

SSL, Credit Card Transactions

CS174

Chris Pollett

Nov. 5, 2007.

Outline

- HTTPS and the Secure Socket Layer
- Credit Card Transactions

HTTPS and the Secure Socket Layer

- When we use HTTP to browse the web, data is typically sent over a TCP connection and is not encrypted.
- This is bad if we want to keep things like our credit card info secret.
- Shortly after the web became popular, Netscape proposed using HTTP over a Secure Socket Layer (SSL).
- When you see a page with the https: uri schema, SSL is being used to encrypt the data that is sent in the TCP connection.
- https uses port 443 rather than port 80.

HTTPS: How it works

- First, a socket connection is made to the server on port 443 using TCP.
- Then the browser attempts to establish an SSL connection with the server:
 1. Browser sends a cipher list, and a random string R_A (nonce)
 2. The server replies with a certificate signed by some certificate authority, a cipher its willing to use from clients list, and a nonce R_B .
 3. The client checks if the certificate has been signed by a certificate authority it knows by applying public keys of known authorities to the certificate, if it checks, a pre-session key is created and encrypted with server's public key, this is sent with encryptions of hashes of previous messages, making use of the nonces and a client literal string.
 4. The server replies with a hash of the previous messages, a server literal, and a key made from a hash of the pre-session key and the nonces.
 5. Secure communication is then done using the hash of the pre-session key and the two nonces, as the session key.
 6. This communication in HTTPS consists then of regular HTTP commands.

Configuring Apache for SSL

- To use SSL with Apache you need to load the SSL module. For example, you might have to uncomment a line like:

```
LoadModule ssl_module libexec/apache2/mod_ssl.so
```

- Usually most of the SSL directives are in a separate configuration file which is included into httpd.conf.
- Find this file and then look for a `<VirtualHost _default_:443>` directive. There should be a `DocumentRoot` directive within this that let's you set the root directory for https connections.
- You will also see somewhere in this configuration file the directives: `SSLCertificateKeyFile` and `SSLCertificateFile`. These should point at a reasonable `server.key` and `server.crt` file.

server.key and server.crt

- These are needed for Step 2 of our description a couple slides back for SSL.
- To get certificates which will work will work without complaints with most browsers you need to buy one from a company like Verisign or Thawte.
- Alternatively, for testing purposes you can create a self-signed certificate.

Creating a self-signed certificate

- First, you need to get a tool to do the necessary cryptography such as openssl (<http://www.openssl.org>)
- Then one first generates a private key:
`openssl genrsa -des3 -out server.key 1024`
- Next one generates a certificate signing request (CSR -- this is actually what you would give to Verisign or Thawte if you were buying a certificate).
`openssl req -new -key server.key -out server.csr`
- Generate a self-signed certificate:
`openssl x509 -req -days 365 -in server.csr -signkey server.key -out server.crt`
- Put `server.key` and `server.crt` in the correct places, restart server.
- Note some packages like XAMPP already come with self-signed certificates.

Credit Card Transactions

- One use for HTTPS is so that you can do credit card transactions on the web.
- One way to do this is to get a merchant account with a bank and use the bank's API to handle the online transaction. This involves sending data to the bank you are a merchant of (the acquiring bank) which in turn communicates with the card issuing bank.
- Another way is to use a third party merchant which then communicates with a bank.
- The latter companies usually have two different ways of allowing you to do the transaction:
 - (1) You can send the client on to the 3rd party merchant with the data needed for the transaction. The transaction gets handled on the 3rd party merchant's site.
 - (2) You can use SSL and communicate with the 3rd party merchant on the server-side. The client never leaves your site.

Authorize.Net

- We will consider the second of these two approaches in the context of Authorize.net, a 3rd party payment gateway.
- Info about their api and sample code can be found at: <http://developer.authorize.net/>
- We will now consider some of the sample code downloaded from their site.
- The basic key to how this works are the `curl_init`, `curl_exec`, and `curl_close` commands which are used to open a connection from your server to Authorize.net

