Apache

Chris Wilson, AfNOG 2016

About this presentation

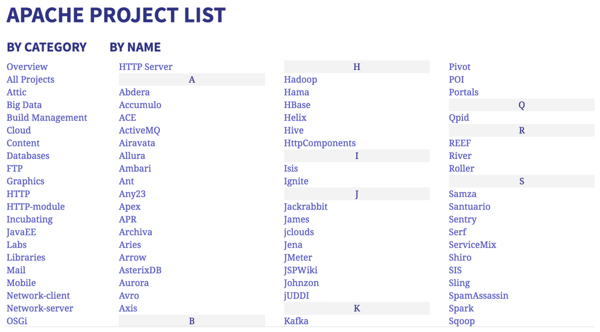
Based on a previous talk by Joel Jaeggli with thanks!

You can access this presentation at:

- Online: http://afnog.github.io/sse/apache/
- Local: http://www.ws.afnog.org/afnog2016/sse/apache/index.html
- Github: https://github.com/afnog/sse/blob/master/apache/presentation.md
- Download PDF: http://www.ws.afnog.org/afnog2016/sse/apache/presentation.pdf

What is Apache?

- An HTTP server (web server)
- A foundation supporting several web-related software projects



APACHE PROJECT LIST

BY CATEGORY	BY NAME		
Overview	HTTP Server	H	Pivot
All Projects	A	Hadoop	POI
Attic	Abdera	Hama	Portals
Big Data	Accumulo	HBase	Q
Build Management	ACE	Helix	Qpid
Cloud	ActiveMQ	Hive	R
Content	Airavata	HttpComponents	REEF
Databases	Allura	I	River
FTP	Ambari	Isis	Roller
Graphics	Ant	Ignite	S
HTTP	Any23	J	Samza
HTTP-module	Apex	Jackrabbit	Santuario
Incubating	APR	James	Sentry
JavaEE	Archiva	jclouds	Serf
Labs	Aries	Jena	ServiceMix
Libraries	Arrow	JMeter	Shiro
Mail	AsterixDB	JSPWiki	SIS
Mobile	Aurora	Johnzon	Sling
Network-client	Avro	jUDDI	SpamAssassin
Network-server	Axis	K	Spark
OSGi	B	Kafka	Sqoop

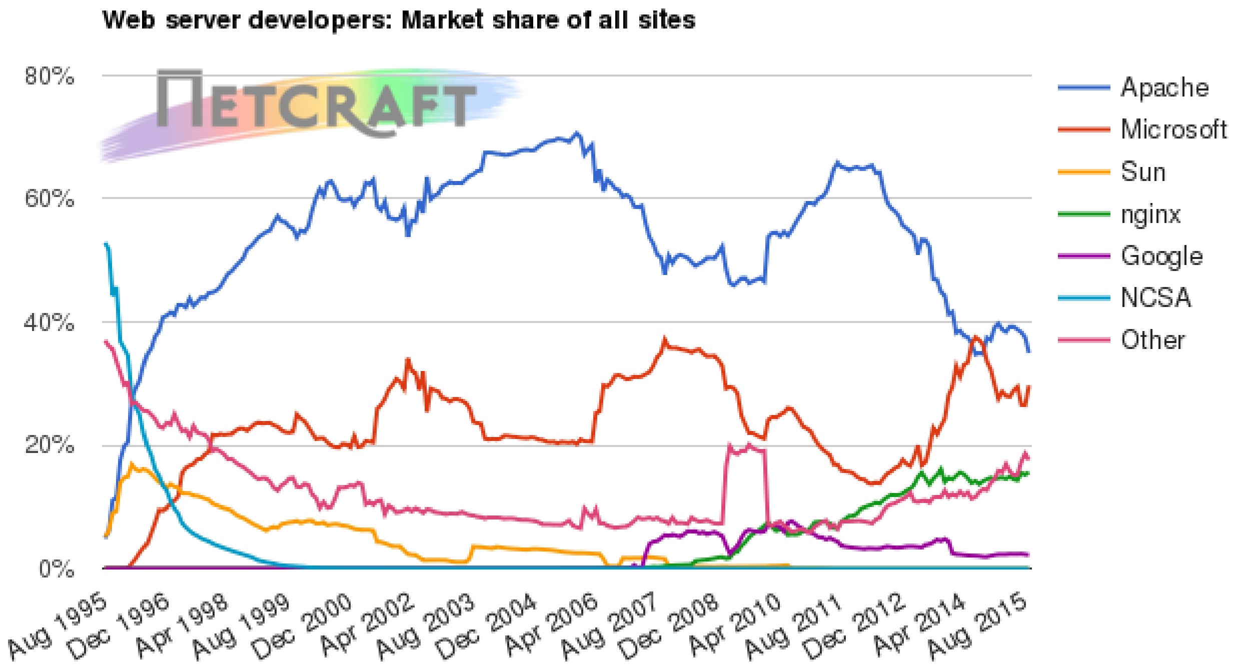
For clarity it might help to talk about "Apache Server" to mean the HTTPD server.

Other HTTP servers

What other HTTP (web) servers are commonly used?

Other HTTP servers

What other HTTP (web) servers are commonly used?



Which one to use?

- Apache
    - Popular, well-documented, flexible, secure, big, slow, heavy, PHP integration.
- Nginx
    - Increasingly popular, quite well-documented, very fast, reverse proxy, SSL support/wrapper, no PHP.
- Lighttpd
    - Simple, fast, no PHP.
- Thttpd
    - Tiny, fast, no PHP.

Apache Features

- Server Side Programming Language Support
    - Apache supports some common language interfaces which include Perl, Python, Tcl, and PHP. It also supports a variety of popular authentication modules like mod_auth, mod_access, mod_digest and many others.
- IPv6 Support
    - On systems where IPv6 is supported by the underlying Apache Portable Runtime library, Apache gets IPv6 listening sockets by default.
- Virtual Hosting
    - Apache will allow one installation instance to serve multiple websites. For instance one Apache installation can serve sse.afnog.org, ws.afnog.org etc
- Simplified configuration (!)

More at: http://httpd.apache.org/docs/2.2/new_features_2_0.html

Virtual Hosting

What does it mean?

Apache support virtual hosting (deciding which website to display) using:

- Name based virtual hosts (`Host` header)
- IP/Port based virtual hosts
- Aliases (subdirectories)

PHP and MySQL

- Many web applications written in PHP and using a MySQL database.
- Relatively easy to deploy under Apache (and most web hosting).
- We will install the necessary software shortly.

Apache and SSL

- SSL is the "Secure Socket Layer"
  - Used to secure several protocols including HTTP
  - When used properly, protects the wrapped protocol from
  - Usually the wrapped protocol has little or no interaction with SSL layer (transparent)
  - This causes problems with virtual hosting!
- HTTPS (HTTP over SSL) runs on port 443 by convention
  - Each SSL-wrapped service runs on a different port than its non-SSL-wrapped version

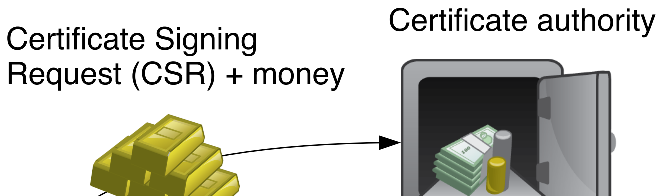
SSL Certificates

- Certificates identify parties (servers and sometimes clients)
    - SSL useless without server auth - why not?
- Need to generate a Certificate Signing Request (CSR) and get someone to sign it
    - Chain of trust, established by signatures
    - Signer needs to be trusted by web browser (directly or indirectly)
- Each SSL certificate* has a Public and Private key
    - The public key is used to encrypt the information
    - The public key is accessible to everyone
    - The private Key is used to decipher the information
    - The private should be not be disclosed to anyone

* The key is included on the certificate, but can be reused on more certificates

as long as not compromised. There is no way to revoke it except to revoke all certs signed with it.

How SSL works



Certificate Authorities

Who are these guys anyway?

- Geotrust, Go Daddy, RSA, Thawte, Verisign, many others…
- Trusted by browsers
- Verify your identity (not really any more)
- Take your money
- Try not to lose their private keys
    - What would happen if they did?

Self-signed certificates

- Useful for testing
- Useful in controlled environments
- Free (as in beer, but take time and skill to manage)
- Useless for clients who won't install the cert

Getting Certificates

So how do I get one again?

- Pay money
- Self-certified (own CA)
- Self-signed

We will use a self-signed certificate in order to proceed quickly. There are tutorials on the Internet on running your own CA with OpenSSL (it's not that hard, really).

Install Apache

```
sudo apt install apache2
```

How do you test that it worked?

Install Apache

```
sudo apt install apache2
```

How do you test that it worked?

```
telnet localhost 80
```

And visit http://pcXX.sse.ws.afnog.org in your browser.

What content is it serving? How do we change it?

Enable and test IPv6

Set your IPv6 address to match your IPv4 address (replace `xx` with your PC number plus 100):

```
$ sudo ip -6 addr add 2001:43f8:220:219::XX/64 dev eth0
```

Then add your default route for IPv6:

```
$ sudo ip -6 route add default via 2001:43f8:220:219::1
```

On the above if you get the error message `RTNETLINK answers: File exists`, it means that the gateway is already in place, as it was auto-configured.

Test your IPv6 connectivity:

```
$ ping6 www.google.com
```

Then browse your IPv6 address at http://[2001:43f8:220:219::XX] (the square brackets are deliberate and essential!).

Apache configuration files

```
* /etc
  * /apache2
    * apache2.conf
    * ports.conf
    * conf-available
      * *.conf
    * conf-enabled
      * symlinks to mods-available for services which are enabled
    * mods-available (and mods-enabled)
      * *.load
      * *.conf
    * sites-available (and sites-enabled)
      * 000-default.conf
      * default-ssl.conf
* /var/www/html (content)
```

```
  * index.html (the test page)
```

Why this structure?

Enabled sites and modules

- `mods-available` and `sites-available` allows packages to ship default configuration files
  - **without** them being enabled automatically
  - more secure than Red Hat/CentOS system
- Enable and disable with commands:
  - `a2enmod` and `a2dismod`: modules (mods)
  - `a2ensite` and `a2dissite`: sites
  - `a2enconf` and `a2disconf`: configuration files (confs)

Which sites, modules and confs are enabled by default, and which are not?

Starting Apache

- Startup scripts are located in `/etc/init.d/`
  - `/etc/init.d/apache2 start`
  - `service apache2 start`
- Other useful commands:
  - `/etc/init.d/apache2 stop`
  - `/etc/init.d/apache2 restart` (stop+start)
  - `/etc/init.d/apache2 reload` (graceful reload config)

Install MySQL and PHP

Install the packages:

```
$ sudo apt install mysql-server apache2 php5 php5-mysql
```

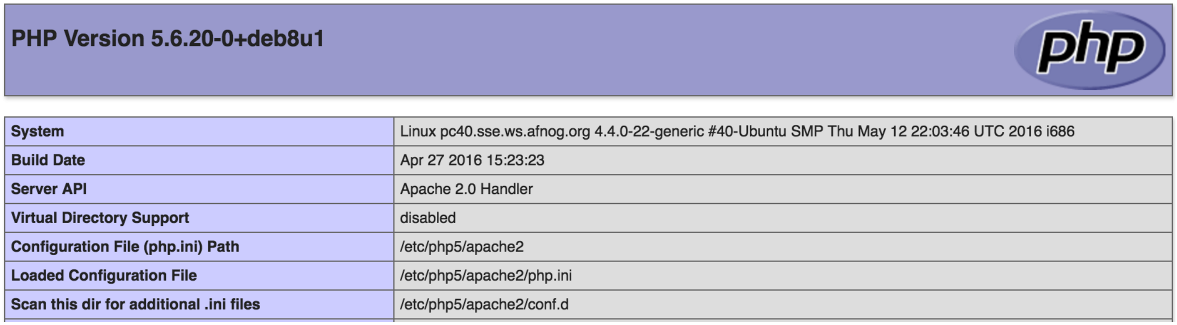
When the mysql-server prompts for a password to be entered use 'afnog' as the password. If not prompted, don't worry, we will set it later.

Testing PHP

Create the file `/var/www/html/test.php` with the following contents:

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

Load it in your browser at http://pcXX.sse.ws.afnog.org/test.php. You should see this:



PHP Version 5.6.20-0+deb8u1

System	Linux pc40.sse.ws.afnog.org 4.4.0-22-generic #40-Ubuntu SMP Thu May 12 22:03:46 UTC 2016 i686
Build Date	Apr 27 2016 15:23:23
Server API	Apache 2.0 Handler
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc/php5/apache2
Loaded Configuration File	/etc/php5/apache2/php.ini
Scan this dir for additional .ini files	/etc/php5/apache2/conf.d

Securing MySQL

Please read the instructions and use the letters "y" or “n” on the keyboard.

```
$ sudo mysql_secure_installation
```

The password for MySQL is probably `afnog` (unless you entered a different password during the installation above).

```
Enter current password for root (enter for none):
OK, successfully used password, moving on...
Remove anonymous users? [Y/n] y
 ... Success!
Disallow root login remotely? [Y/n] n
... Success!
Remove test database and access to it? [Y/n] y
Reload privilege tables now? [Y/n] y
 ... Success!
Cleaning up...
```

Testing MySQL

Log in to mysql console to check if the password was set properly using command below.

```
$ mysql -u root -p
```

```
Password:
```

Type the password at the prompt. Then you should see a `mysql>` prompt, which means that you authenticated successfully and can enter SQL commands.

You can exit from the `mysql>` prompt by typing the command `exit`.

Configuring SSL

To create a secure virtual host accessed via https rather than http, you will need to configure your Apache server to use OpenSSL for encrypting the data served from the web server.

NOTE:

- Each virtual host must have its own certificate file see comments on "CommonName".
- The "CommonName" is the FQDN in this case pcXX.sse.ws.afnog.org
- The path is where the certificate File and Keys are located, in this case `/etc/apache2/ssl`.

Configuring SSL

Create your public and private key

Generate a public and private key-pair:

```
$ sudo mkdir /etc/apache2/ssl/
```

```
$ cd /etc/apache2/ssl/
```

```
$ sudo openssl genrsa -des3 -out server.key 2048
```

NOTE: A passphrase will be requested to encrypt the key. For this exercise, use "afnog" as the pass phrase. However, this pass-phrase will be needed at every apache restart. To get rid of the passphrase prompts at every apache restart and maintain the original key, run these commands:

```
$ sudo cp server.key server.key.orig
```

```
$ sudo openssl rsa -in server.key.orig -out server.key
```

Configuring SSL

Create a Certificate Signing Request (CSR)

Use this command to generate a new Certificate Signing Request (CSR):

```
$ sudo openssl req -new -key server.key -out server.csr
```

This will prompt for some information, which will appear in the certificate.

The Common Name **must match** the hostname that you will use to access the Apache server, for example `pcXX.sse.ws.afnog.org`, where XX is your computer number.

Configuring SSL

Self-sign your certificate

Use this command to sign the certificate with the same public key (a self-signed certificate):

```
$ sudo openssl x509 -req -days 365 -in server.csr -signkey
  server.key -out server.crt
```

Enable SSL in Apache

We need to tell Apache where to find the certificate and the private key files that we want it to use.

- Edit `/etc/apache2/sites-available/default-ssl.conf`
- Find and modify the `SSLCertificateFile` and `SSLCertificateKeyFile` lines to read:

```
SSLCertificateFile    /etc/apache2/ssl/server.crt
```

```
SSLCertificateKeyFile /etc/apache2/ssl/server.key
```

- Enable the SSL module and the default SSL site:

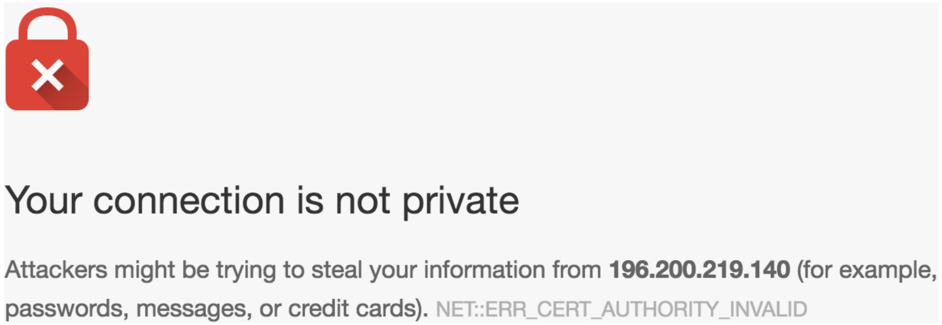
```
sudo a2enmod ssl
```

```
sudo a2ensite default-ssl
```

```
sudo service apache2 reload
```

Testing SSL

Open https://pcXX.sse.ws.afnog.org in your browser. What do you see?



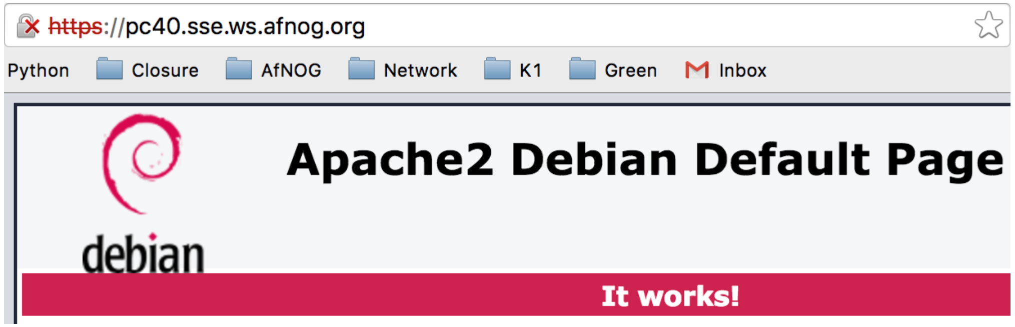
Your connection is not private

Attackers might be trying to steal your information from **196.200.219.140** (for example, passwords, messages, or credit cards). NET::ERR_CERT_AUTHORITY_INVALID

Testing SSL

You must bypass this warning to open the page.

- On Chrome: click on *Advanced* and then *Proceed to pcXX.sse.ws.afnog.org (unsafe)*.


~~https~~://pc40.sse.ws.afnog.org

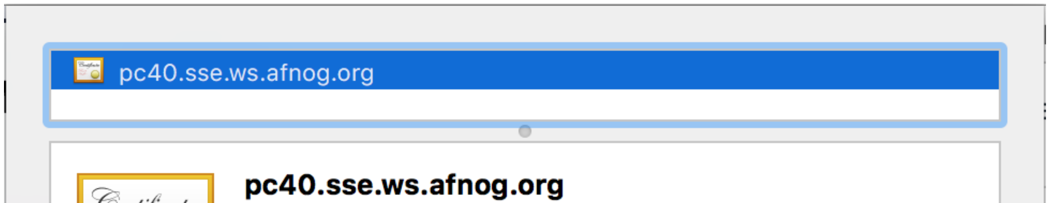
Apache2 Debian Default Page

It works!

Success! (kind of)

Testing SSL

What about the red padlock? Click on it, and then *Details* and *View Certificate* (or similar):

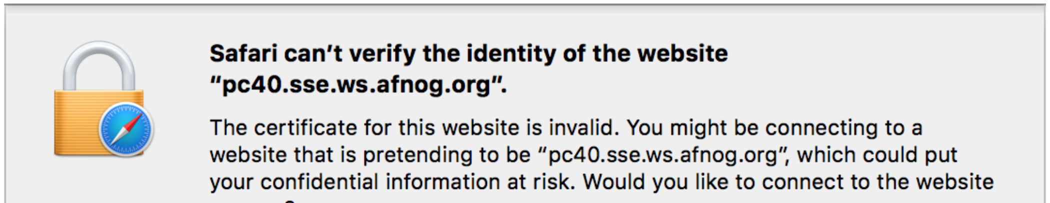


pc40.sse.ws.afnog.org

pc40.sse.ws.afnog.org

Solving the security warning

- Submit the (same) CSR to a well-known CA, or
- Install the cert in your browser's certificate store:



Safari can't verify the identity of the website "pc40.sse.ws.afnog.org".

The certificate for this website is invalid. You might be connecting to a website that is pretending to be "pc40.sse.ws.afnog.org", which could put your confidential information at risk. Would you like to connect to the website

FIN

Any questions?

(yeah, right!)