Securing Communications with your Apache HTTP Server

Lars Eilebrecht
`Lars@apache.org`

About Me

- Lars Eilebrecht
- Independent IT Consultant – based in London, UK
- Contributor to the Apache HTTP Server project since 1996
- Co-founder and member of The Apache Software Foundation
- Member of the ASF Security Team
- `www.eilebrecht.net`

Agenda

- Overview
- X.509, Keys and Certificates
- SSL/TLS protocol
- Apache HTTP Server configuration
  - Basic configuration details
  - Virtual Hosting and ACME Protocol Module
  - Cipher and Protocol configuration
  - Session Caching and TLS Session Tickets
  - Advanced Features

Why HTTPS and TLS?

- **Confidentiality and Data Privacy**
  - protects data from eavesdropping
  - only the intended recipient can read the data

- **Authentication**
  - allows for identification of server and optionally, the client

- **Data Integrity**
  - ensures that nobody can tamper with the data that is being transmitted

Keys and Certificates

- X.509: ITU-T standard (1988) for PKIs
- PKI: Public-Key Infrastructure
- CA: Certification Authority
- CSR: Certificate Signing Request
- CRL: Certificate Revocation List

Common X.509 File Types and Extensions

- **PEM**: base64-encoded DER certificate(s) or private key(s)
- **DER**: binary format based on Distinguished Encoding Rules (encoded ASN.1 values)
- **p12**: PKCS#12 format, certificate(s) and/or private key(s)
- **key**: commonly used for a PEM-encoded private key
- **crt/cer**: commonly used for a PEM-encoded certificate
- **csr**: commonly used for a PEM-encoded certificate signing request

PEM-encoded Certificate Example

```
-----BEGIN CERTIFICATE-----
MIIC2zCCAkSgAwIBAgIJANWZuQf40KViMA0GCSqGSIb3DQEBBQUAMFMxCzAJBgNV
BAYTAlhYMQwwCgYDVQQIEwNYWFgxDDAKBgNVBAcTA1hYWDEMMAoGA1UEChMDWFhY
MQwwCgYDVQQLEwM2NjYxDDAKBgNVBAMTAzY2NjAeFw0wODEwMDEyMzU1MDlaFw0w
[...]
BgNVHRMEBTADAQH/MA0GCSqGSIb3DQEBBQUAA4GBAFlaHQEXQdMVfvTay5x6fECa
QiefllN/69931EFmNX0mlpV8pFZ448PtoGlXiNd+rnfe2ttjPfmh4CXDN9q7NPUO
qntygrcWsGJxmVlu5s2q6KumrysEdqr+Da70zyed3Tfj/QYJfG1HAzfLCVZRKFQE
EuxxMbZd6XBXcXenuZzn
-----END CERTIFICATE-----
```

Certificate Structure

- Certificate
  - Version
  - Serial Number
  - Signature Algorithm
  - Issuer
  - Validity Period
  - Subject
  - Subject Public Key Info
  - Issuer Unique Identifier (*optional*)
  - Subject Unique Identifier (*optional*)
  - Extensions (*optional*)
- Certificate Signature Algorithm
- Certificate Signature

Certificate Subject DN

- **DN**: Distinguished Name
  - a sequence of identifiers in X.500 notation
- Common DN Keys:
  - **CN**: Common Name (e.g., first/last name or hostname)
  - **C**: Country (2-letter code)
  - **S**: State or province
  - **L**: Locality (e.g, City)
  - **O**: Organization
  - **OU**: Organizational Unit
- Example DN:

```
C=DE, L=Berlin, O=Example Inc.,
CN=www.example.com
```

Common Name for Server Certificates

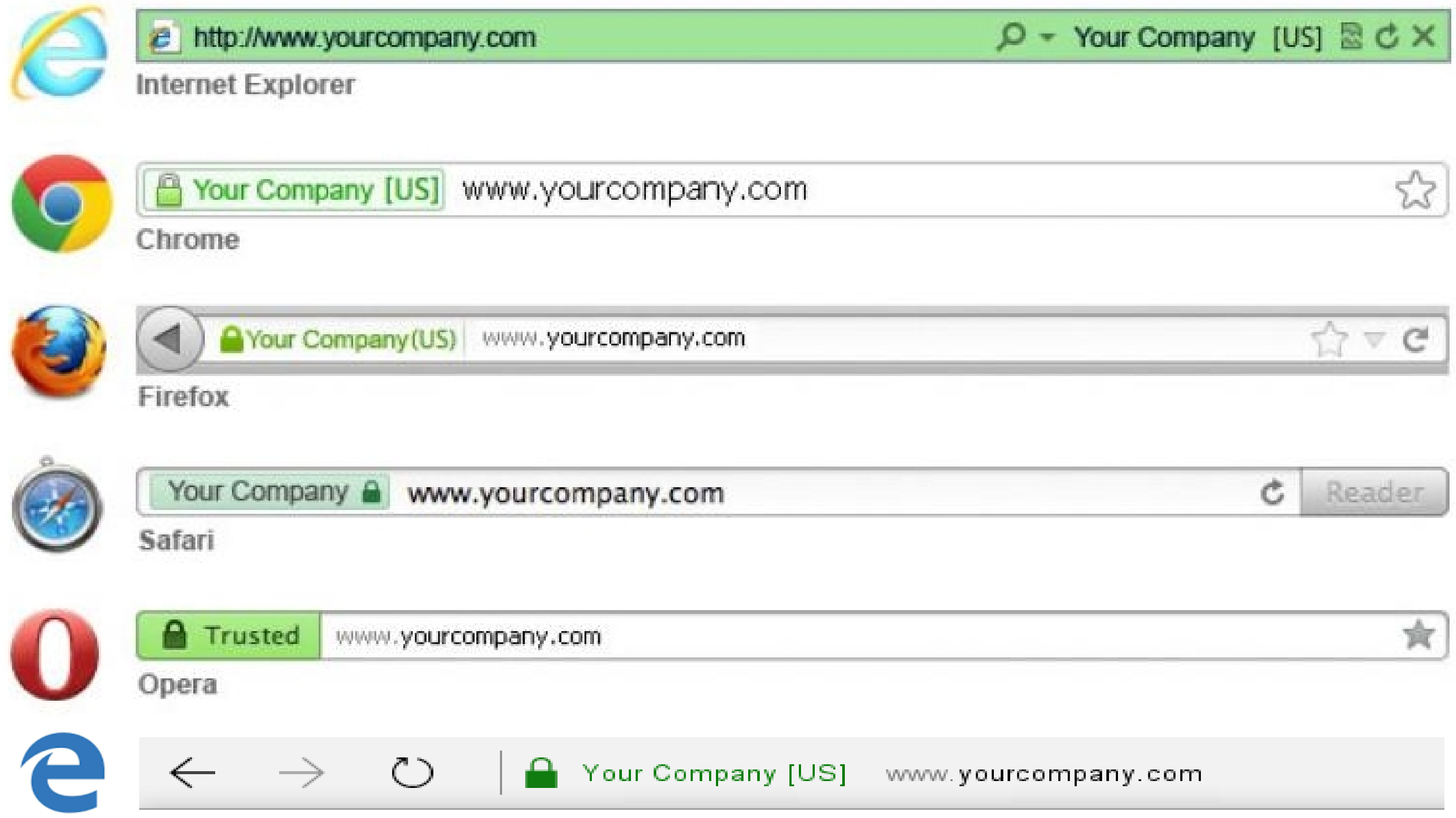
- Fully-qualified domain name (FQDN)
  - e.g., `www.example.com`
  - does not match `example.com`

- Wildcard domain
  - e.g., `*.example.com`
  - matches `example.com` and hosts such as `foo.example.com`
  - does not match `www.foo.example.com` or `example.com.foo`

Certificate Types

- Single-domain certificates
- Wildcard certificates
- Multi-domain (SAN) certificates
  - uses *SubjectAlternativeName* X.509 extension
- Extended validation (EV) certificates
  - available since 2007 and supported by Firefox 3+, IE 7+, Edge 12+, Opera 9.5+, Safari 3.2+ and Chrome 1+

Extended Validation Certificates



Obtaining a Certificate

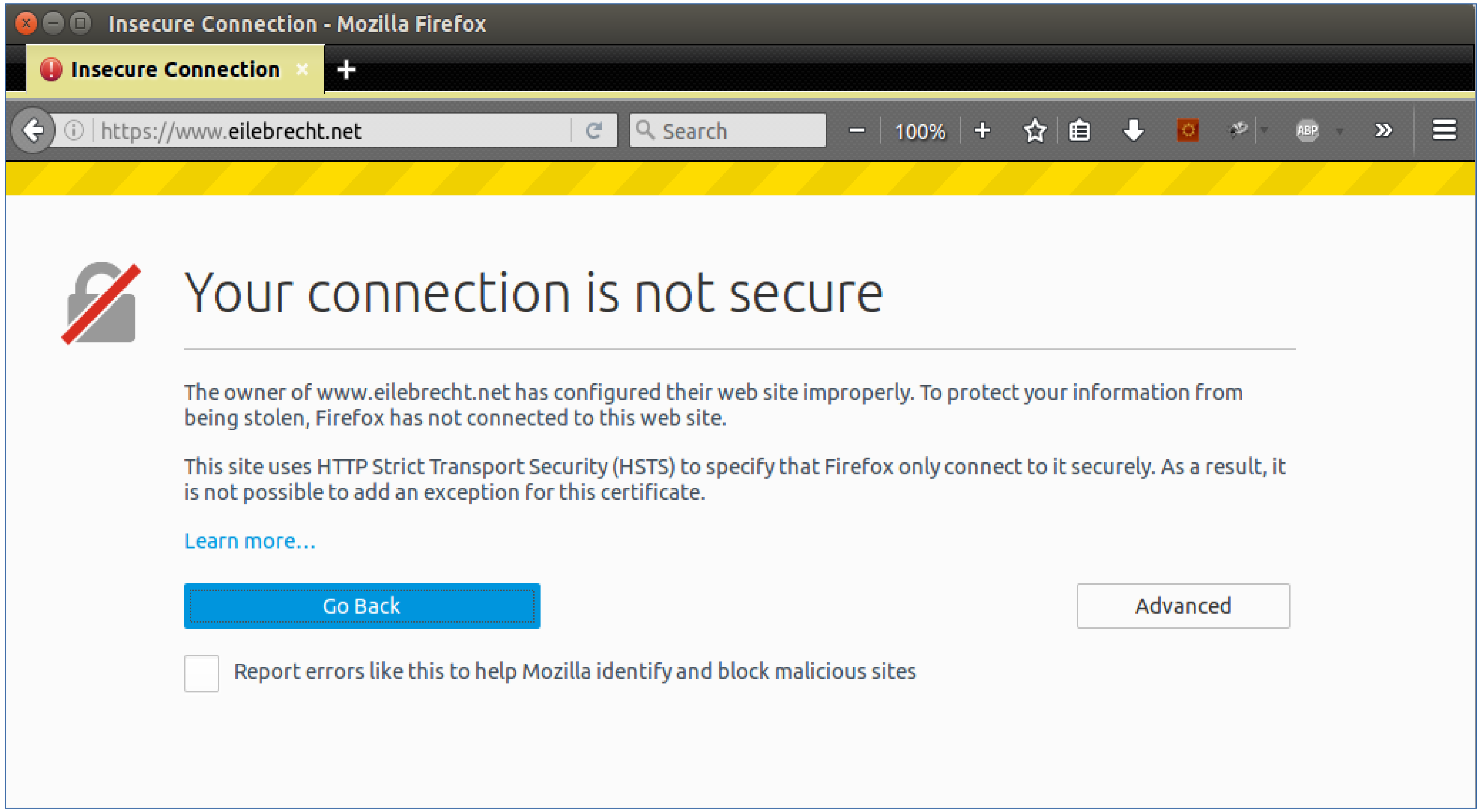
- create your own
  - self-signed certificate
  - signed by your own CA
- get a free certificate
  - free certificates from “Let's Encrypt” CA
  - trial or free certificates from commercial CAs
- buy a certificate from a CA
  - domain-only, organization or extended validation (6€ up to 1000€ per year)

Let's Encrypt CA

- `https://letsencrypt.org`
- Certificates are free of charge
- Fully automated validation
- Standard domain-validation certificates
- Multi-domain/SAN certificates
- Certificates are valid for 90 days
- Not valid as client certificate
- Supported by all modern Web clients
- Service provided by Internet Security Research Group (ISRG) since April 2016 (non-profit organisation)

Browser SSL Warnings

If the browser doesn't know the issuing CA or if the server hostname does not match the certificate it displays a warning to the user.


Insecure Connection - Mozilla Firefox
Insecure Connection
https://www.eilebrecht.net
Search
100%

Your connection is not secure

The owner of www.eilebrecht.net has configured their web site improperly. To protect your information from being stolen, Firefox has not connected to this web site.

This site uses HTTP Strict Transport Security (HSTS) to specify that Firefox only connect to it securely. As a result, it is not possible to add an exception for this certificate.

Learn more…

Report errors like this to help Mozilla identify and block malicious sites

Certificate Chain

- ➔ Root Certificate
    - ➔ Intermediate Certificate 1
        - ➔ Intermediate Certificate *n*
            - ➔ End-Entity (Leaf) Certificate
              (Server/Client Certificate)

SSL vs. TLS

- SSL: Secure Sockets Layer
  - originally developed by Netscape (1994)
  - SSL 2.0 and 3.0 deprecated and insecure
- TLS: Transport Layer Security
  - IETF standard (1999)
  - TLS 1.0, 1.1, 1.2, and 1.3
  - TLS 1.0, 1.1 should no longer be used
- When people talk about SSL these days they actually mean TLS.
- An “SSL certificate” is an X.509 certificate for use with TLS.

Apache SSL/TLS Module - mod_ssl

- Included as default module since Apache HTTP Server version 2.0
- Uses OpenSSL library
- Supports TLS 1.0, 1.1, 1.2 protocols
  - TLS 1.3 supported in Apache 2.5-dev (with OpenSSL 1.1+)
- SSL 3.0 is still supported, but SSL 2.0 support was removed in Apache HTTP Server version 2.4
- (Apache HTTP Server 2.0 and 2.2 are end of life!)

Module Configuration

- Required modules:
  - `LoadModule ssl_module modules/mod_ssl.so`

  -
    ```
    LoadModule socache_shmcb_module \
               modules/mod_socache_shmcb.so
    ```
- SSL configuration file:
  - `Include conf/extra/httpd-ssl.conf`

Basic Configuration

- Certificate and private key (PEM format):
  - `SSLCertificateFile \`
    `/usr/local/apache2/conf/ssl/server.crt`
  - `SSLCertificateKeyFile \`
    `/usr/local/apache2/conf/ssl/server.key`
    - Ensure the key file is only readable by root
- Enable SSL (per virtual host):
  - `SSLEngine On`
  - `Listen 443`

Intermediate CA Certificates

- Add server and all intermediate certificates to a single file and use `SSLCertificateFile`
  - Sort multiple certificates from leaf to root certificate!
- Multiple server certificates can be added to support (different authentication algorithms (ECC, RSA, DSA, etc.)
- `SSLCertificateChainFile` became obsolete with version 2.4.8

TLS Virtual Hosting

- TLS can be enabled for any virtual host
- Name-based virtual hosts with SSL/TLS only possible with SNI support available in Apache 2.4
- SNI: TLS Server Name Indication
- Clients must support SNI as well
- Clients without SNI support get either the first virtual host or a “403 Forbidden” response if `SSLStrictSNIVHostCheck` is enabled

ACME Protocol (Let's Encrypt) Module

- mod_md (Managing Domains)
- Available since 2.4.30, but still experimental!
- Enable certificate management for a virtual host:
  - `MDomain example.com www.example.com`
  - `MDCertificateAgreement`

    `https://letsencrypt.org/documents/LE-SA-v1.2-November-15-2017.pdf`
  - `ServerAdmin webmaster@example.com`

Ciphers and Protocols (default)

- Define ciphers and protocol:
  - `SSLCipherSuite HIGH:MEDIUM:!MD5:!RC4:!3DES`
  - `SSLHonorCipherOrder On`
  - `SSLProtocol All -SSLv3`

- Cipher string format (`SSLCipherSuite`):
  - prefix with “`!`” to permanently remove ciphers
  - prefix with “`-`” to remove ciphers
  - prefix with “`+`” to add ciphers (unless they have been removed with “`!`”)

Ciphers and Protocols (recommendation)

- Only use TLS 1.2 (or higher) with strong ciphers supporting forward secrecy:
  - `SSLCipherSuite HIGH:!MD5:!RC4:!3DES:!CAMELLIA:!kRSA`
  - `SSLProtocol All -SSLv3 -TLSv1 -TLSv1.1`
- Check which ciphers are enabled:
  - `openssl ciphers -v 'HIGH:MEDIUM:!MD5:!RC4:!3DES'`
  - Apache and OpenSSL force-disable certain ciphers
- Check “ciphers” man page for meanings of the various cipher strings such as “`HIGH`”, “`MEDIUM`”, “`ECDH`”, etc.

Random Seeds

- Define random seeds:
  - `SSLRandomSeed startup file:/dev/urandom 2048`
  - `SSLRandomSeed connect file:/dev/urandom 2048`

- multiple sources can be defined
- Apache's built-in default is not very secure (provides very little entropy)

TLS Session Cache

- Using SHM session cache is recommended
  - `SSLSessionCache shmcb:/var/run/ssl_cache(1024000)`
  - `SSLSessionCacheTimeout 600`

- avoid DBM session cache, it's slow and unstable under load
- each TLS session is about 150 bytes
- Using a very large session cache and/or long timeout compromises forward secrecy!

TLS Session Tickets

- Session tickets are enabled by default:
  - `SSLSessionTickets On`
- Disabling session tickets decreases performance!
- Recommendation when using TLS 1.2:
  - Disable session tickets if forward secrecy is a required.
  - If enabled, restart Apache at least once a day to reduce the impact on forward secrecy (this rotates the encryption key).
- Recommendation when using TLS 1.3:
  - Enable session tickets

OCSP Stapling

- OCSP: Online Certificate Status Protocol
- OCSP Stapling is known as the
  “TLS Certificate Status Request Extension”

- `SSLUseStapling on`
- `SSLStaplingReturnResponderErrors off`
- `SSLStaplingCache shmcb:/var/run/ocsp(128000)`

Client Certificate Authentication

- `SSLVerifyClient require`

- Using `SSLVerifyClient` in a per-directory context triggers renegotiation and should be avoided if possible.

Defining allowed Client Certificates

- Path to “bundle” file with one or more PEM-encoded CA certificates:
  - `SSLCACertificateFile`
- Path to CRL file:
  - `SSLCARevocationFile`
- Use CRL if possible, but OCSP can be used as an alternative:
  - `SSLOCSPEnable On`

Apache as an TLS Reverse Proxy

- `SSLProxyEngine`
- `SSLProxyCipherSuite`
- `SSLProxyProtocol`
- `SSLProxyCACertificateFile`
- `SSLProxyCACertificatePath`
- `SSLProxyCARevocationFile`
- `SSLProxyCARevocationPath`
- `SSLProxyCheckPeerCN`
- `SSLProxyCheckPeerExpire`
- `SSLProxyCheckPeerName`
- `SSLProxyMachineCertificateFile`
- `SSLProxyMachineCertificatePath`

HTTP Strict Transport Security

- Web security policy mechanism to protect against protocol downgrade. Enforce use of HTTPS.
- Example header:
  - `Strict-Transport-Security: max-age=31536000`
- Once the browser has cached the header, using plain HTTP or untrusted certificates is no longer possible.
- Can be configured with mod_md (incl. redirect to HTTPS):
  `MDRequireHttps permanent`

Securing Communications with your Apache HTTP Server

Any Questions?

Securing Communications with your Apache HTTP Server

Lars Eilebrecht
Lars@apache.org

Useful OpenSSL Commands

- Create self-signed certificate

  -
    ```
    openssl req -x509 -nodes -days 3650 -newkey rsa:2048 \
    -subj '/C=XX/L=Foo/CN=www.example.com' \
    -keyout server.key -out server.crt
    ```
- Remove passphrase from private key:

  -
    ```
    openssl rsa -in server.key -out server-nopass.key
    ```
- List available ciphers

  -
    ```
    openssl ciphers -v
    openssl ciphers -v 'HIGH:MEDIUM:!MD5:!RC4'
    ```

Useful OpenSSL Commands

- Display certificate contents
- `openssl x509 -text -in server.crt`
- Verify if a private key matches a certificate
- `openssl x509 -noout -modulus -in server.crt | md5sum`
- `openssl rsa -noout -modulus -in server.key | md5sum`
- Connect to a Web server using HTTPS
- `openssl s_client -connect www.example.com:443`

Useful OpenSSL Commands

- Check if OCSP response or client certificate authentication request is sent by server:

  -
    ```
    openssl s_client -connect www.example.com:443 -status
    ```
- Connect and define SNI server name:

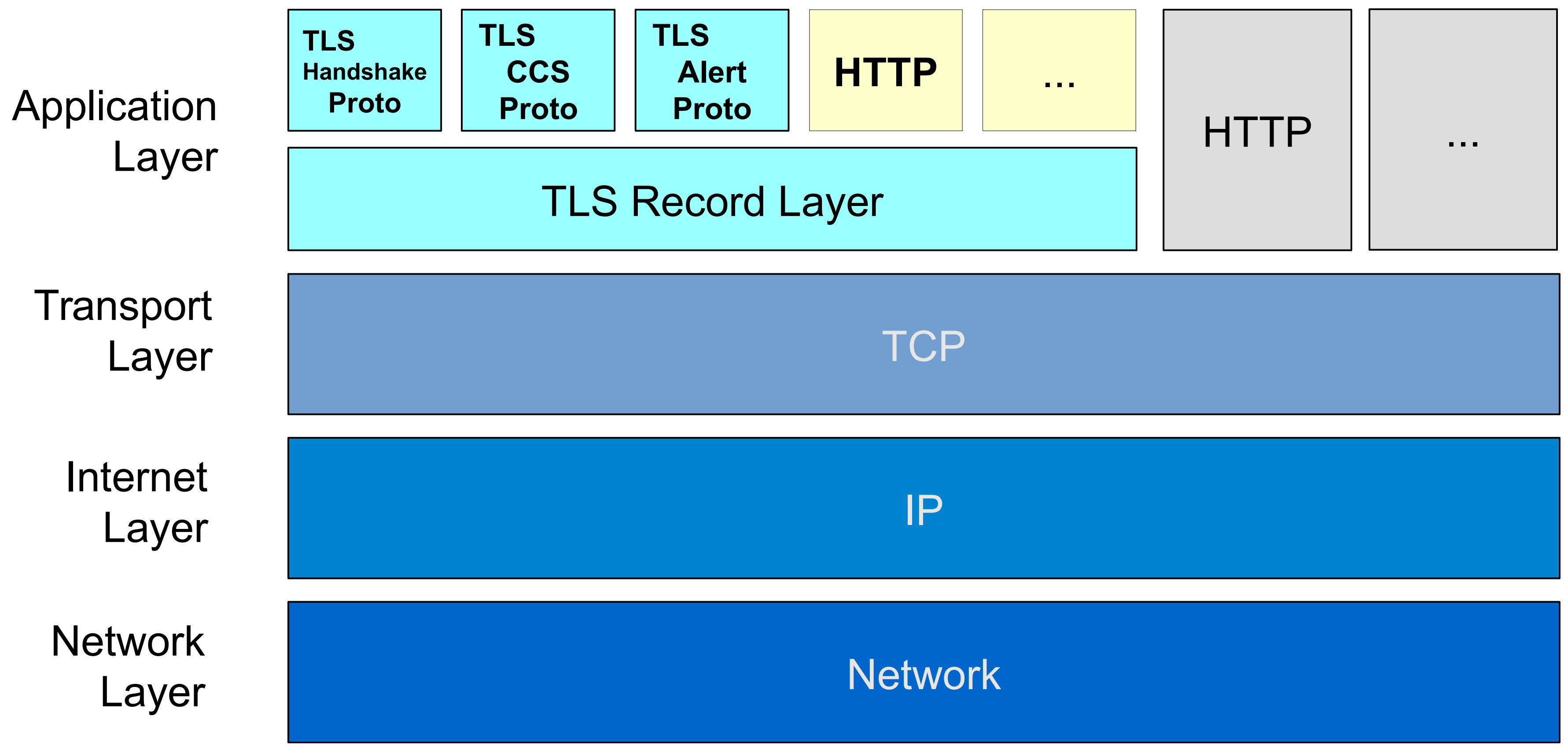
  -
    ```
    openssl s_client -connect www.example.com:443 \
                   -servername www.example.com
    ```
- Show description of error code:

  -
    ```
    openssl errstr <ERROR-NUMBER>
    ```

Cryptography Essentials

- Public-Key (asymmetric) Cryptography (e.g., RSA, DSA, ECC)
  - Data encrypted with the public key can only be decrypted with the corresponding private key
  - Data signed with the private key can be verified by anyone using the public key
- Symmetric-Key Cryptography (e.g., AES, Twofish)
- Hash Function (e.g., SHA-2, SHA-3)
- Message Authentication Code (e.g., HMAC)

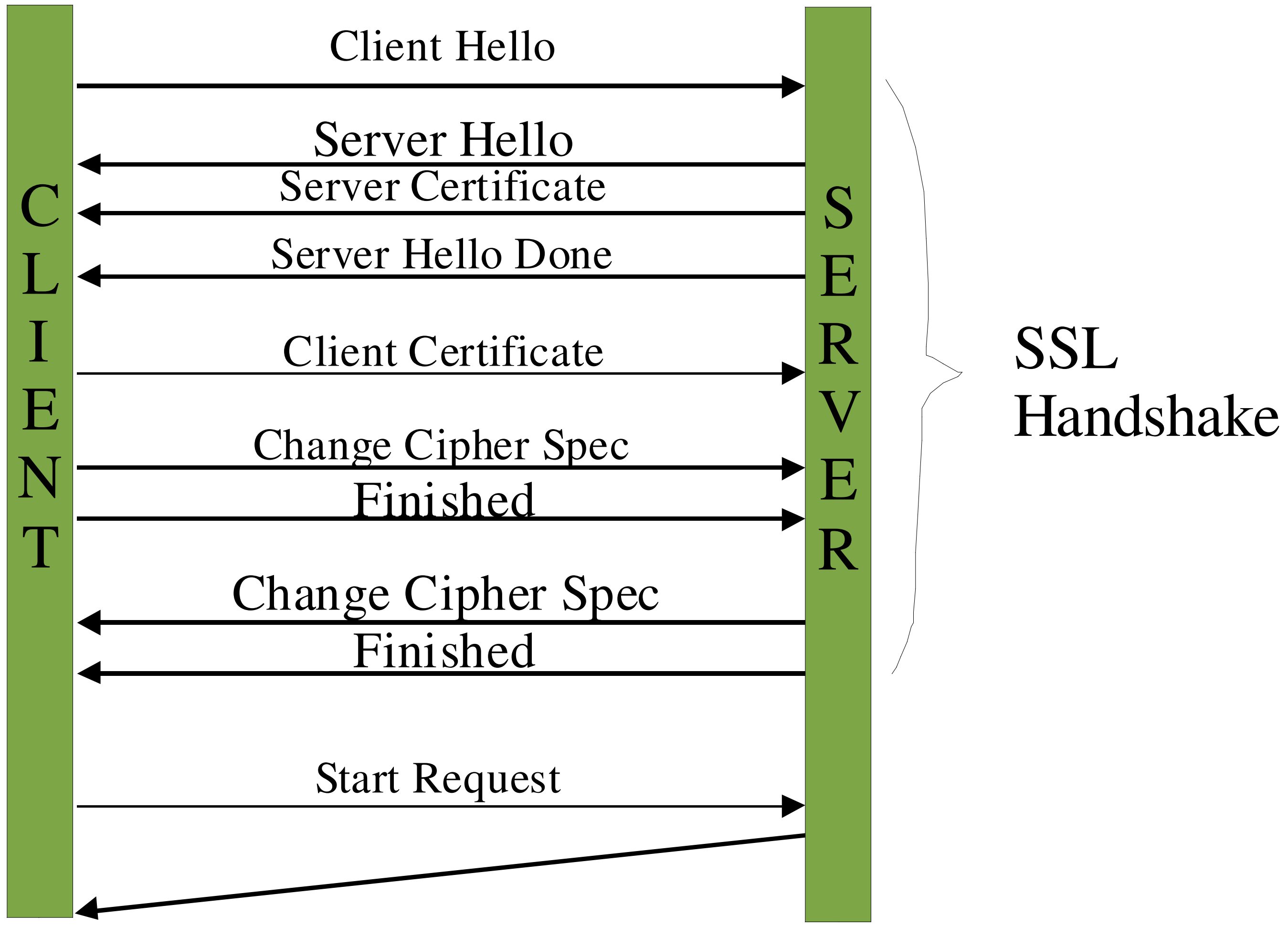
TLS Protocol



TLS Handshake

- Perform server and optionally client authentication
- Select cryptographic algorithms (ciphers) supported by client and server
- Generate and exchange session key
- Establish an encrypted connection

TLS Handshake Protocol



TLS and SSL Versions

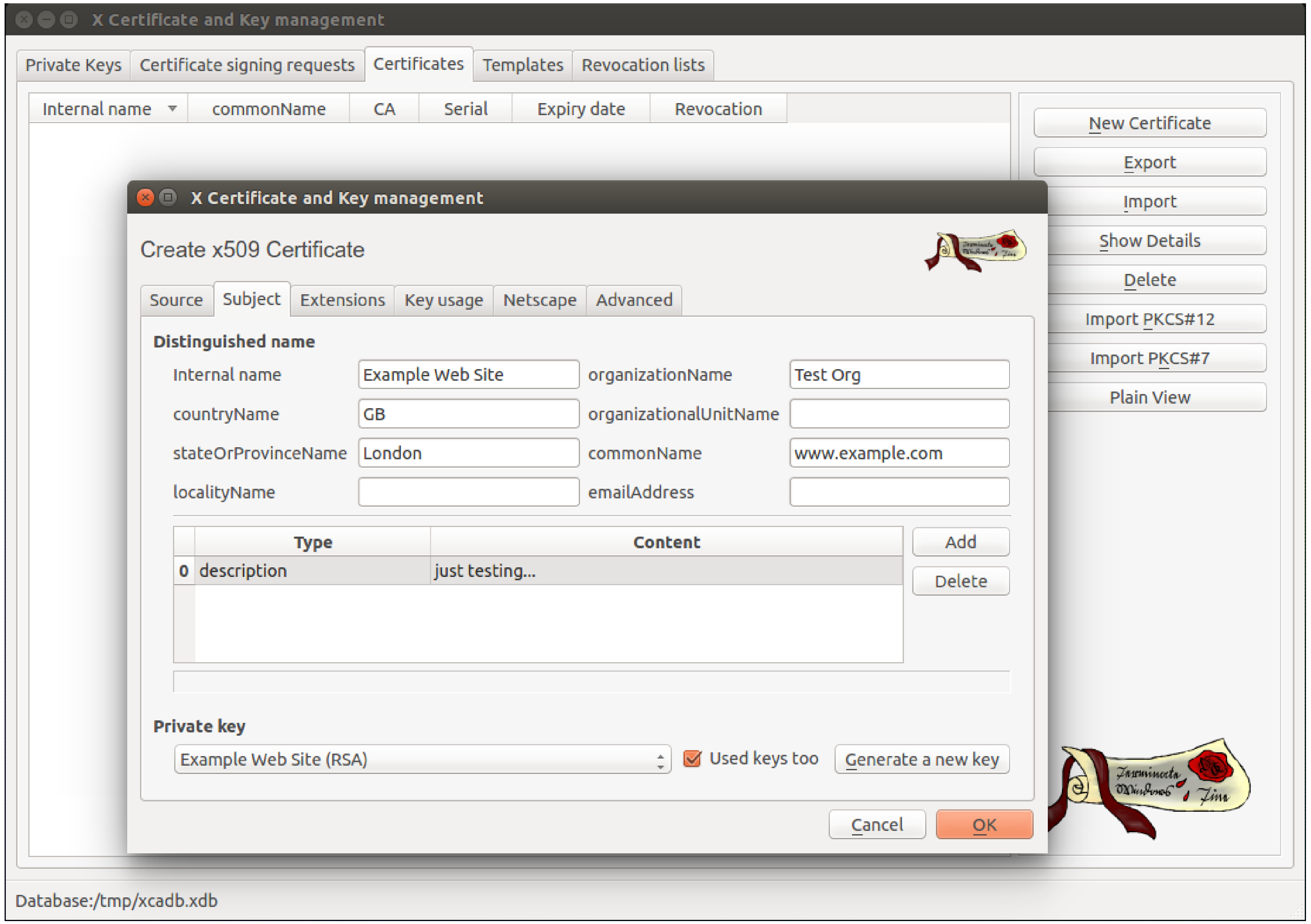
- SSL 2.0: original Netscape standard (no longer secure)
- SSL 3.0: revised version to fix various security vulnerabilities (no longer secure)

- TLS 1.0: first IETF standard
- TLS 1.1: protection against CBC attacks
- TLS 1.2: SSL 2.0 and MD5 no longer supported
- TLS 1.3: draft (as of July 2016)

XCA Tool

- Open Source
- Graphical user interface for OpenSSL

- `https://hohnstaedt.de/xca`



X Certificate and Key management

Create x509 Certificate

Distinguished name

Internal name	Example Web Site	organizationName	Test Org
countryName	GB	organizationalUnitName	
stateOrProvinceName	London	commonName	www.example.com
localityName		emailAddress	

	Type	Content
0	description	just testing...

Private key

Example Web Site (RSA) — ✓ Used keys too

Database:/tmp/xcadb.xdb

Restricting Client Certificates

- Restrict access based on client certificate details or any other SSL environment variable
  - `Require expr "<expression>"`

- Example: accept only certificate with specific common name

  -
    ```
    Require expr "{SSL_CLIENT_S_DN_CN} \
        in {'client.example.com', 'other.example.org'}"
    ```

Online Certificate Status Protocol

- OCSP issues:
  - End-user privacy
  - Efficiency
  - Does not mitigate against MITM attacks after server key compromise
- “OCSP Stapling” exists as an alternative to OCSP and should be enabled