How to get SSL into Apache?

Tutorial at Open Source Convent

99/08/22
Holger Reif
What is SSL?

- SSL = Secure Socket Layer
- Socket == concept of Unix network API
- Means for securing The Web?
- Means for securing _any_ connection oriented communication
- Most often used to secure HTTP (transport protocol for the web)
- Other Protocols use SSL as well (NNTP, POP3, IMAP, ...)
Properties

- authenticates server
- optionally authenticate client
- protect confidentiality (privacy)
- protect integrity (reliability)
- end-2-end security for communication
Typical Applications that use SSL

• e-commerce - ordering
  ⇒ protect form input sent to server
  ⇒ sensitive personal data?

• payment
  ⇒ protect credit card information
  ⇒ SET is different

• secure web-based intranet access
  ⇒ secure transmission of confidential content
What can SSL do for you?

- protect data transmitted to and from your server
- reduce some spoofing attacks
- provide reliable authentication
- show your security and privacy awareness
What can SSL *not* do for you?

- protect your server from being hacked
- protect data on your server from being stolen
- provide non-repudiation
- make you a security genius
- secure your server!!!
Competitor: Secure HTTP (S-HTTP)

- Attempt by Terisa Systems 1994-1996
  ⇒ http://www.terisa.com/shttp/intro.html
- Variety of features
  - Client/Server Authentication
  - Spontaneous Encryption
  - Request/Response Nonrepudiation
- better concept then HTTPS and fitted needs better for HTTP
- no browser support
- it’s dead!
## Comparison HTTP, HTTPS, S-HTTP

<table>
<thead>
<tr>
<th></th>
<th>unprotected content</th>
<th>protected content</th>
</tr>
</thead>
<tbody>
<tr>
<td>unprotected lines</td>
<td>HTTP</td>
<td>S-HTTP</td>
</tr>
<tr>
<td>protected lines</td>
<td>HTTPS</td>
<td></td>
</tr>
</tbody>
</table>
History of SSL

- developed and introduced by Netscape
- 1994 version 1 had serious security flaws and was never used
- 1994 version 2 in Navigator 1
- 1995 battle against PCT by MS (Private Communications Technology)
- 1996 version 3 in Navigator 3 - public review
- 1996-1999 TLS (transport layer security) a.k.a. SSL 3.1 by IETF
Design Goals of SSL

- Cryptographic secure
  - too much snake oil out there
- Interoperability
  - Can two person speaking same protocol communicate?
- Extensibility
  - What about new requirements?
- Relative efficiency
  - don’t require to much resources!
Ways to reach the goal

- Symmetric Encryption
- Asymmetric Authentication
  - digital signatures
- clear specification
- Parameter negotiation
  - Handshake
  - parameter reuse
Excursion: Crypto Primer

• Basic Algorithms
  – Symmetric (Secret Key) Cryptography
  – Asymmetric (Public Key) Cryptography
  – Hash Functions

• Their Use
  – Session keys / Enveloping
  – Digital signatures
  – Certificates, x.509, CAs, cert chains and CRLs

• Strength of ciphers
  – Key length
Symmetric (Secret Key) Cryptography

\[ \begin{align*}
\text{sender} & \quad \rightarrow \quad \text{encrypt} \quad \rightarrow \quad \text{recipient} \\
\text{same key!} & \\
\text{decrypt} & \\
\end{align*} \]
Symmetric (Secret Key) Cryptography (contd.)

- same key for both sender and recipient
- key needs to kept secret!!!
- Problem: key distribution doesn‘t scale
- work fast (Mbytes per second)
- mostly bit shuffling, depending on key
- examples: DES (56 bit), 3-DES (112 bit), RC4 (128 Bit)
Asymmetric (Public Key) Cryptography

public key

key pair (public + private)

encrypt

decrypt

sender

recipient
Asymmetric (Public Key) Cryptography (contd.)

• Recipient has key pair (public + private part)
• Sender needs public key
• Public key needs only to kept authentic (not secret!)
• underlaying math problem
  – e.g. factoring a product of two large primes (RSA)
• you can make public key operations fast
• slow private key operations (dozends a sec.)
• Examples: RSA, Diffie-Hellmann
Hash Functions

- no keys involved
- one way function
  - can’t regenerate original from the hash
- compresses size down to 128 or 160 Bit
- generates a fingerprint of the text
  - changing one bit in the text changes half the bits of the hash
- work fast (Mbytes per second)
- Examples: MD5, SHA-1
Use of cryptographic functions

- Session keys / Enveloping
- Digital signatures
- Certificates, x.509, CAs and cert chains
Data encryption:  
Session keys and Enveloping

• **Problem**
  – secret key crypto: key distribution
  – public: speed

• **Solution:** put your text in an envelope!

• securely agree on a (symmetric) session key with your Public Key

• fast bulk encryption with that symmetric key

• long term key exchange key (rarely used)

• compromise of short term Session key harmless
Digital signatures

• Idea: use your private key to show your will
• Problem: private key operations slow on huge documents
• Solution:
  – generate a hash
  – encrypt this hash with your private key
Digital signatures (contd.)

Signing

Data

MD5

MD5-Hash

private key

RSA

Signature

Verification

Data

MD5

MD5-Hash

public key

RSA

Signature

comparison

MD5-Hash?

Signatur correct?
Certificates

- Problem: how do I know it’s your pub key?
- Solution: use a passport!
- Certificate is a digital passport
  - contains identity of holder
  - public key of the holder
  - tied together with an issuers signature
  - opt.: name of the issuer
  - opt.: administrative information like date of issuance and validity
  - opt.: allowed usage (type of pass)
Certificates (contd.)

• How to prove to be the valid holder of the cert (no picture included)?
  – prove possession of private key
  – encrypt a challenge
  – verifier checks with pub key taken from cert

• “Your key is your identity”
X.509

- Important format for certificates
- broad usage
  - $\Rightarrow$ SSL/TLS
  - $\Rightarrow$ S/MIME
  - $\Rightarrow$ IPSec
- current version: V3
  - v1 starting
  - v2 provided for better identification of used keys
  - v3 provides for extensions
Example: X.509 certificate

Data:

  Version: 1 (0x0)
  Serial Number: 15179 (0x3b4b)
  Signature Algorithm: md5WithRSAEncryption
  Issuer: C=ZA, ST=Western Cape, L=Cape Town, O=Thawte Consulting
         cc, OU=Certification Services Division, CN=Thawte Server
         CA/Email=server-certs@thawte.com
  Validity
         Not Before: Jan 22 15:14:10 1999 GMT
         Not After: Feb 5 15:14:10 2000 GMT
  Subject: C=DE, ST=Bavaria, L=Dachau, O=The OpenSSL Project,
         OU=Security Services Division, CN=www.openssl.org
Example: X.509 certificate (contd.)

Subject Public Key Info:
    Public Key Algorithm: rsaEncryption
    RSA Public Key: (1024 bit)
        Modulus (1024 bit):
            00:cd:04:8f:03:df:f1:dc:0c:30:63:bf:14:e8:59:
            ........
        Exponent: 65537 (0x10001)
    Signature Algorithm: md5WithRSAEncryption
        ........
        c5:79
Certification Authorities

- Problem: how do I verify the validity of the cert?
- Solution: check the signature of the issuer
- Problem: How do I know issuer’s pub key?
- Doesn’t he has a certificate itself...
- Endless loop???
- You need to trust somebody
  - Trusted third parties identify the individual for you and issue them with a certificate
  - Certification Authorities
  - embedded into browsers or email clients
Certificate chains

- How to adopt the model of hierarchies?
  - Organizations
  - different roles and purposes
- Solution: hierarchical CAs
- Consequence: certificate chains

Root CA

```
  /   \
 /     \  
/       \
 intermediate CA
 /     \  
/       \
 end user  

/     \  
/       \
intermediate CA
 /     \  
/       \
 end user  

/     \  
/       \
end user  
```

end user
Certificate chains example

C=ZA
ST=Western Cape
L=Cape Town
O=Thawte Consulting
OU=Certification Services Division
CN=Thawte Personal Basic CA
Validity 1995.12.31 to 2020.12.31

C=ZA
ST=Western Cape
L=Durbanville
O=Thawte Consulting
OU=Thawte SB RSA IK 1998.9.16 17:55
CN=Thawte Server Basic RSA Issuer 1998.9.16
Validity 1998.09.16 to 2000.09.15

CN=Thawte Freemail Member
Email=Holger.Reif@SmartRing.de
Validity 1999.07.24 to 2000.07.23
Certificate Revocation Lists (CRL)

- Problem: What happens if a certified key is stolen and can be misused?
- Solution: issuer has a blacklist of invalid (revoked) certificates -> CRL
- Must be checked additionally
- CRL contains
  - Issuer’s name
  - date of issue
  - serial numbers of revoked certificates
  - signature of issuer
Strength of ciphers

- how to attack a cipher
- key length
- key length comparison
How to attack a cipher?

• search through all possible solutions (brute force)
  – go through key space
  – SSL challenge 8/1996
    ⇒ break 40-Bit RC4
  – DES challenge at RSA conference in January
    ⇒ break 56 bit DES

• find a principle problem in the algorithm
  – complete compromise
  – shortcut to be faster than brute force
Key length

- strength of modern crypto only lies in keys (otherwise: security by obscurity)
- key length $\Rightarrow$ strength
- strength $\Rightarrow$ effort to break a cryptographically secure message
- effort measured in time, space, computing power etc.
- relation length/strength different various algorithms
Key length comparison

• symmetric encryption: only brute force
  – 128 Bit == 2^128 possible keys

• asymmetric encryption: mathematical tricks
  (factoring, ...)
  – 1024 Bit == 200 digit number factoring
  – provided no better technique than factoring known!

• Digests: birthday attack
  – 160 Bit
Legal

• Patents, trade secrets and the like
• export control
• usage restrictions
Patents, trade secrets and the like

- RSADSI has a patent on RSA algorithm (expiring 2000/09/20) in USA
- IDEA is patented across Europe (Ascom)
- RC2, RC4 is considered a trade secret by RSADSI
- several other cryptographic algorithms are protected as well
- Ask your lawyer!

Export control

• (strong) crypto is treated as munitions
• munitions can’t be easily exported from USA
  – except for weak cryptography
  – except to Canada (general exception)
  – except for foreign subsidiaries of US companies (on request)
  – except for financial purposes (on request)
  – "export crippled"

• Wasenaar agreement
  – public domain software is not export controlled
  – countries may have stricter handling (cf. USA)

• Ask your lawyer!
Usage restrictions

- e.g. France had restrictions in use of crypto until currently (now only declaration is needed)
- e.g. Russia demands a license to develop or use crypto
- Ask your lawyer!
SSL Protocol Overview

- Protocol components
- Session concept
- Ciphersuite concept
- Handshake Protocol
Protocol components

<table>
<thead>
<tr>
<th>Application</th>
</tr>
</thead>
<tbody>
<tr>
<td>Handshake protocol</td>
</tr>
<tr>
<td>Alert protocol</td>
</tr>
<tr>
<td>Change cipher spec protocol</td>
</tr>
<tr>
<td>App data protocol</td>
</tr>
</tbody>
</table>

Record Layer

TCP Layer

IP Layer

SSL/TLS layer

Socket interface

network stack
Protocol components (contd.)

- **Record layer (Record protocol)**
  - requires *reliable* transport (no missing packets, correct order)
  - Blocking, compression, encryption, integrity

- **Handshake protocol**
  - (Re-)Negotiate parameters

- **Alert protocol**
  - Notify about possible problems

- **Change cipher spec protocol**
  - short cut
Session concept

- secret values
  - premaster / master secret
- Ciphersuite
  - see next slide
- cryptographic parameters
  - encryption keys
  - integrity preserving keys
  - initialization vectors
    - Session Keys / Enveloping!!!
Ciphersuite concept

- compression algorithm
- key exchange algorithm
- authentication algorithm
- encryption algorithm
- MAC algorithm
- examples
  - EXP-RC4-MD5
    ⇒ Kx=RSA(512), Au=RSA, Enc=RC4(40), Mac=MD5, exp
  - DES-CBC3-SHA
    ⇒ Kx=RSA, Au=RSA, Enc=3DES(168), Mac=SHA1
Handshake Protocol

- hello messages
- Certificate messages
- Key Exchange Messages
- Finished messages
Handshake Protocol (graphic)

ClientHello

(HelloRequest)

ServerHello

Certificate

CertificateRequest

ServerKeyExchange

ServerHelloDone

Certificate

CertificateVerify

CertificateRequest

ChangeCipherSpec

Finished

ClientKeyExchange

Application Data

bright messages for session continuing not needed

bold messages only for server authentication

italic messages only for client authentication

ChangeCipherSpec

Finished

Application Data

Application Data
Handshake Protocol (contd.)

• hello messages
  – start with negotiation
  – exchange random values
  – agree on algorithms
  – check for session resumption.
Handshake Protocol (contd.)

- **Certificate messages. Key Exchange Messages**
  - Exchange certificates and cryptographic information to
    ⇒ allow the client and server to authenticate themselves.
  - Exchange the necessary cryptographic parameters to allow
    ⇒ the client and server to agree on shared key
Handshake Protocol (contd.)

• **Finished messages**
  - Allow the client and server to verify that their peer has
    ⇒ calculated the same security parameters and that the handshake
    ⇒ occurred without tampering by an attacker.
SSL processing

Application Data

Fragment/Combine

Record Protocol Units

Compress

Compressed Unit

Encrypt

MAC

Encrypted

Encrypt

Transmit

TCP Packet

picture taken from mod_ssl manual
SSL and Browsers

- The obvious picture
- How to get a better browser
- Proxy solutions
The obvious picture

- most popular browsers (Netscape Navigator, MS Internet Explorer) are from USA
  ⇒ export crippled
- Opera from Norway
  ⇒ only for Win platform (yet)
- Lynx+SSL (patches are from US....)
  ⇒ Lynx enhanced with OpenSSL
  ⇒ Does anybody really uses this?
- curl
- Mosaic 2.6 was patched with SSLeay 0.4.1 ...
How to get a better browser?

- **Fortify**
  - Patch your Netscape
  - www.fortify.net
- **Have a full strength Crypto Service provider**
  - Win + MSIE
- **Get US-Versions of the browsers**
  - ftp.hacktic.nl
  - ftp.replay.com
Proxy solutions

- **SafePassage**
  - www.c2.net

- **stunnel**
SSL and Apache

- History
- Architecture
History

• Past development
• comparison of solutions
Open Source Software Convention Tutorial
How to get SSL into Apache?

Development in past

Apache-SSL

Sioux

Apache-SSL U.S.

1.0

1.1

(free)

Stronghold

mod_ssl

1.0

RH SWS

mod_ssl 2.0

Raven SSL

RH SWS

1.1

Hockey

Raven SSL


Holger Reif <holger@reif.net>
## Comparison

<table>
<thead>
<tr>
<th>Product</th>
<th>Apache-SSL</th>
<th>mod_ssl</th>
<th>RH SS</th>
<th>Raven SSL</th>
<th>Stronghold</th>
<th>Hockey</th>
</tr>
</thead>
<tbody>
<tr>
<td>Author</td>
<td>B. Laurie</td>
<td>R. Engeschall</td>
<td>RedHat</td>
<td>Covalent</td>
<td>C2 Net</td>
<td>M. Steiger</td>
</tr>
<tr>
<td>Location</td>
<td>UK</td>
<td>DE</td>
<td>US</td>
<td>US</td>
<td>US</td>
<td>US</td>
</tr>
<tr>
<td>License</td>
<td>open-source</td>
<td>open-source</td>
<td>commercial</td>
<td>commercial</td>
<td>commercial</td>
<td>commercial</td>
</tr>
<tr>
<td>Price</td>
<td>$0</td>
<td>$0</td>
<td>$249</td>
<td>$357</td>
<td>$995</td>
<td>$149</td>
</tr>
<tr>
<td>Availability</td>
<td>world wide</td>
<td>world wide</td>
<td>US only</td>
<td>US only</td>
<td>world wide</td>
<td>US only</td>
</tr>
<tr>
<td>US Usage</td>
<td>restricted</td>
<td>unlimited</td>
<td>conceding, 90 d. free</td>
<td>conceding, 90 d. free</td>
<td>unlimited</td>
<td>Unlimited</td>
</tr>
<tr>
<td>Support</td>
<td>voluntary, always free</td>
<td>voluntary, always free</td>
<td>conceding, 90 d. free</td>
<td>conceding, 90 d. free</td>
<td>90 d. free</td>
<td>90 d. free</td>
</tr>
<tr>
<td>SSL Engine</td>
<td>OpenSSL (+ RSAref)</td>
<td>OpenSSL (+ RSAref)</td>
<td>OpenSSL + BSafe</td>
<td>OpenSSL + BSafe</td>
<td>SSLeay</td>
<td>OpenSSL + BSafe</td>
</tr>
<tr>
<td>Version</td>
<td>1.38</td>
<td>2.3.10</td>
<td>2.0</td>
<td>2.4.2</td>
<td>2.2.8</td>
<td></td>
</tr>
</tbody>
</table>
General Architecture

• 3 parts: Apache, SSL provider, glue

• why not just one distribution?
  – remember: munitions is export controlled
  – “Hooks” are disallowed as well
mod_ssl specific architecture

1. OpenSSL
2. Apache Core

Open Source Software Convention Tutorial
How to get SSL into Apache?
Holger Reif <holger@reif.net>
OpenSSL

• How to build
• History
• Components
• recent developments
How to build (in real time)

- ./Configure
- make
- make test (optional)
- make install (optional)
History of OpenSSL / SSLeay

- **SSLeay**
  - done by Eric Young and (partly) Tim Hudson
  - First public release 0.4 at 1995/05/31
  - major releases each year
  - only free tool for implementing SSL
  - development partly founded by C2Net
  - Development stopped mid 1999

- **OpenSSL**
  - started end of 1998
  - active development committed to Open Source
Components

- General purpose Big Number library
- General Purpose Basic Crypto library (ciphers)
- ASN.1 functions
- General Purpose High Level Crypto library (PKCS)
- X.509 library
- General purpose utilities (openssl command)
- SSL/TLS library
Recent development

- better support for various standards
  - PKCS#7 (S/MIME)
  - PKCS#12 (certificate import/export)
- renegotiations handling
- certificate extension handling
- new algorithms
  - OAEP
- cleanups (compiler warnings, configuration handling)
mod_ssl

- recent development
- how to build
- Basic Configuration
- Where do I get my certificate from?
- Advanced Topics
- Future development
Recent mod_ssl development

- shared memory session cache
- DH/DSA support
- CRL support
- session renegotiation
- SSL related status available
- more SSL env vars for CGI programs
How to build (in real-time)

- configure
- cd ../apache
- make
How to configure

• it’s easy, really!
• examining adopted httpd.conf
How to configure - Basic Options

- SSLEngine on
- SSLMutex
  file:/app/sources/apache_1.3.6/logs/ssl_mutex
How to configure - See what happens

- SSLLog
  /app/sources/apache_1.3.6/logs/ssl_engine_log
- SSLLogLevel info
How to configure - Set Key and certificate

- `SSLCertificateFile`  
  /app/sources/apache_1.3.6/conf/ssl.crt/server.crt

- `SSLCertificateKeyFile`  
  /app/sources/apache_1.3.6/conf/ssl.key/server.key

- `SSLPassPhraseDialog` built-in
How to configure - Decrease load

- SSLSessionCache
dbm:/app/sources/apache_1.3.6/logs/ssl_scache
- SSLSessionCacheTimeout 300
How to configure - Improve Security

- SSLRandomSeed startup builtin
- SSLRandomSeed connect builtin
How to configure - The certificate

- Make cert (real time demonstration)
- Request a cert from Thawte
The certificate - rolling your own

- make certificate
The certificate - request one at Thawte

- You should have
- Generate a key pair
  - $ openssl genrsa -des3 -out server.key 1024
- See what’s the content
  - $ openssl rsa -noout -text -in server.key
- Remove the protection from the key
  - $ openssl rsa -in server.key -out server.key.unsecure
- Generate the request from existing key
  - $ openssl req -new -key server.key -out server.csr
- Examine the content
  - $ openssl req -noout -text -in server.csr
- Go to a CA of your choice
The certificate - request at Thawte (contd.)

- Proceed through different steps (see Annex)
  - paste in cert request
  - select contact persons
  - payment and address information
  - finished and summary
  - status page
  - load cert
  - install cert
What to do next

• Some more advanced examples and their effect
  – Client auth
    ⇒ Check the certs of your clients
  – PRNG seeding
    ⇒ provide source for real randomness

• Advanced features
  – GID
    ⇒ get strong encryption from newer export browsers
  – DSO
    ⇒ compile mod_ssl as dynamic module
Future developments

- Improved per directory renegotiations
  - less cryptographic operations
- Full HTTPS support for mod_proxy
  - gather data from a SSL hosts
- SSLListen Directive
  - add the SSL with just one directive
- LDAP support
- Improved stability
- See README.Wishes
Closure

• Any questions?
  - holger@reif.net
  - www.modssl.org (+ mailing list)
  - www.apache-ssl.org (+ mailing list)
  - www.openssl.org (+ mailing list)
  - comp.www.servers.unix