Apache Security with SSL

SANOG IV: IP Services Track

July 2004

Presented by 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 an RPM package
- Advantages and disadvantages of both.
- Configure our own local certificate
  - Configuration session for reference
- Solving problems:
  - iptables
  - /var/log/httpd/
  - /var/log/messages
- Summary

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

With Fedora Core 2 Apache with SSL is already installed if you choose to install Apache

Fedora Core 2 uses an RPM package for mod_ssl that is named mod_ssl-2.0.xxxx.rpm.

The packet generates and installs the following:

- Local digital certificates in /etc/httpd/conf.
- The mod_ssl module in /etc/httpd/modules.
- The configuration file /etc/httpd/conf.d/ssl.conf.

Installing SSL support cont.

Another form to install mod_ssl is to compile Apache with mod_ssl together from source.

You can download the code from:

- http://www.apache.org/
- http://www.modssl.org/

And, you can specify *many* options that you cannot do using the RPM install method (Fedora has already decided on the options for you).

Advantages and Disadvantages

RPM Package

- It's easy, easy, 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 Fedora (Red Hat) have lots of experience with SSL...?

Advantages and Disadvantages cont.

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?

Configure a local certificate

- Take a look at /etc/httpd/conf.d/ssl.conf.
- Look at the virtual server section. Line 88 in the file (in vi ":88").
- The first lines and the lines that point to the certicate files are the most interesting:
  - /etc/httpd/conf/ssl.crt/server.crt
  - /etc/httpd/conf/ssl.key/server.key

Configure a certificate cont.

- Fedora places some of the certificate components in slightly different locations, but this will permit us to run multiple virtual sites with certificates more easily.
- Now we are going to generate a certficiate. First we will backup our current Apache configuration, just in case:
  - mkdir /tmp/apache
  - cp -r /etc/httpd/conf/* /tmp/apache/.

Configure a certificate cont.

Do the following steps:

- `mkdir /etc/httpd/conf/tmp`
- `cd /etc/httpd/conf/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`

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.

We'll show why this is useful a bit later.

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.

A configuration session

Now we'll show each step with some answer you can use:

First step

```
[root@localhost tmp]# openssl genrsa -des3 -out server.key 2048
Generating RSA private key, 1024 bit long modulus
.....++++++
.............++++++
e is 65537 (0x10001)
Enter pass phrase for server.key:                [password goes here]
Verifying - Enter pass phrase for server.key:    [password goes here]
[root@localhost tmp]#
```

A configuration session cont.

Second step:

[root@localhost tmp]# openssl rsa -in server.key -out server.pem
Enter pass phrase for server.key: [password from before goes here]
writing RSA key
[root@localhost tmp]#

A configuration session cont.

Third step:

```
[root@localhost tmp]# openssl req -new -key server.key -out server.csr
Enter pass phrase for server.key:          [password from before goes here]
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [GB]:NP
State or Province Name (full name) [Berkshire]:Kathmandu
Locality Name (eg, city) [Newbury]:Kathmandu
Organization Name (eg, company) [My Company Ltd]:SANOG
Organizational Unit Name (eg, section) []:IP Services Workshop
Common Name (eg, your name or your server's hostname) []:localhost
Email Address []:root@localhost

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:                   [empty]
An optional company name []:               [empty]
[root@localhost tmp]#
```

A configuration session cont.

Fourth step:

```
[root@localhost tmp]# openssl x509 -req -days 60 -in server.csr -signkey server.key -out server.crt
Signature ok
subject=/C=NP/ST=Kathmandu/L=Kathmandu/O=SANOG/OU=IP Services Workshop/CN=IP
Services Workshop/emailAddress=root@localhost
Getting Private key
Enter pass phrase for server.key:          [password from before goes here]
[root@localhost tmp]#
```

Installing the certificate

Go to /etc/httpd/conf and do the following:

- `cd /etc/httpd/conf`
- `cp tmp/server.crt ssl.crt/.`
- `cp tmp/server.key ssl.key/.`
- `cp tmp/server.csr ssl.csr/.`
- `service httpd stop`
- `service httpd start`

Installing the certificate cont.

When the Apache server asks for a password give the password you used previously.

Now take a look at /var/log/messages. If you had a problem most likely there would be a message describing the problem in /var/log/message.

Now try to open the page:

https://localhost/

Note the “https”. What happens? Examine the certificate that is presented to you.

Remove the password

You probably noticed that Apache asked for a password in order to start. Unfortunately this is not likely to work in a server environment.

To remove the password use the file server.pem. This is the same as server.key, but it's not encoded. To make this change do the following in /etc/httpd/conf:

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

Remove the password cont.

And, now restart the Apache server:

```
service httpd restart
```

This time you should not have received any request for a password to start Apache.

Go to https://localhost/ again and examine the certficate. Repeat this process we just did if you find anything you need to change.

Go to the IP address of your neighbor's machine (https in a web browser) and see if you can see their start page and certificate.

Solving problems

If you cannot connect to the server check the following:

- Check if iptables is running and blocking access to port 443.
- If the certificate is properly created.
- If the configuration in /etc/httpd/conf.d/ssl.conf is correct.
- 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/httpd/ssl_error_log

And, as always, you can use:

http://www.google.com/

or

http://www.google.com/linux

More resources

- http://www.modssl.org/
- http://www.apache.org/
- http://www.openssl.org/
- http://www.ws.afnog.org/
- http://www.oreilly.com/ and check out the books that deal with SSL.

Conclusion

The installation of Apache with mod_ssl permits you to run a “secure” web server.

If you run webmail a secure server is *essential* for your security and your client's security.

Apache with mod_ssl=https. This is an extra load on your server. If you have many webmail clients you may need to plan accordingly.

These days you can get a signed certificate for not too much money. We'll take a look at some of the signing authorities in your web browser now.

Without a signed certificate there is a fundamental problem of trust when connecting to a server.