` command.

htpasswd command

To create an initial .htpasswd file with a user and password you do:

```
# htpasswd -c .htpasswd username
```

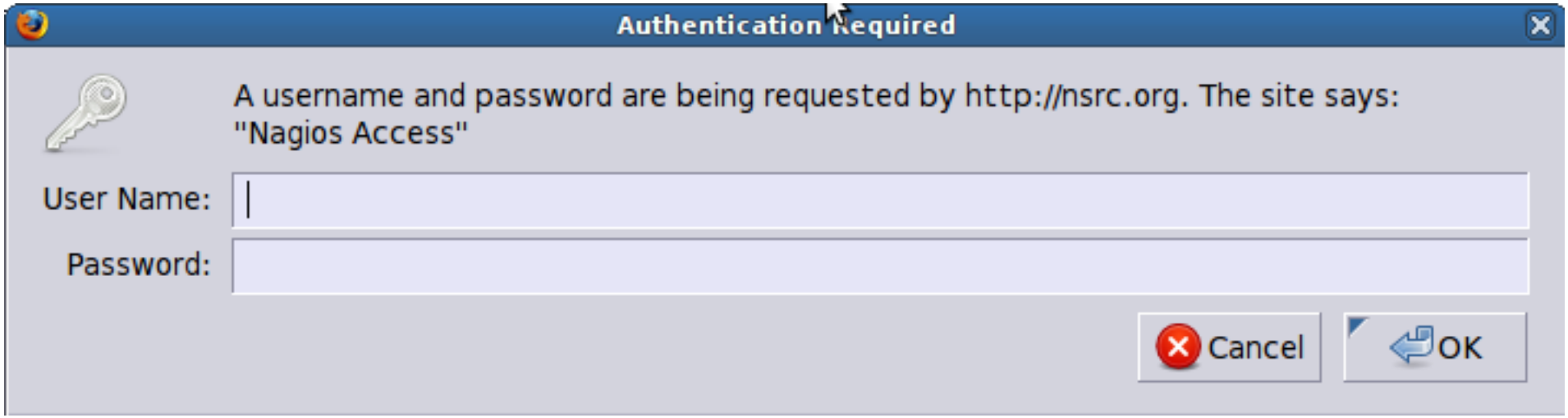
The “-c” parameter says to create the file. Enter in the password when prompted. For the next user do:

```
# htpasswd .htpasswd username
```

To change a password just run the command again.

And, in the end you'll see a prompt like this...

htaccess



Questions?