Configuration and Hardening

Building a Fortress in a greenfield

Dr. Hale

University of Nebraska at Omaha

Secure Web Application Development – Lecture 6

Today's topics:

Hardening your deployment (production) environment

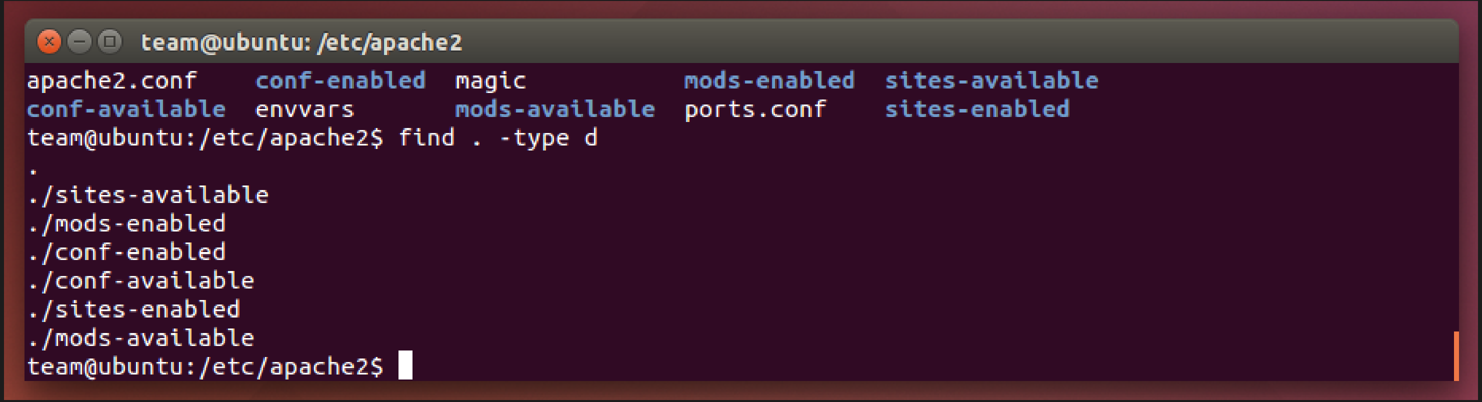
Apache

Lets take a look at apache, the most popular web server on the planet, you’ll be using it for your production environment to host Django on

Apache structure

- Apache installs to /etc/ on Ubuntu
- Consists of modules and configuration files

- Modules provide add-on functionality
- Configuration files select directives from modules that customize server functionality


team@ubuntu: /etc/apache2

```
apache2.conf     conf-enabled  magic           mods-enabled  sites-available
conf-available   envvars       mods-available  ports.conf    sites-enabled
team@ubuntu:/etc/apache2$ find . -type d
.
./sites-available
./mods-enabled
./conf-enabled
./conf-available
./sites-enabled
./mods-available
team@ubuntu:/etc/apache2$ 
```

Apache has modules 'mods'

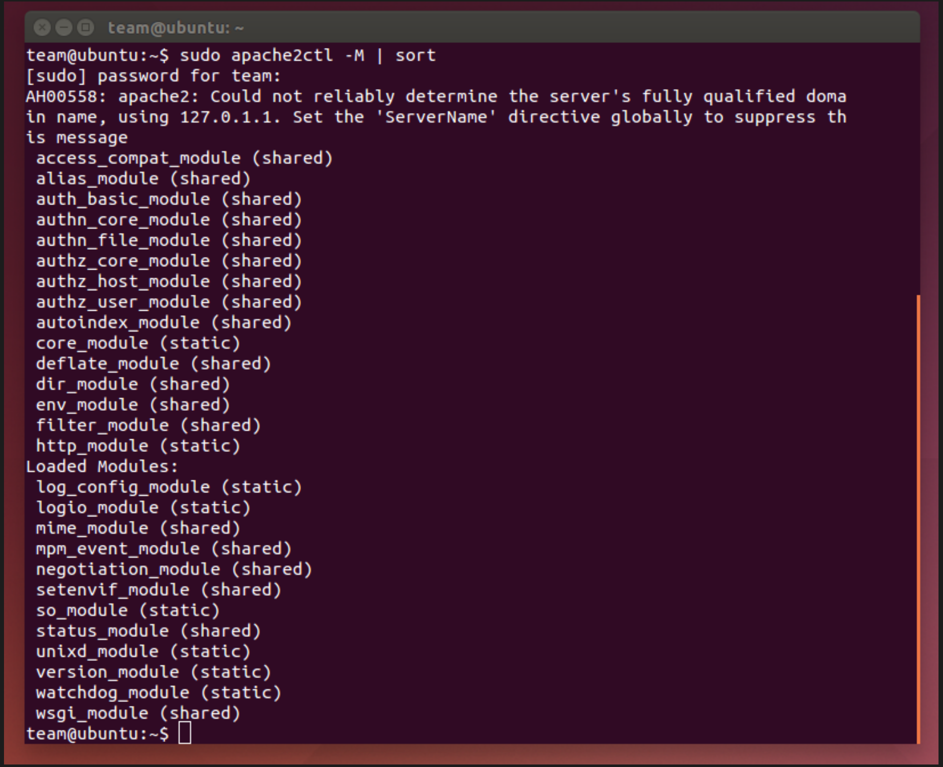
A
p
a
c
h
e

Enabled mods are stored in /etc/apache2/mods_enabled

They can be listed using
sudo apache2ctl -M | sort

Modules have configurable directives that allow functionality to be turned on and off

A full list of mods can be found at
http://httpd.apache.org/docs/2.2/mod/ (for 2.2)
Or
http://httpd.apache.org/docs/2.4/mod/ (for 2.4)



```
team@ubuntu:~$ sudo apache2ctl -M | sort
[sudo] password for team:
AH00558: apache2: Could not reliably determine the server's fully qualified doma
in name, using 127.0.1.1. Set the 'ServerName' directive globally to suppress th
is message
 access_compat_module (shared)
 alias_module (shared)
 auth_basic_module (shared)
 authn_core_module (shared)
 authn_file_module (shared)
 authz_core_module (shared)
 authz_host_module (shared)
 authz_user_module (shared)
 autoindex_module (shared)
 core_module (static)
 deflate_module (shared)
 dir_module (shared)
 env_module (shared)
 filter_module (shared)
 http_module (static)
Loaded Modules:
 log_config_module (static)
 logio_module (static)
 mime_module (shared)
 mpm_event_module (shared)
 negotiation_module (shared)
 setenvif_module (shared)
 so_module (static)
 status_module (shared)
 unixd_module (static)
 version_module (static)
 watchdog_module (static)
 wsgi_module (shared)
team@ubuntu:~$
```

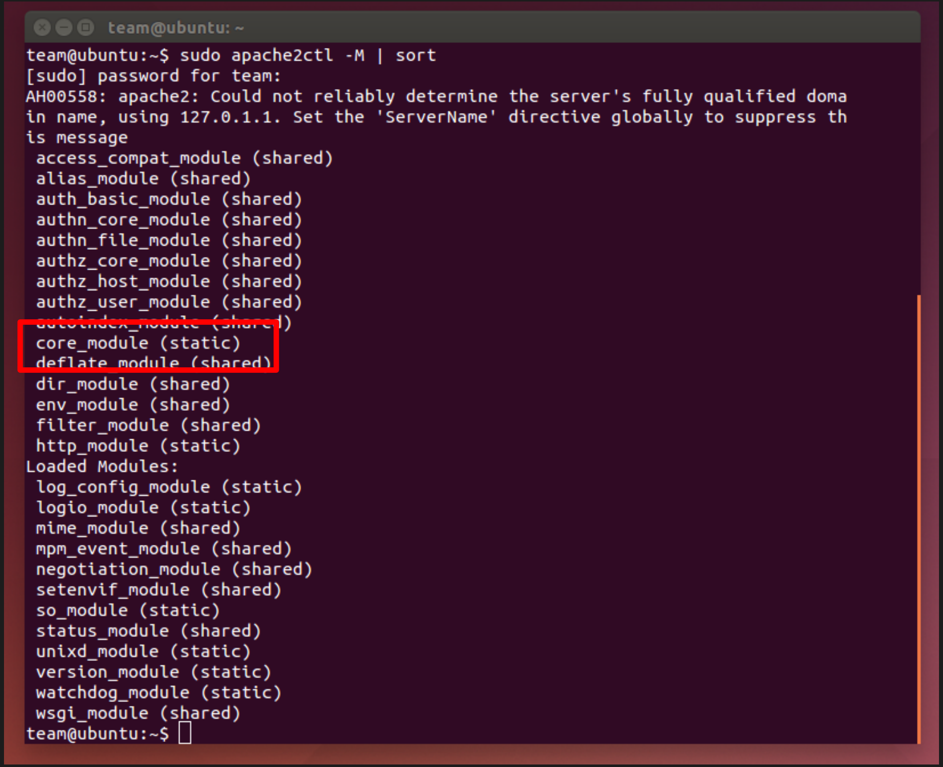
Core modules

Apache

Core modules include ‘core’, mpm_common and others

Core docs can be found at http://httpd.apache.org/docs/2.4/mod/core.html

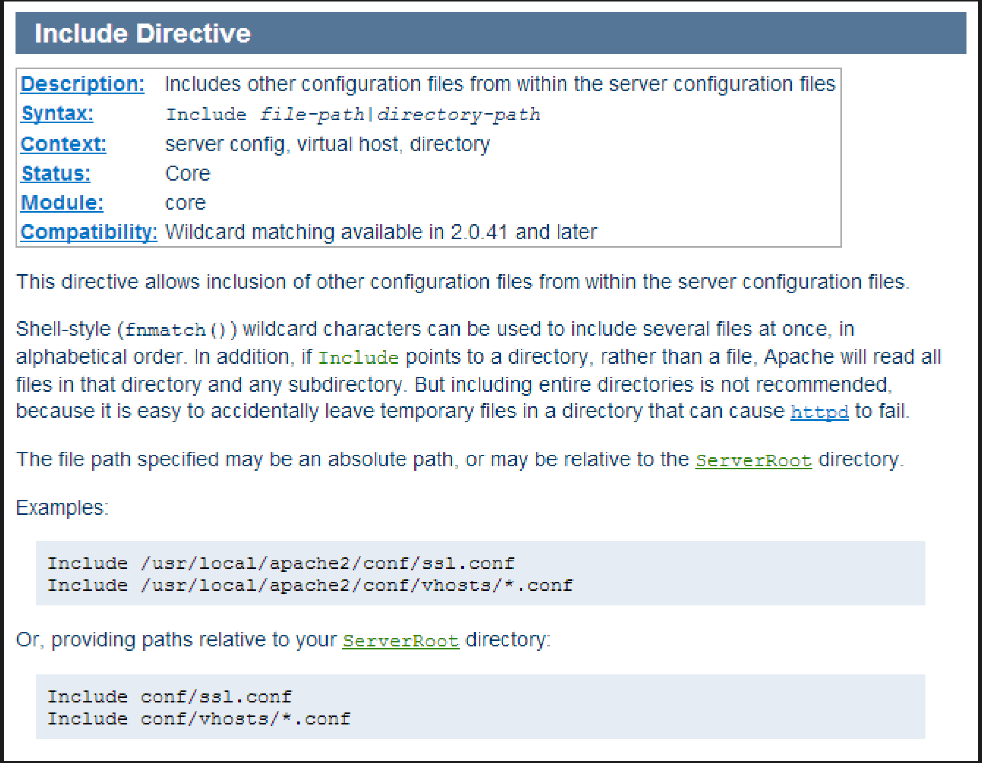
Core provides the main functionality used by the server including directives: ‘include’, ‘keepalive’, ‘files’, and ‘directory’



```
team@ubuntu:~$ sudo apache2ctl -M | sort
[sudo] password for team:
AH00558: apache2: Could not reliably determine the server's fully qualified doma
in name, using 127.0.1.1. Set the 'ServerName' directive globally to suppress th
is message
 access_compat_module (shared)
 alias_module (shared)
 auth_basic_module (shared)
 authn_core_module (shared)
 authn_file_module (shared)
 authz_core_module (shared)
 authz_host_module (shared)
 authz_user_module (shared)
 autoindex_module (shared)
 core_module (static)
 deflate_module (shared)
 dir_module (shared)
 env_module (shared)
 filter_module (shared)
 http_module (static)
Loaded Modules:
 log_config_module (static)
 logio_module (static)
 mime_module (shared)
 mpm_event_module (shared)
 negotiation_module (shared)
 setenvif_module (shared)
 so_module (static)
 status_module (shared)
 unixd_module (static)
 version_module (static)
 watchdog_module (static)
 wsgi_module (shared)
team@ubuntu:~$ 
```

A
p
a
c
h
e

Core modules: include



Include Directive

Description:	Includes other configuration files from within the server configuration files
Syntax:	`Include file-path|directory-path`
Context:	server config, virtual host, directory
Status:	Core
Module:	core
Compatibility:	Wildcard matching available in 2.0.41 and later

This directive allows inclusion of other configuration files from within the server configuration files.

Shell-style (`fnmatch()`) wildcard characters can be used to include several files at once, in alphabetical order. In addition, if `Include` points to a directory, rather than a file, Apache will read all files in that directory and any subdirectory. But including entire directories is not recommended, because it is easy to accidentally leave temporary files in a directory that can cause `httpd` to fail.

The file path specified may be an absolute path, or may be relative to the `ServerRoot` directory.

Examples:

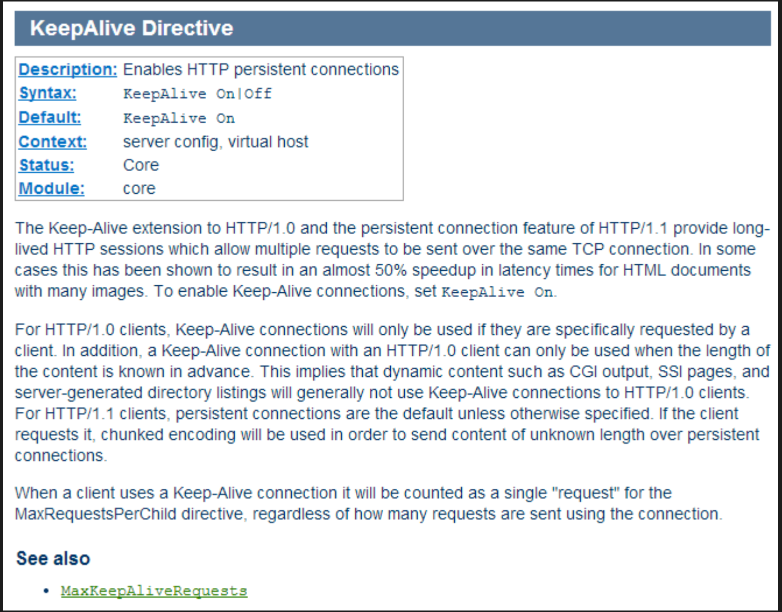
```
Include /usr/local/apache2/conf/ssl.conf
Include /usr/local/apache2/conf/vhosts/*.conf
```

Or, providing paths relative to your `ServerRoot` directory:

```
Include conf/ssl.conf
Include conf/vhosts/*.conf
```

Core modules: keepalive / loglevel

A
p
a
c
h
e



KeepAlive Directive

Description:	Enables HTTP persistent connections
Syntax:	`KeepAlive On|Off`
Default:	`KeepAlive On`
Context:	server config, virtual host
Status:	Core
Module:	core

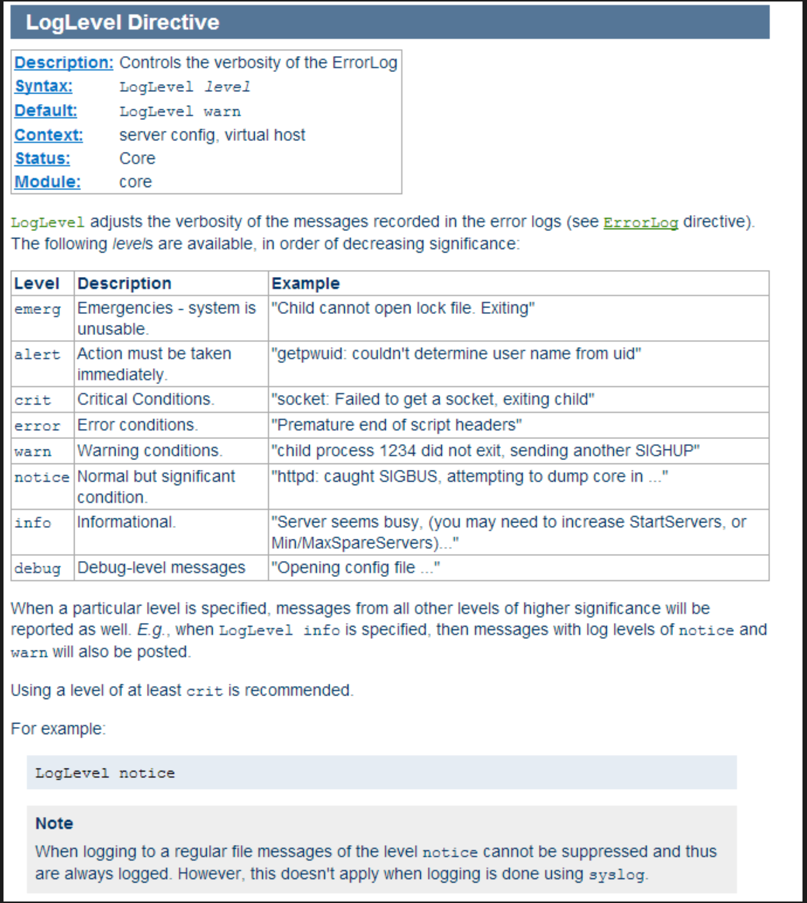
The Keep-Alive extension to HTTP/1.0 and the persistent connection feature of HTTP/1.1 provide long-lived HTTP sessions which allow multiple requests to be sent over the same TCP connection. In some cases this has been shown to result in an almost 50% speedup in latency times for HTML documents with many images. To enable Keep-Alive connections, set `KeepAlive On`.

For HTTP/1.0 clients, Keep-Alive connections will only be used if they are specifically requested by a client. In addition, a Keep-Alive connection with an HTTP/1.0 client can only be used when the length of the content is known in advance. This implies that dynamic content such as CGI output, SSI pages, and server-generated directory listings will generally not use Keep-Alive connections to HTTP/1.0 clients. For HTTP/1.1 clients, persistent connections are the default unless otherwise specified. If the client requests it, chunked encoding will be used in order to send content of unknown length over persistent connections.

When a client uses a Keep-Alive connection it will be counted as a single "request" for the MaxRequestsPerChild directive, regardless of how many requests are sent using the connection.

See also

- `MaxKeepAliveRequests`



LogLevel Directive

Description:	Controls the verbosity of the ErrorLog
Syntax:	`LogLevel` *`level`*
Default:	`LogLevel warn`
Context:	server config, virtual host
Status:	Core
Module:	core

`LogLevel` adjusts the verbosity of the messages recorded in the error logs (see `ErrorLog` directive). The following *level*s are available, in order of decreasing significance:

Level	Description	Example
`emerg`	Emergencies - system is unusable.	"Child cannot open lock file. Exiting"
`alert`	Action must be taken immediately.	"getpwuid: couldn't determine user name from uid"
`crit`	Critical Conditions.	"socket: Failed to get a socket, exiting child"
`error`	Error conditions.	"Premature end of script headers"
`warn`	Warning conditions.	"child process 1234 did not exit, sending another SIGHUP"
`notice`	Normal but significant condition.	"httpd: caught SIGBUS, attempting to dump core in ..."
`info`	Informational.	"Server seems busy, (you may need to increase StartServers, or Min/MaxSpareServers)..."
`debug`	Debug-level messages	"Opening config file ..."

When a particular level is specified, messages from all other levels of higher significance will be reported as well. *E.g.*, when `LogLevel info` is specified, then messages with log levels of `notice` and `warn` will also be posted.

Using a level of at least `crit` is recommended.

For example:

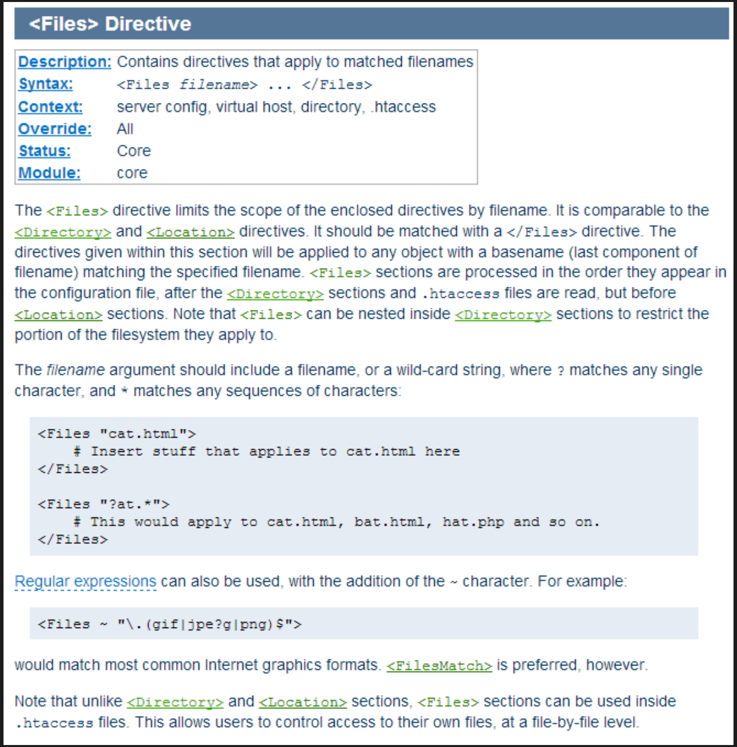
```
LogLevel notice
```

Note

When logging to a regular file messages of the level `notice` cannot be suppressed and thus are always logged. However, this doesn't apply when logging is done using `syslog`.

Core modules: files / directory

A
p
a
c
h
e



<Files> Directive

Description:	Contains directives that apply to matched filenames
Syntax:	`<Files filename> ... </Files>`
Context:	server config, virtual host, directory, .htaccess
Override:	All
Status:	Core
Module:	core

The `<Files>` directive limits the scope of the enclosed directives by filename. It is comparable to the `<Directory>` and `<Location>` directives. It should be matched with a `</Files>` directive. The directives given within this section will be applied to any object with a basename (last component of filename) matching the specified filename. `<Files>` sections are processed in the order they appear in the configuration file, after the `<Directory>` sections and `.htaccess` files are read, but before `<Location>` sections. Note that `<Files>` can be nested inside `<Directory>` sections to restrict the portion of the filesystem they apply to.

The *filename* argument should include a filename, or a wild-card string, where `?` matches any single character, and `*` matches any sequences of characters:

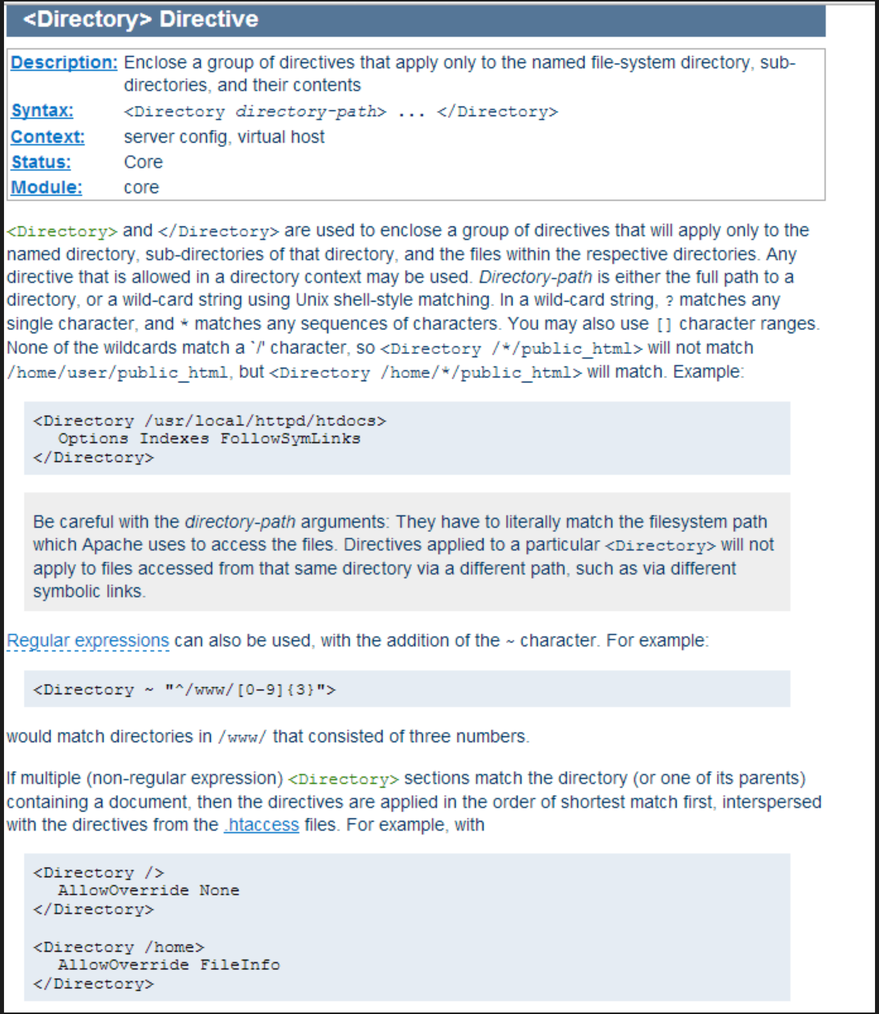
```
<Files "cat.html">
    # Insert stuff that applies to cat.html here
</Files>

<Files "?at.*">
    # This would apply to cat.html, bat.html, hat.php and so on.
</Files>
```

Regular expressions can also be used, with the addition of the `~` character. For example:

```
<Files ~ "\.(gif|jpe?g|png)$">
```

would match most common Internet graphics formats. `<FilesMatch>` is preferred, however.

Note that unlike `<Directory>` and `<Location>` sections, `<Files>` sections can be used inside `.htaccess` files. This allows users to control access to their own files, at a file-by-file level.



<Directory> Directive

Description:	Enclose a group of directives that apply only to the named file-system directory, sub-directories, and their contents
Syntax:	`<Directory directory-path> ... </Directory>`
Context:	server config, virtual host
Status:	Core
Module:	core

`<Directory>` and `</Directory>` are used to enclose a group of directives that will apply only to the named directory, sub-directories of that directory, and the files within the respective directories. Any directive that is allowed in a directory context may be used. *Directory-path* is either the full path to a directory, or a wild-card string using Unix shell-style matching. In a wild-card string, `?` matches any single character, and `*` matches any sequences of characters. You may also use `[]` character ranges. None of the wildcards match a `/' character, so `<Directory /*/public_html>` will not match `/home/user/public_html`, but `<Directory /home/*/public_html>` will match. Example:

```
<Directory /usr/local/httpd/htdocs>
  Options Indexes FollowSymLinks
</Directory>
```

Be careful with the *directory-path* arguments: They have to literally match the filesystem path which Apache uses to access the files. Directives applied to a particular `<Directory>` will not apply to files accessed from that same directory via a different path, such as via different symbolic links.

Regular expressions can also be used, with the addition of the `~` character. For example:

```
<Directory ~ "^/www/[0-9]{3}">
```

would match directories in `/www/` that consisted of three numbers.

If multiple (non-regular expression) `<Directory>` sections match the directory (or one of its parents) containing a document, then the directives are applied in the order of shortest match first, interspersed with the directives from the .htaccess files. For example, with

```
<Directory />
  AllowOverride None
</Directory>

<Directory /home>
  AllowOverride FileInfo
</Directory>
```

Apache security step 1: limit access to filesystem

Two concepts:
file system (<Directory> / <Files> directives)
web space (<Location> directives)

Apache provides the mapping

Apache security tip 1: limit access to filesystem

Use 'Order' and 'Allow' / 'Deny' commands in mod_authz_host
e.g.

```
Order Deny, Allow
Deny from all
Allow from unomaha.edu
```

Would deny all by default and allow only requests from unomaha.edu domain

This is equivalent to Require in 2.4 e.g.

```
Require host unomaha.edu
```

Allow,Deny
First, all Allow directives are evaluated; at least one must match, or the request is rejected. Next, all Deny directives are evaluated. If any matches, the request is rejected. Last, any requests which do not match an Allow or a Deny directive are denied by default.

Deny,Allow
First, all Deny directives are evaluated; if any match, the request is denied unless it also matches an Allow directive. Any requests which do not match any Allow or Deny directives are permitted.

Replaced by Require in 2.4

Apache security tip 1: limit access to filesystem

Use in combination with <Location> or <Directory>

Require and Order Allow/Deny work within the scope of a Location or Directory to limit access to a URL or set of files on the file system respectively.

e.g.

```
<Directory “/var/www/api”>
      Require all denied #deny all access to files in /var/www/api/
</Directory>
```

Or

```
<Location “/api/>
     Require all denied #deny access to the url /api
</Location>
```

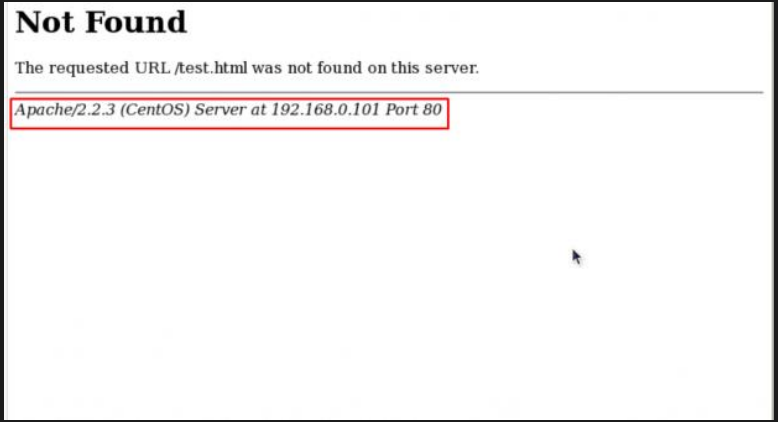
Apache security tip 2: hide apache version and OS identy from 404 errors

A
p
a
c
h
e

Use ServerSignature and ServerTokens directives

e.g.

```
ServerSignature Off
ServerTokens Prod
```



Not Found

The requested URL /test.html was not found on this server.

Apache/2.2.3 (CentOS) Server at 192.168.0.101 Port 80

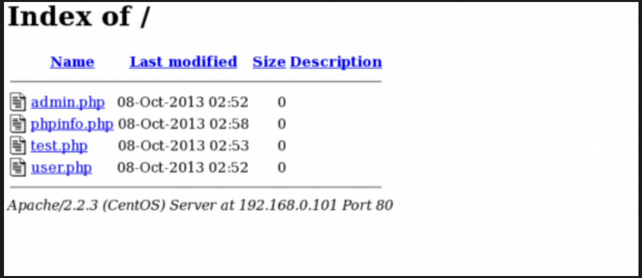
Apache security tip 3: turn off index for sites without an index.html file

A
p
a
c
h
e

Use options -indexes and in <Directory>

e.g.

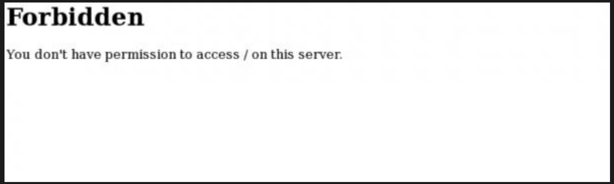
```
<Directory /var/www>
   Options -Indexes
</Directory>
```



Index of /

```
 Name        Last modified      Size Description
-------------------------------------------------
 admin.php   08-Oct-2013 02:52    0
 phpinfo.php 08-Oct-2013 02:58    0
 test.php    08-Oct-2013 02:53    0
 user.php    08-Oct-2013 02:52    0
-------------------------------------------------
Apache/2.2.3 (CentOS) Server at 192.168.0.101 Port 80
```

$\longrightarrow$



Forbidden

You don't have permission to access / on this server.

Apache security tip 4: disable modules and update regularly

A
p
a
c
h
e

Edit apache2.conf and comment out unused modules (# symbol in front of LoadModule directive)

Update apache to newest version regularly using apt-get or other package manager

Use the command apache2 –v to list the current version

Apache security tip 5: create apache user and run server as non-root

This is done by default with apt-get installation

Default user/group is www-data/www-data
This means all files must be accessible to www-data for apache to properly serve them as web content.

It also limits malicious escalation of privilege attempts.
Note that sudo service apache2 restart DOES NOT give apache processes root – it only uses root for the process spawner to bind to port 80 (and others). The actual processes that run with a user session are run using the apache user

Apache security tip 6: use mod_security and mod_evasive*

A
p
a
c
h
e

mod_security (protects against brute force attacks and allows traffic monitoring)

```
sudo apt-get install libapache2-mod-
security
sudo a2enmod mod-security
sudo /etc/init.d/apache2 force-reload
```

mod_evasive* (third party library for preventing DDOS by limiting page requests to a few times per second, limiting concurrent requests, and temporarily blacklisting offending IPs)

```
sudo apt-get install apache2-utils
cd /usr/src
wget http://www.zdziarski.com/blog/wp-
content/uploads/2010/02/mod_evasive_1.10.1.tar.gz
tar xzf mod_evasive_1.10.1.tar.gz
cd mod_evasive
apxs2 -cia mod_evasive20.c
sudo nano /etc/apache2/apache2.conf
```

Add in:

```
 LoadModule evasive20_module /usr/lib/httpd/modules/mod_evasive20.so
<IfModule mod_evasive20.c>
    DOSHashTableSize 3097
    DOSPageCount 2
    DOSSiteCount 50
    DOSPageInterval 1
    DOSSiteInterval 1
    DOSBlockingPeriod 60
    DOSEmailNotify <someone@somewhere.com>
</IfModule>
```

Save and then:

```
sudo /etc/init.d/apache2 restart
```

A
p
a
c
h
e

Apache security tip 6: use mod_security and

Mod security needs rules to work – by default it doesn't do much

```
nano /etc/modsecurity/modsecurity.conf
Find SecRuleEngine DetectionOnly and change to SecRuleEngine On

Then set directives like
SecRequestBodyLimit
SecRequestBodyNoFilesLimit
SecRequestBodyLimit
```

and others to limit or restrict requests -
This can help prevent denial of service attacks by rejecting certain types of requests that often signal DOS.
OWASP Has a set of best practices for mod security defined
See https://github.com/SpiderLabs/owasp-modsecurity-crs/

You can load these rules into mod security using:

```
sudo mkdir /etc/apache2/crs
sudo cd /etc/apache2/crs
sudo wget https://github.com/SpiderLabs/owasp-modsecurity-crs/tarball/master
sudo tar -xvf master
sudo  mv SpiderLabs-owasp-modsecurity-crs-* owasp-modsecurity-crs
sudo cd  /etc/apache2/crs/owasp-modsecurity-crs/
```

```
sudo nano /etc/apache2/conf/apache.conf
Add in:
<IfModule security2_module>
   Include /etc/apache2/crs/owasp-modsecurity-crs/modsecurity_crs_10_setup.conf
   Include /etc/apache2/crs/owasp-modsecurity-crs/base_rules/*.conf
</IfModule>
sudo service apache2 restart
sudo nano /etc/apache2/modsecurity.d/mod_security.conf
Add in
<IfModule mod_security2.c>
   SecRuleEngine On
   SecRequestBodyAccess On
   SecResponseBodyAccess On
   SecResponseBodyMimeType text/plain text/html text/xml application/octet-stream
   SecDataDir /tmp
</IfModule>
sudo service apache2 reload
sudo service apache2 restart
```

Apache security tip 7: limit request size, timeouts, request fields, and maxclients

A
p
a
c
h
e

Set timeout directive (Amount of time server waits before it fails (500 error). Default is 300 secs)

```
Timeout X
```

Set Maxclients/MaxRequestWorkers directive (Number of connections to be served simultaneously. Default is 256)

```
MaxClients Y (Apache 2.2) MaxRequestWorkers Y (Apache 2.4)
```

Set keepalivetimeout directive (Amount of time server will wait for subsequent request before closing connection. Default is 5 secs)

```
KeepAliveTimeout Z
```

Set LimitRequestFields directive (Number of fields accepted from client http headers. Default is 100 http header fields)

```
LimitRequestFields A
```

Set LimitRequestBody directive (Size of HTTP request accepted in bytes. Can be up to 2GB)

```
LimitRequestBody B
```

Apache security step 8: enable logging

A
p
a
c
h
e

Use ErrorLog and Custom Log directives (should be enabled by default, buy level can be set as)

```
LogLevel X
```

Level	Description	Example
emerg	Emergencies - system is unusable.	"Child cannot open lock file. Exiting"
alert	Action must be taken immediately.	"getpwuid: couldn't determine user name from uid"
crit	Critical Conditions.	"socket: Failed to get a socket, exiting child"
error	Error conditions.	"Premature end of script headers"
warn	Warning conditions.	"child process 1234 did not exit, sending another SIGHUP"
notice	Normal but significant condition.	"httpd: caught SIGBUS, attempting to dump core in ..."
info	Informational.	"Server seems busy, (you may need to increase StartServers, or Min/MaxSpareServers)..."
debug	Debug-level messages	"Opening config file ..."

A
p
a
c
h
e

Apache security tip 9: Use HTTPS/SSL

1. Use an online certificate authority (CA) or create a self-signed cert
2. Copy certification files to your server (.crt) – either self-signed (testing) or from a CA
3. Edit your .conf file to include an SSL block, e.g.

   ```
   <VirtualHost 192.168.0.1:443>
                      DocumentRoot /var/www/
                      SSLEngine on
                      SSLCertificateFile /path/to/your_domain_name.crt
                      SSLCertificateKeyFile /path/to/your_private.key
                      SSLCertificateChainFile /path/to/intermediate_CA_cert.crt
   </VirtualHost>
   ```

   for 2.4 change SSLCertificateChainFile to SSLCACertificateFile
4. sudo service apache2 reload
5. sudo service apache2 restart

Apache is just the http server.
What about the web framework?

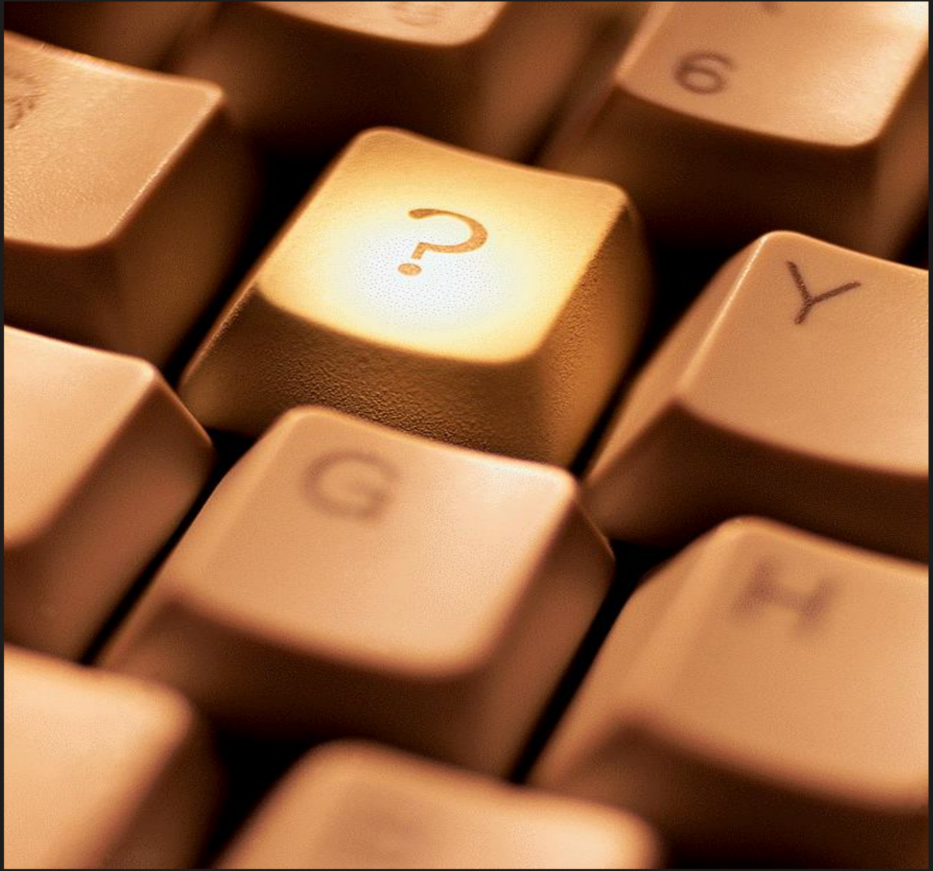
Now you have some basic server-level protections in place. You still need application-level protections. The most important ones are filtering inputs – we’ve already talked about Django protections.

Next Next Time

Keep working on Project 2 (next iteration of your app)

I should have Project 1 graded or nearly graded

Questions?



Matt Hale, PhD

University of Nebraska at Omaha

Interdisciplinary Informatics

mlhale@unomaha.edu

Twitter: @mlhale_