Securing Communications with your Apache HTTP Server

Lars Eilebrecht
Lars@apache.org
About Me

• Lars Eilebrecht
• Independent IT Consultant
• Contributor to the Apache HTTP Server project since 1996
• Member of the ASF Security Team
• Co-founder and member of The Apache Software Foundation
• www.eilebrecht.net
Securing Communications with your Apache HTTP Server

Agenda

- Overview
- Cryptography Essentials
- X.509, Keys and Certificates
- SSL/TLS protocol
- Apache HTTP Server configuration
- Useful OpenSSL commands
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)
- Message Authentication Code (e.g., HMAC)
Keys and Certificates

- X.509: ITU-T standard (1988) for PKIs
- PKI: Public-Key Infrastructure
- CA: Certification Authority
- RA: Registration Authority
- CSR: Certificate Signing Request
- CRL: Certificate Revocation List
Common X.509 File Types and Extensions

- **PEM**: base64-encoded DER certificate(s) or public-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-----
MIIC2zCCAkSgAwIBAgIJANWZuQf40KVvMA0GCSqGSIb3DQEBBQUAMFMxCzAJBgNV
BAYTA1hYMQwwCgYDVQQIEwNYWFgxDDAKBgNVBAcTA1hYMQwwCgYDVQQLEwM2NjYxDDAKB
gNVBAMTAzY2NjAeFw0wODEwMDEyMzU1MDlaFw0w

BgNVHRMEBTADAQH/MA0GCSqGSIb3DQEBBQUAA4GBAFlaHQEXQdMVfvTay5x6fECa
QiefllN/69931EFmNX0mlpV8pFZ448PtoGlXiNd+rnfe2ttjPfmh4CXDN9q7NPUO
qntygrcWsGJxmVlu5s2q6KumrySEdqr+Da70zyed3Tfj/QYJfG1HAzfLCVZRKFQE
EuxxMbZd6XBxXenuZzn
-----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
  - **CN**: Common Name
  - **C**: Country (2-letter code)
  - **S**: State or province
  - **L**: Locality (e.g., City)
  - **O**: Organization
  - **OU**: Organizational Unit

- Example DN:  
  C=ES, L=Seville, 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/UC) 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

Internet Explorer

Chrome

Firefox

Safari

Opera
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 certificates from commercial CAs

• buy a certificate from a CA
  • domain-only, organization or extended validation
    (10€ up to 1000€)
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 almost all Web clients
- Service provided by Internet Security Research Group (ISRG) since April 2016 (non-profit organisation)
XCA Tool

- Open Source
- Graphical user interface for OpenSSL
Browser SSL Warnings

If the browser doesn't know the issuing CA it displays a warning to the user.
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 draft 1.3

- When people talk about SSL these days they actually mean TLS.
- An “SSL certificate” is an X.509 certificate for use with SSL/TLS.
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
Apache SSL/TLS Module - mod_ssl

• Included as default module since Apache HTTP Server version 2.0
• Supports TLS 1.0, 1.1, 1.2 protocols
• SSL 3.0 is still supported, but SSL 2.0 support was removed in Apache HTTP Server version 2.4
• Uses OpenSSL library
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
    (chmod 400 server.key)

• Enable SSL (per virtual host):
  - SSLEngine On
  - Listen 443
Intermediate CA Certificates

- add server and all intermediate certificates (sorted from leaf to root) to a single file and use `SSLCertificateFile`
- Multiple server certificates can be added to support different authentication algorithms (ECC, RSA, DSA, etc.)
- `SSLCertificateChainFile` became obsolete with version 2.4.8
SSL Ciphers and Protocols

- Define ciphers and protocol:
  - `SSLCipherSuite HIGH:MEDIUM:!MD5:!RC4`
  - `SSLHonorCipherOrder On`
  - `SSLProtocol All -SSLv2 -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 “!”)
SSL Ciphers

- Disable Camellia and Seed ciphers for HIPAA compliance:
  - `SSLCipherSuite \ HIGH:MEDIUM:!MD5:!RC4:!CAMELLIA:!SEED`

- Check which ciphers are enabled:
  - `openssl ciphers -v 'HIGH:MEDIUM:!MD5:!RC4'`

- Check “ciphers” man page for meanings of the various ciphers 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 SSL 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
- Restart Apache at least once a day to reduce the impact on forward secrecy.
- If forward secrecy is required session tickets should be disabled.
  - Disabling session tickets decreases performance!
SSL Virtual Hosting

- SSL can be enabled for any virtual host
- Name-based virtual hosts with SSL 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 the first virtual host or a “403 Forbidden” response if SSLStrictSNIVHostCheck is enabled
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
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
OCSP Stapling

• OCSP Stapling is known as the “TLS Certificate Status Request Extension”
  
  • SSLUseStapling on
  • SSLStaplingReturnResponderErrors off
  • SSLStaplingCache shmcb:/var/run/ocsp(128000)
Apache as an SSL Reverse Proxy

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

Securing Communications with your Apache HTTP Server
HTTPS with MS Internet Explorer

• SSL implementations in MSIE up to version 5 are broken, but market share for these versions is basically zero.

• The following directive from the Apache default configuration can be disabled:

  *BrowserMatch "MSIE [2-5]" \  
  nokeepalive ssl-unclean-shutdown \  
  downgrade-1.0 force-response-1.0
Test your SSL/TLS Server

Qualys SSL Labs:
https://www.ssllabs.com/ssltest/

High-Tech Bridge:
https://www.htbridge.com/ssl/
HTTP Strict Transport Security

- Web security policy mechanism to protect against protocol downgrade.
- Example header:
  - `Strict-Transport-Security: max-age=31536000`
- Once the browser has cached the header plain HTTP or untrusted certificates can no longer be used for that Web site.
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>`
Securing Communications with your Apache HTTP Server

Lars Eilebrecht
Lars@apache.org