# 4.3 Apache Security Hardening

There are lots of way to increase the security on the Apache Web Server and there is no one size fits all. Some hardening steps will beak specific needs.

But before you start, as advised in the Apache Setup, ensure Apache works with its intended integrated purpose in a test environment. Verify **vanilla** and then basic hardening first. Then carefully apply each security setting and test at intervals.

Use the Center for Internet Security Security and search "apache benchmark" and look for your version of Apache to get hardening documentation.

I'll create my own abridged version in the future with additional perspective of impact to web applications.



# Apache Basic Server Hardening

Before hardening your Web Server you should make sure it works with it's intended integrated purpose in a test environment. Otherwise you may spend lots of wasted time trouble-shooting.

So, assuming that your Web Server passes testing of it's intended purpose, you may perform "Basic Hardening". Because this is "basic" I often perform these all at once and then test.

Here are some of the basic hardening steps I take today by default,

> As with any security notes, I will write a disclaimer that there are more advanced ways to secure Apache. You can go as far as compiling your own custom version but that's out of scope for now.

## Disable Server Information Banner

By default Apache provides extra information about your server when 403, 404, 502 or similar error pages are invoked. The information could be used to look up vulnerabilities on the particular version of Apache you are running.

If you visit a page that does not exist you will invoke a 404 error resulting in a page Not Found similar to below,

```
Not Found

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

Apache/2.4.18 (Ubuntu) Server at www.bonsaiframework.com Port 80
```

Edit **/etc/apache2/conf-available/security.conf**,

set ServerTokens Prod - This turns off all the extra header information sent by Apache.

set ServerSignatures Off - Removes footer information from default apache pages. For example, page not found.

> Older versions of Apache use /etc/apache2/conf.d/security

Restart Apache to take effect and verify by invoking a 404 again.

## Disabling Unnecessary Modules

Less loaded, less vulnerabilities and you will also get performance increases too.

### Disable Status Module

I found that you can save about 3MB of memory if the status apache module is disabled. Here's how to disable interactively,

```
sudo a2dismod
Your choices are: alias auth_basic authn_file authz_default authz_groupfile
authz_host authz_user autoindex cgid
                 deflate dir env filter jk mime negotiation proxy
proxy_http rewrite setenvif status substitute
Which module(s) do you want to disable (wildcards ok)?
NOTE: make sure you only disable the following one ONLY!!! type:
status
Module status disabled.
To activate the new configuration, you need to run:
  service apache2 restart
sudo service apache2 restart
```

### Disable More Modules

Will flush this out some more ...

## Turn off Default Website

...

# Intermediate Hardening

These items are best practice and but may impact integrated modules. If there are any known impacts, I will document them here too.

## Setup Cross-Frame Scripting Policy in Header

Control what external domains iframes can communicate to on your your website.

**Risk** - internally developers may introduce iframe code to subvert your website or collect sensitive data. Externally, if your application is open ton an injection attack, a malicious iframe may be placed on your website.

**Possible Impact** - The success of this policy is dependant have a proper inventory of external domains used by iframes.

**Considerations** - If the website is an application, you may want to use code to set headers instead of using the web server.

## Setup CORS Policy in Header

CORS (Cross-Origin Resource Sharing) allows a domain to set policies to control if resources on the server can be access by a web page hosted on a different domain.

**Risk** - an overly permissive CORS can allow a malicious application leverage assets on your website  leading to spoofing, data theft, relay and other attacks.

## Disable Etag Header

Etag (entity tag) was introduced to help improve caching. However, besides **not** being very effective in an enterprise clusters environments), it also provides sensitive information like inode number, multipart MIME boundary and child processes. It allows hackers to uniquely identify a particular server.

Unless you have a compelling reason you may,

1. Disable etag - rely on the default Expire or Cach-Control header information.
2. Minimally disable INode

... to fill out

## Advanced Hardening

The advanced hardening is more likely to break your system so apply carefully one step at a time.

...