SSL Presentation

1. Practical SSL With Apache and PHP
2. Part I: The Players
3. Basics: Hashing
4. Basics: Symmetric Encryption
5. Basics: Asymmetric Encryption
6. Certificates
7. Certificate Authorities
8. Transport Layer Security
9. TLS: What It Does
10. TLS: The Handshake Protocol
11. TLS: Server Handshake Response
12. TLS: Server Certificate
13. TLS: Server Certificate Verification
14. TLS: Root Certificates
15. TLS: Failed Verification
16. TLS: Client Handshake Response
17. TLS: Handshake Wrapup
18. TLS: Record Protocol
40. Decrypting
41. Verifying
42. Checking The Key Passphrase
43. Parsing X.509 Data
44. Discussion
45. Thank You!
46. Resources
47. Download
Practical SSL With Apache and PHP

... and OpenSSL, mod_ssl, X.509 Certificates.

Chris Snyder and Mike Southwell
New York PHP
22 November 2005
Part I: The Players

Cryptography basics

Certificates

Transport Layer Security

What you need:

- mod_SSL
- OpenSSL toolkit
- PHP --with-openssl
Basics: Hashing

not reversible
has native PHP support

md5('hello') : 5d41402abc4b2a76b9719d911017c592
sha1('hello') : aaf4c61ddcc5e8a2dabede0f3b482cd9aea9434d
Basics: Symmetric Encryption

Symmetric: both sides use the same key

Reversible
Requires out-of-band key exchange.
PHP needs mcrypt extension

3DES
AES
Blowfish
RC4

We each have the same secret key.
You encrypt a value with this key, and send it to me.
The only way to decrypt it is with our shared secret key.
Basics: Asymmetric Encryption

Asymmetric: encryption and decryption use different keys

Reversible
Allows public knowledge of encryption key
PHP needs OpenSSL extension

RSA

I give you my public key.
You encrypt a value with my public key, and send it to me.
The only way to decrypt it is with my private key (which only I have).
Certificates

X509 Public Key Infrastructure

RFC3280 defines Certificates and Certificate Revocation Lists.

http://rfc.net/rfc3280.html
Certificate Authorities

Certificate Authorities provide secured identification of a server, and enable asymmetric encryption of messages between client and server.
Transport Layer Security

A 30-second history

Secure Sockets Layer was developed by Netscape in 1994 as a protocol which permitted persistent and secure transactions. In 1997 an Open Source version of Netscape’s patented version was created, which is now OpenSSL. In 1999 the existing protocol was extended by a version now known as Transport Layer Security (TLS). By convention, the term "SSL" is used even when technically the TLS protocol is being used.
TLS: What It Does

- TLS encrypts messages.
- TLS makes message alteration detectable.
- TLS authenticates message senders/receivers.
SSL encrypts every transaction between the client browser and the server, which makes it possible to send sensitive information back and forth without fear that it will be readable by anybody who might intercept it. It can do this because, before it begins transferring encrypted information, the server engages in an elaborate negotiation with the client, called the Handshake Protocol. This negotiation has the following parts:

- The client sends a request to the server which, because it uses the https (as opposed to http) schema, initiates the negotiation.
TLS: Server Handshake Response

The server responds with a plain-text message consisting of the following parts:

1. An exchange method to be used for passing back and forth the keys to be used for encrypting information. This is typically either RSA or Diffie-Hellman-Merkle. If it is RSA, the server must send along also a Certificate (discussed below).

2. The type of encryption to be used (RC4 or preferably 3DES).

3. The technique to be used for calculating the Message Authentication Code, a checksum appended to messages and used to verify that the message contents haven't been tampered with. Typically MD5 or SHA-1.
TLS: Server Certificate

RSA key exchange enhances security by requiring the server to send a Certificate to the client. This Certificate is a binary collection of the following information:

1. Its identity

2. Its own attestation that it really is who it has said

3. A Certificate Authority’s attestation that the server’s attestation is true

4. Its public key, which may be used to encrypt the message encryption key, randomly generated by the client
TLS: Server Certificate

[Diagram of TLS handshake]

Sequence number: 1 (relative sequence number)

Header Length: 20 bytes

Window size: 4,096

Acknowledgement number: 1,06 (relative ack number)

T: This is an ACK to the segment in frame 8

Secure socket layer

 funcion encoded packet: SSL

<table>
<thead>
<tr>
<th>SeqNo</th>
<th>Time</th>
<th>Flags</th>
<th>SeqNum</th>
<th>AckNum</th>
<th>DataLen</th>
<th>Checksum</th>
<th>Reserved</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
</tr>
<tr>
<td>0001</td>
<td>0001</td>
<td>0001</td>
<td>0001</td>
<td>0001</td>
<td>0001</td>
<td>0001</td>
<td>0001</td>
</tr>
<tr>
<td>0002</td>
<td>0002</td>
<td>0002</td>
<td>0002</td>
<td>0002</td>
<td>0002</td>
<td>0002</td>
<td>0002</td>
</tr>
<tr>
<td>0003</td>
<td>0003</td>
<td>0003</td>
<td>0003</td>
<td>0003</td>
<td>0003</td>
<td>0003</td>
<td>0003</td>
</tr>
<tr>
<td>0004</td>
<td>0004</td>
<td>0004</td>
<td>0004</td>
<td>0004</td>
<td>0004</td>
<td>0004</td>
<td>0004</td>
</tr>
<tr>
<td>0005</td>
<td>0005</td>
<td>0005</td>
<td>0005</td>
<td>0005</td>
<td>0005</td>
<td>0005</td>
<td>0005</td>
</tr>
<tr>
<td>0006</td>
<td>0006</td>
<td>0006</td>
<td>0006</td>
<td>0006</td>
<td>0006</td>
<td>0006</td>
<td>0006</td>
</tr>
<tr>
<td>0007</td>
<td>0007</td>
<td>0007</td>
<td>0007</td>
<td>0007</td>
<td>0007</td>
<td>0007</td>
<td>0007</td>
</tr>
</tbody>
</table>

secure socket layer (SSL)
TLS: Server Certificate Verification

- The client browser recognizes the Certificate Authority and thus verifies the authenticity of the connection.
TLS: Root Certificates

It can do this because it has pre-installed certificates from many Authorities.
TLS: Failed Verification

If there is a conflict between the name on the certificate and the name of the server, the browser pops up a Domain Name Mismatch notice, allowing the user to decide whether to continue.

Security Error: Domain Name Mismatch

You have attempted to establish a connection with 'mail.optonline.net'. However, the security certificate presented belongs to 'webmail.optonline.net'. It is possible, though unlikely, that someone may be trying to intercept your communication with this web site.

If you suspect the certificate shown does not belong to 'mail.optonline.net', please cancel the connection and notify the site administrator.

View Certificate

OK Cancel Help
TLS: Client Handshake Response

- Assuming the client does trust the server, it generates a random key to be used by the server for message encryption. It then encrypts that key with the server’s public key, and sends it back to the server.
TLS: Handshake Wrapup

At this point, the Handshake Protocol is finished. Both the server and the client know the following:

1. What key and encryption method to use for message encryption

2. What key and hashing method to use for message signing

There is, then, a great deal going on behind the scenes, which explains why SSL connections are sometimes slower than insecure ones.
Now the Record Protocol starts to manage all future communication. The server finally is ready to respond to the original request, as follows:

1. It encrypts its http response.
2. It calculates a hash of its response
3. It appends that hash as a signature.
TLS: Record Protocol (cont'd)

Upon receiving the server’s response, the client does the following:

1. It detaches the signature.
2. It calculates a new hash of the response, and compares that to the detached signature.
3. (Assuming they match,) it decrypts the response.
4. It displays the requested html.
Apache's mod_ssl

Installed by default in Apache 2, and available for Apache 1.3, mod_ssl provides cryptography so that Apache can interact with the OpenSSL toolkit.

http://www.modssl.org/
OpenSSL

The OpenSSL project provides a toolkit and cryptography library for implementing the TLS protocol.

http://www.openssl.org/
PHP's OpenSSL Support

PHP reports that it must be compiled --with-openssl in order to use its wrapper support for OpenSSL, although that appears not to be completely true.

http://www.php.net/openssl
The Big Picture

Green parts are created by local CA, pink parts by third-party CA.
Part II: SSL-enabled Webservers

HTTPS Requests and Responses are:

- Encrypted
- Tamper-proof
- Authenticated (Server, anyway)
HTTPS Requirements

- Apache with mod_ssl
- An IP Address
- Private server key (passphrase encrypted)
- Signed Server Certificate
- (optional) Certificate Authority Certificate(s)
- (optional) PassphraseDialog script
Using PHP to simplify PKI

phpca is a command line utility to create and manage a simple Certificate Authority for use with Apache's mod_ssl.

```shell
./phpca <command> [<server> | "ca"]
```
A Local Certificate Authority

./phpca cagen
A New Server Certificate

./phpca newserver ssl.example.org
Global SSL Directives

## Global SSL Directives

```bash
#LoadModule ssl_module modules/mod_ssl.so
Listen 443

# Random number generation
SSLRandomSeed startup file:/dev/urandom 512
SSLRandomSeed connect file:/dev/urandom 512

# Session Cache
SSLSessionCache none
SSLSessionCacheTimeout 600
SSLPassphraseDialog builtin
SSLMutex
file:/Applications/xampp/xamppfiles/logs/ssl_mutex

AddType application/x-x509-ca-cert .crt
AddType application/x-pkcs7-crl .crl
```
A Passphrase Dialog

Create /root/httpdkey:

```
#!/bin/sh
  echo 'my secret password'
```

Make sure only root has access:

```
chmod 700 /root/httpdkey
```

Use an external SSLPassphraseDialog:

```
SSLPassphraseDialog exec:/root/httpdkey
```
Apache Per-host SSL Directives

## SSL Virtual Host Context

```html
<VirtualHost _default_:443>
  # General setup for the virtual host
  DocumentRoot "~/Users/csnyder/Sites/https"
  ServerName localhost
  ServerAdmin csnyder@chxo.com
  ErrorLog logs/sslerror.log

  # SSL Configuration
  SSLEngine on
  SSLCipherSuite HIGH:MEDIUM
  SSLCertificateFile /etc/httpd/ssl/localhost.cert
  SSLCertificateKeyFile /etc/httpd/ssl/localhost.key
  SSLCertificateChainFile /etc/httpd/ssl/ca.cert
  SSLCARevocationFile /etc/httpd/ssl/ca.crl

  # Client Authentication (Type):
  #SSLVerifyClient require
  #SSLVerifyDepth 10

  # SSL Engine Options:
  <Files ~ "\.(cgi|shtml|phtml|php?)$">
    SSLOptions +StdEnvVars
  </Files>

  # SSL Protocol Adjustments:
  SetEnvIf User-Agent ".*MSIE.*" 
    nokeepalive ssl-unclean-shutdown 
    downgrade-1.0 force-response-1.0

  # Per-Server Logging:
  CustomLog logs/ssl_request_log "%t %h %{SSL_PROTOCOL}x %{SSL_CIPHER}x" %b
</VirtualHost>
```
</VirtualHost>
Cipher Suite

DIY suite-ness!

`openssl ciphers -v 'HIGH:MEDIUM'`

Manual
Cross Your Fingers
Revoking A Server Certificate

./phpca revoke ssl.example.org
Part III: Application Level SSL

A PHP application can use OpenSSL's RSA support to:

- Sign values to prevent tampering
- Encrypt values to keep them private
Example: Stored Secrets

On storage, application needs:

- Application Certificate (Public Key) to encrypt values

On retrieval, application needs:

- Application Private Key to decrypt values
- ...and Private Key passphrase
Class openSSL

```php
class openSSL
{
    public function __construct()
    {
        include_once( './openSSL.php' );
        $this = new openSSL();
        $this->privateKey( file_get_contents( $server_key ) );
        $this->certificate( file_get_contents( $server_cert ) );
    }
}
```

http://www.nyphp.org/content/presentations/SSL/init.html[9/12/2009 7:01:59 PM]
$value_signed = $ssl->sign( $value , $passphrase );

Looks like:

So long, and thanks for all the fish.
-----BEGIN openSSL.php SIGNATURE-----
e35fbZmTXYbVP33HY1CX31hxirAiENmEY4A0JHiGqPDFs87BmDQszGHF
CSvnMAyyqF7OuwRYnRS+VdiDNy2y3Uh2qqB7SNGXgWZnRNsMTeGX4OKX
NB5uwUQ+zNu7QjYfapQyliYoOWS5L+GcmFRHveTwAw=
-----END openSSL.php SIGNATURE-----
Encrypting

Short values only! (56 chars max)

```
$secret = $ssl->encrypt( $value, $client_certificate );
```

Looks like:

Nlmc97wDavLSF7R078NcluVX8SrYMXdDFu9otugP5LDIPTW++VUQ5DOsZ9qVS9+AL01gv1H2Ub8eNRcV67WgQw/5lMA92+T5KprSqV+CP/FXfcNCnd3pq/H6TkP0LT7zX9Q7GZg/4UuMHmc9EcGHhoOxBM=
Decrypting

$value = $ssl->decrypt( $secret, $passphrase );
Verifying

$verified = $ssl->verify( $signed_value, $client_certificate );
Checking The Key Passphrase

$success = $ssl->checkKey( $passphrase );
Parsing X.509 Data

```php
$ssl->certificate( $certificate_data );
$dn = $ssl->getDN();
print_r( $dn );
```

Looks like:

```plaintext
```
Discussion

How do you use SSL?
Thank You!

For many many more details, please consult "Pro PHP Security", published in September by Apress.
Resources

Some of the many resources used in the making of this presentation.

Cryptography

- Diffie-Hellman-Merkle Key Exchange
- PKCS #7: Cryptographic Message Syntax Standard
- PKCS #12: Personal Information Exchange Syntax Standard
- RSA Cryptosystem
- S/MIME Specification 3.1 (RFC 3851)

Legal

- CNET: Netscape Patents Crypto Protocol (Sept. 16, 1997)
- United States Patent: 4,405,829 (RSA)
- United States Patent: 5,657,390 (SSL)

Manuals

- mod_ssl Directives
- OpenSSL Manuals
- PHP OpenSSL Functions
Other How-Tos

- Certificate Management and Generation with OpenSSL
- Generating an SSL Certificate with Apache+mod_ssl

SSL/TLS Protocol

- The SSLv2 Protocol
- The SSLv3 Protocol
- The TLS Protocol v1.0
- X.509 Public Key Infrastructure Spec (RFC 3280)
Download

Presentation files as a [ZIP archive].

Contents:

- psslwap/FCNYLicense.txt
- psslwap/README
- psslwap/cnf_template.php
- psslwap/config-dist.php
- psslwap/configure
- psslwap/fcnyCLI.php
- psslwap/gpl.txt
- psslwap/openSSL.php
- psslwap/openSSLDemo.php
- psslwap/openssl.cnf
- psslwap/phpca
- psslwap.Players.png