# Apache Security with SSL

pre SANOG VI Workshop

January 2005

Hervey Allen
Network Startup Resource Center



# Summary

- Apache running with mod+ssl – what is it?
- Digital certificates signed and not signed.
- How to install support for ssl in Apache:
  - Compiled from source, or
  - From a ports package
- Advantages and disadvantages of both.
- Configure your own local certificate
- Solving problems:
  - iptables
  - /var/log/httpd-error.log
  - /var/log/messages
- Summary
- Practical exercises in class




# Apache+mod_ssl – What is it?

Together Apache and mod_ssl create a system of security with digital certificates that allows you to offer secure, encrypted connections to your web server.

mod_ssl is an Apache module that adds “secure sockets layer” (ssl) and “transport layer security” (tls) between a web server and it's clients (web browsers).




# Digital certificates and signatures

If you generate a local digital certificate you can pay a signing authority to verify your certificate and they'll send it back to you with their “signature”.

With the signing authority's signature your certificate will be accepted by clients (web browsers) without requiring that they accept your certificate to create a secure connection.

A digitally signed certificate implies trust that you are who you say you are between your server and the clients who connect to it.





# Installing support for SSL with Apache

FreeBSD includes several methods for supporting Apache with ssl. We'll use mod_ssl with Apache located in /usr/ports/www.apache13-modssl.

The package generates and installs the following:

- Local digital certificates in /usr/local/etc/apache.
- mod_ssl module: /usr/local/libexec/apache/libssl.so
- Documentation, additional libraries, etc.



# Apache with ssl port vs. source install

<u>From /usr/ports</u>

- It's easy.
- Configuration (which can be hard) is already done.
- Updating the package in the future is much easier.
- You might suppose that the folks at FreeBSD have lots of experience with SSL...?



## Apache with ssl port vs. source install

Advantages of Compiling from Source

- You can specify exactly how you want to install SSL support in Apache.
- You'll learn *a lot* about SSL with Apache...
- Can anyone think of any more?



## Digital certificate pieces

Read through the README files in:

- /usr/local/etc/apache/ssl.crl
- /usr/local/etc/apache/ssl.crt
- /usr/local/etc/apache/ssl.csr
- /usr/local/etc/apache/ssl.key
- /usr/local/etc/apache/ssl.prm

mod_ssl with Apache port makefile generates a whole set of sample digital certificate files that you can use to understand how they work with http.



## Digital certificate pieces cont.

- apache/ssl.crt/server.crt
  - Public server certificate.
- apache/ssl.csr/server.csr
  - Public key plus domain to be signed by a CA. Signed version replaces ssl.crt/fn.crt public key file.
- apache/ssl.key/server.key
  - Server's private key.



## Configure a local certficate.

You can do the following steps:

- `mkdir /usr/local/etc/apache/tmp`
- `cd /usr/local/etc/apache/tmp`
- `openssl genrsa -des3 -out server.key 2048`
- `openssl rsa -in server.key -out server.pem`
- `openssl req -new -key server.key -out \ server.csr` (*answer the series of questions*)
- `openssl x509 -req -days 60 -in server.csr \ -signkey server.key -out server.crt`
- `Move files to corresponding apache/ssl.xxx/ directories.`



## Configure a certificate cont.

**Explanation**

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

generates a 2048 bit RSA key using the OpenSSL libraries. The key is encoded with the des3 (triple des) algorithm.

This key is private.



## Configure a certificate cont.

```
openssl rsa -in server.key -out server.pem
```

This removes the passphrase from the private key and places the private key in server.pem for future use.

You can replace server.key with server.pem to avoid a password prompt when you start your web server, i.e.:

- `cp apache/tmp/server.pem apache/ssl.key/server.key`



## Configuring a certificate cont.

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

This generates a “csr” so that you can have the key signed, or to generate a self-signed certificate.

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

This generates a certificate that's good for 365 days. You can make this shorter if you wish. For instance to hold you over while asking for a signed certficate from a signing authority.



## Local certificate in action

We'll take a look at a local, unsigned certficate.

You will use the default “snake oil” certificate installed by th Apache+mod_ssl port.



## Solving problems

Server connection problems?

- Check if iptables is running and blocking access to port 443.
- If the certificate is properly created.
- The configuration in /usr/local/etc/apache/httpd.conf
  - Note “ServerName” directive
- To see certificate and/or configuration file errors look in: ==>



## Solving problems cont.

See errors in:

- /var/log/messages (tail -f /var/log/messages)
- /var/log/httpd-error.log
- /var/log/ssl_engine_log
- /var/log/ssl_request_log

And, as always, you can use:

http://www.google.com/

or

http://www.freebsd.org/



## More resources

- http://www.modssl.org/
- http://www.apache.org/
- http://www.openssl.org/
- http://www.sanog.org/
- http://www.oreilly.com/ and check out the books that deal with SSL, including *Web Security, Privacy & Commerce*.




## Conclusion

- Apache with mod_ssl = “secure” web server
- Webmail services *require* https connections
- SSL/TLS creates additional cpu overhead. With many clients plan accordingly.
- Signed certificates (server.csr) are fairly inexpensive. Signing authorities in browser.
- Without a signed certificate there is a fundamental problem of trust when connecting to a server.