Apache and Virtual Sites and SSL

Dorcas Muthoni

Scope

- What is Apache
- What is Apache+mod_ssl+Vhosts
- Digital Signatures
- Installing Apache+mod_ssl
- Configuring Apache+Vhosts
- Your webserver
- Configuring Apache+mod_ssl

What is Apache

- HTTP Webserver: **accepts HTTP requests from clients (web browsers), and serves them HTTP responses** along with optional data contents
- By Apache Group and originally written for UNIX, but now runs under Linux, OS/2, Windows and other platforms.
- As of April 2008 Apache served **50.42% of all websites**.
- Developed and maintained by an open community of developers under the auspices of the **Apache Software Foundation.**

What is Apache+mod_ssl+Vhosts

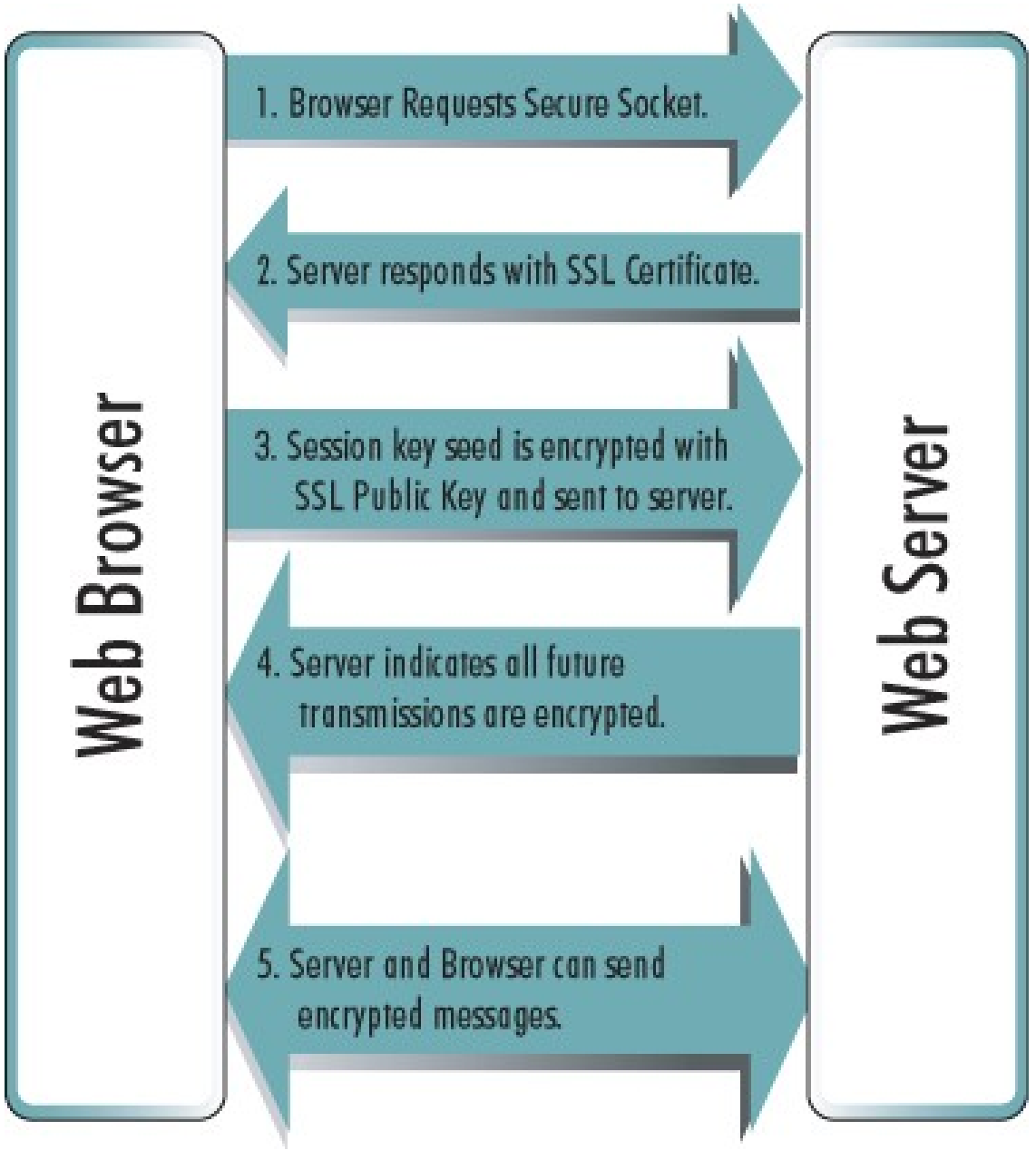
- **mod_ssl**
  - Apache HTTP Server module mod_ssl **provides an interface to the OpenSSL library**, which provides Strong Encryption using the Secure Sockets Layer and Transport Layer Security protocols.
  - SSL **provides for secure communication between client and server by allowing mutual authentication**, the use of digital signatures for integrity, and encryption for privacy

- **virtual hosts**
  - allows **one Apache installation to serve many different actual websites.**
  - For example, one machine, with one Apache installation could simultaneously serve *www.example.com* , *www.test.com*

Digital Signatures

- Transport Layer Security (**TLS**) and its predecessor, Secure Sockets Layer (**SSL**), are .**cryptographic protocols** that provide **secure communications on the Internet** for such things as **web browsing, e-mail, Internet faxing, instant messaging and other data transfers**
- **digital signature:** type of asymmetric cryptography used to simulate the security properties of a handwritten signature on paper.
  - one for signing which involves the user's secret or private key, and one for verifying signatures which involves the user's public key.

Secure Transaction



src:www.ingeo.com

Installing Apache+mod_ssl

- Lets install Apache with mod_ssl
  - mod_ssl: module provides strong cryptography for the Apache webserver via SSL and TLS protocols by the help of the Open Source SSL/TLS toolkit OpenSSL
- Installation
  - **# portinstall apache**
- Enable apache to start automatically on boot

  # vi /etc/rc.conf

  - Add

  apache22_enable="YES"

Configuring Apache+Vhosts

- **cd /usr/local/etc/apache22/**
- **vi httpd.conf**

- Review key the conf file
  - **ServerRoot "/usr/local"** : *top of the directory tree under which the server's configuration, error, and log files are kept*
  - **Listen 80** : *bind Apache to specific IP addresses and/or ports*
  - **ServerAdmin:** *Your address, where problems with the server should be e-mailed.*
  - **DocumentRoot***: The directory out of which you will serve your documents.*
  - **ErrorLog***: The location of the error log file*

Configuring Apache+Vhosts

- **Supplemental configuration**
  - The configuration files in the etc/apache22/extra/ directory can be included to add extra features or to modify the default configuration of the server
- **Virtual hosts**

  # Virtual hosts

  Include etc/apache22/extra/httpd-vhosts.conf
- **SSL/TLS**

  # Secure (SSL/TLS) connections

  # Include etc/apache22/extra/httpd-ssl.conf (to be enabled in a later session)

Your webserver

- Create the directory for your files in the Document Root

  # mkdir /usr/local/www/apache22/data

- Test apache:

  # telnet localhost 80

  not running

- Start apache

  # apachectl start

- Create a page for your home

  # ee /usr/local/www/apache22/data/index.html

- Visit your homepage, on your browser
  - http://localhost Or http://yourIPaddress

Configuring Virtual Hosts

- Supplemental configuration

  cd /usr/local/etc/apache

  ee extra/httpd-vhosts.conf

  (last directive for those who did not install apache22)

- If you want to **maintain multiple domains/hostnames on your machine** you can setup VirtualHost containers for them.
  - e.g. med.youruni.ac.ke, bs.youruni.ac.ke
- With **name-based virtual hosts** the server doesn't need to worry about IP addresses
- **Almost any Apache directive may go into a VirtualHost container**.

Configuring Virtual Hosts

```
<VirtualHost *:80>
    ServerAdmin webmaster@site1.example.com
    DocumentRoot /usr/local/www/data/site1
    ServerName site1.test.sae.ws.afnog.org
    ErrorLog "/var/log/site1-error_log"
#   CustomLog "/var/log/site1-access_log"
</VirtualHost>

<VirtualHost *:80>
    ServerAdmin webmaster@site2.example.com
    DocumentRoot /usr/local/www/data/site2
    ServerName site2.test.sae.ws.afnog.org
    ErrorLog "/var/log/site2-error_log"
#   CustomLog "/var/log/site2-access_log"
</VirtualHost>
```

Configuring Apache+mod_ssl

- Supplemental configuration
  - On the httpd.conf

    # Secure (SSL/TLS) connections

    Include etc/apache22/extra/httpd-ssl.conf (to be enabled in a later session)

    # ee extra/httpd-ssl.conf

Configuring Apache+mod_ssl

- Supplemental configuration
  - On the httpd.conf

    # Secure (SSL/TLS) connections

    Include etc/apache22/extra/httpd-ssl.conf (to be enabled in a later session)

    # ee extra/httpd-ssl.conf

Key Generation

- **Generate a Private Key**
- The openssl toolkit is used to generate an RSA Private Key and CSR (Certificate Signing Request). It can also be used to generate self-signed certificates which can be used for testing purposes or internal usage.
- The first step is to create your RSA Private Key. This key is a 1024 bit RSA key which is encrypted using Triple-DES and stored in a PEM format so that it is readable as ASCII text.

 # openssl genrsa -des3 -out server.key 1024

- Enter paraphrase

Simple paraphrase

Generate a CSR (Certificate Signing Request)

- Once the private key is generated a Certificate Signing Request can be generated. The CSR is then used in one of two ways. Ideally, the CSR will be sent to a Certificate Authority, such as Thawte or Verisign who will verify the identity of the requestor and issue a signed certificate.

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

Remove Passphrase from Key

- Apache will ask for the pass-phrase each time the web server is started

  # cp server.key server.key.org

  # openssl rsa -in server.key.org -out server.key

Generating a Self-Signed Certificate

- At this point you will need to generate a self-signed certificate because you either don't plan on having your certificate signed by a CA

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

Installing the Private Key and Certificate

- **Ensure**
  - server.crt
  - server.key
- Are in the Apache config directory

Configuring SSL Enabled Virtual Hosts

edit /extra/httpd-ssl.conf

```
SSLEngine on

SSLCertificateFile /usr/local/apache/conf/ssl.crt/server.crt

SSLCertificateKeyFile /usr/local/apache/conf/ssl.key/server.ke
    y

SetEnvIf User-Agent ".*MSIE.*" nokeepalive ssl-unclean-
    shutdown

CustomLog logs/ssl_request_log \

   "%t %h %{SSL_PROTOCOL}x %{SSL_CIPHER}x \"%r\" %b"
```

Restart Apache and Test

Restart Apache and Test