Web Server Security and Access Control

Prof. Francesco Bergadano

Dipartimento di Informatica
Università degli Studi di Torino

francesco.bergadano@di.unito.it

Sommario

Issues to be addressed....3
The Apache Web Server....4
Installation of the Apache Web Server....5
Configuration of the Apache Web Server....8
Global Directives for the Apache Web Server....9
User authentication....10
Virtual hosting....15
HTTPS....16
Form-based authentication....21
PHP / MySQL integration in Apache....22
Example of form authentication....24

Issues to be addressed

- Apache Web Server
- User Authentication
- Session management and cookies
- Cookie authentication

The Apache Web Server

An open source and multiplatform Web Server

History:

- Version 1.0 released in 1995
- Starting in 1996 it becomes the most popular Web Server

Usage:

- Often combined with PHP and MySQL
- Often under Unix/Linux
- Hence the so-called solution stack “LAMP”
- Alternatively “WAMP” with Windows
- Available as a Cloud PAAS (Platform As A Service)

Installation of the Apache Web Server

Download version 2.2.13 from
http://httpd.apache.org/
http://archive.apache.org/dist/httpd/

Later also install OpenSSL (version 1.0.0b)

Create an “apache” directory in your HOME

Do configure, make and make install:

```
./configure --enable-module=SO
                        --enable-ssl
                        --prefix=$HOME/apache
make
make install
```

The file apache/bin/apachectl is used to start, stop and restart the Apache Server Daemon (httpd), that will wait for client connections:

```
apachectl start / restart
apachectl stop
```

If you do not have administrator privileges that will allow you to start a daemon on the well known port 80, you will need to use another port, e.g. port 8080:

Modify the configuration file conf/httpd.conf with the lines:

```
Listen 8080
ServerName your_server_DNS_name
```

The Unix command

```
ps auxw | grep httpd
```

will show all active processes related to the http daemon.

To check configuration file correctness, simply run

```
httpd –t
```

Move to the directory “htdocs”. Here are the files to be served by Apache.

Change htdocs/index.html as desired

Run the CURL command line browser:

```
curl http://localhost:8080  -v
```

Access http://localhost:8080/index.html from a browser

Configuration of the Apache Web Server

conf/httpd.conf is the main Apache configuration file

- It contains directives in plain text
- Restart the server after modifying httpd.conf in order to make the changes effective.
- # is prepended to comments

Directives are included in blocks:
<Directory>, <DirectoryMatch>, <Files>, <FilesMatch>, <Location>, <LocationMatch>, <VirtualHost>, …

e.g.
<Directory /usr/local/httpd/docs> directives </Directory>
<DirectoryMatch "^/www/.*/[0-9]{3}"> directives </Directory>

Blocks may be nested
Top level directives are for the entire server

Global Directives for the Apache Web Server

- ServerRoot *path*
  - Base directory
- Listen *port*
  - Port where httpd is listening
- ServerAdmin *emailaddress*
  - Email address of server administrator
- ServerName *URL*
  - Domain name for this server
- Timeout *n*
  - Max request timeout in seconds
- DefaultType *MIMEtype*
  - Default MIME type for server responses (will go in the HTTP response Content-type header)

#

User authentication

- Create a subtree of htdocs, to be accessed only by some users
- Create a password file, in a directory that may not be accessed from the outside (e.g. apache/passwd)
- Use the command htpasswd
  - **htpasswd [options] password_file_path username**
  - Options:
    - **-c** : create a new file
    - **-m**, **-s** : encode with MD5 or SHA1
    - **-p** : save password in plaintext
    - **-D** : delete the user
    - …

e.g. htpasswd -c /usr/home/…/apache/passwd/password mario
will ask for mario's password and store it in the specified file

be careful, there is a salt and Unix crypt() is used, so the file will not be the same if the above is done more than once. To verify the password, the salt must be given as input (first two chars after the username: string). For example, for user fpb with password fpb, we have:

```
# more fpbpass
fpb:/oI1Mqa1cZV5o
# openssl passwd -crypt -salt /o fpb
/oI1Mqa1cZV5o
```

In order to specify which directories require passwords, directives must be added in httpd.conf, e.g.:

```
<Directory /usr/home/…/apache/htdocs/protected>
   AuthType Basic
   AuthName “plumbing”
   AuthBasicProvider file
   AuthUserFile /usr/home/…/apache/passwd/password
   Require user mario
</Directory>
```

AuthType Basic indicates the type of authentication (Basic in this case)
AuthName “plumbing” specifies the authentication realm
AuthBasicProvider file indicates that passwords are in a file
AuthUserFile /usr/home/…/apache/passwd/password names the file
Require user Mario tells the server which user can access the directory

On can try to access via a password with

- curl http://localhost:8080/protected -v

or by accessing localhost:8080/protected with a Browser

- In order to restrict to any authenticated user, use **Require valid-user**
- In order to restrict to a group, use the following syntax in a group file:
  GroupName: userName1 userName2 …
  e.g.
  plumbers: mario luigi

```
<Directory /usr/home/…/apache/htdocs/protected>
        AuthType Basic
        AuthName “plumbing”
        AuthBasicProvider file
        AuthUserFile /usr/local/apache/passwd/passwords
        AuthGroupFile /usr/local/apache/passwd/groups
        Require group plumbers
</Directory>
```

Where the group file in the above example is
usr/local/apache/passwd/groups
And it will contain the simple line
plumbers: mario luigi

One may use many access restrictions, and the directives ‘order’ (allow, deny or deny, allow) and ‘satisfy’ (all or any) will specify the combination methods.

The order “allow, deny” will be interpreted as follows: first try all ‘allow’ directives, at least one must match for the request to be allowed; then try all ‘deny’ directives, and reject the request if at least one matches, otherwise accept.

The satisfy directive will work as follows:
- Any: pass if either user auth or address auth passes
- All: pass if both pass

```
<Directory /usr/home/…/apache/htdocs/protected>
   AuthType Basic
   AuthName “construction”
   AuthBasicProvider file
   AuthUserFile /usr/local/apache/passwd/passwords
   Require valid-user
   Order allow, deny
   Allow from 192.168.1.1
   Deny from 192.231.2.2
   Satisfy Any
</Directory>
```

User authentication and address based access control may be centralized as done above, through the httpd.conf file

Alternatively, it may be programmed in .htaccess files within every directory (and will be inherited in subdirectories). This distributed access control specification method may be combined with centralized access control, and .htaccess directives override or add specifications to centralized directives.

To allow distributed access control through .htaccess files, use the AllowOverride directive, as follows:

```
<Directory …>
        AllowOverride All / None / directive_type
</Directory>
```

Use restrictive access control for the root directory, e.g.:

```
<Directory />
        Options FollowSymLinks
        AllowOverride None
        Order deny, allow
        Deny from all
</Directory>
```

The option FollowSymLinks will direct Apache to follow symbolic links in the file system, when looking for the requested resource.

Virtual hosting

In httpd.conf, remove # from
#Include conf/extra/httpd-vhosts.conf

Create two directories (loopback and localhost) in htdocs

In conf/extra/httpd-vhosts.conf create two Virtual Hosts:

```
<VirtualHost *:8080>
	ServerAdmin email_addr
	DocumentRoot "…/htdocs/localhost"
	ServerName localhost
	ServerAlias …
	ErrorLog "logs/localhost-error_log"
	CustomLog "logs/localhost-access_log" common
</VirtualHost>

<VirtualHost *:8080>
	ServerAdmin email_addr
	DocumentRoot "…/htdocs/loopback"
	ServerName 127.0.0.1
	ServerAlias …
	ErrorLog "logs/loopback-error_log"
	CustomLog "logs/loopback-access_log" common
</VirtualHost>
```

Use a browser with

- http://localhost:8080
- http://127.0.0.1:8080

HTTPS

Basic authentication sends passwords in cleartext

Server-side certificates and HTTPS may prevent this

Client side authentication may provide even higher security levels

Creating key pairs and Public Key Certificates (with OpenSSL)

Create an RSA key pair and a self-signed certificate (ca-bundle.crt)
Create an RSA key pair for the server. Rename the secret key to server.key
With the CA key, create a certificate for the Web Server (server.crt)

In order to create the CA’s self-signed certificate:

openssl genrsa -out CA.key 2048 (generate a key pair in CA.key)
(alternatively use
openssl genpkey -out CA.key -algorithm RSA -pkeyopt rsa_keygen_bits:2048
instead of the first line)

```
openssl rsa -pubout -in CA.key -out CA.pub
openssl req -key CA.key -new -x509 \
        -days 365 -out ca-bundle.crt
```

In order to create the server certificate:

```
openssl genrsa -out server.key 2048
openssl rsa -pubout -in server.key -out server.pub
openssl req -key server.key -new -out server.req
openssl x509 -days 365 -CA ca-bundle.crt -CAkey
        CA.key -CAcreateserial -CAserial ca.srl
        -req -in server.req -out server.crt
```

Now insert the following directives in httpd.conf:

```
Include conf/extra/httpd-ssl.conf
<IfModule ssl_module>
        SSLRandomSeed startup builtin
        SSLRandomSeed connect builtin
</IfModule>
```

The save the keys and certificates on the server machine:

- server.key and servert.crt in apache/conf
- ca-bundle.crt in a new directory apache/conf/ssl.crt
- Modify conf/extra/httpd-ssl.conf as follows:

```
Listen 8443
…
<VirtualHost *:8443>
        …
        ServerName localhost:8443
        DocumentRoot "…/apache/htdocs/ssl"
        SSLEngine on
        SSLCertificateFile "…/apache/conf/server.crt"
        SSLCertificateKeyFile "…/apache/conf/server.key"
        SSLCACertificateFile "…/apache/conf/ssl.crt/ca-bundle.crt"
        …
</VirtualHost>
```

Try https://localhost:8443 with any browser.

Be careful in order to set the URL and the common name in the certificate to the same value.

Now we may:

- Use basic authentication as above, and passwords are encrypted with the rest of the session
- Require SSL client-side authentication, if the user has a certificate.

SSL client authentication in Apache

Any certificate, but signed directly by the CA in the specified file ca.crt

```
SSLVerifyClient require
SSLVerifyDepth 1
SSLCACertificateFile conf/ssl.crt/ca.crt
```

Selected certificate, CA in the specified file ca.crt

```
SSLVerifyClient      none
SSLCACertificateFile conf/ssl.crt/ca.crt
SSLCACertificatePath conf/ssl.crt

<Directory /usr/local/apache2/htdocs/secure/area>
  SSLVerifyClient      require
  SSLVerifyDepth       5
  SSLOptions           +FakeBasicAuth
  SSLRequireSSL
  SSLRequire       %{SSL_CLIENT_S_DN_O}  eq "mario" \
               and %{SSL_CLIENT_S_DN_OU} in {"plumbers", "friends"}
</Directory>
```

Form-based authentication

As an alternative to basic authentication or SSL client authentication, we may use form-based authentication.

This may be combined with SSL with server certificates only to produce a pontentially secure solution.

PHP / MySQL integration in Apache

This will be done, in our examples, using PHP and MySQL. First, we need to follow these instructions:

Download version 5.3.6 (www.php.net).
Download the library LibXML2 2.7.8 (ftp://xmlsoft.org/libxml2).
Extract the above in your home.
Execute the following commands:

```
./configure -prefix=$HOME/php \
   --with-apxs2=$HOME/apache/bin/apxs \
   --with-libxml-dir=$HOME/libxml2-2.7.8
make
```

Now, in order to make Apache interpret php files, we need to:

1. copy the php-5.3.6/libs/libphp5.so in apache/modules
2. add the following directives to the httpd.conf configuration file:

```
LoadModule php5_module modules/libphp5.so
<Files *.php>
SetOutputFilter PHP
SetInputFilter PHP
LimitRequestBody 9524288
</Files>
AddType application/x-httpd-php .php
AddType application/x-httpd-php-source .phps
DirectoryIndex index.php
```

To test, make a mytest.php page with the following content:

```
<?php
        phpinfo();
?>
```

And save it in the htdocs directory.

Try accessing
http://localhost:8080/mytest.php
from any Browser, a default info page should be displayed (not the text content of mytest.php).

Example of form authentication

Examples … see sec.educ.di.unito.it/examples (first start apache)