Webseiten mit HTTPS bereitstellen und mit HSTS und HPKP sichern

Jörg Kastning

26. Oktober 2017

Inhalt

Inhalt

Inhalt

Der Prozess im Überblick

1. Erzeugung eines privaten Schlüssels
2. Generierung einer Zertifikatsanfrage (CSR)
3. Übermittlung des CSR an eine Zertifizierungsstelle (CA)
4. Ausstellung eines signierten Zertifikats durch die CA
5. Implementierung des privaten Schlüssels und des Zertifikats auf dem Webserver

Erzeugung eines privaten Schlüssels

```
/ tmp$ openssl genrsa -aes256 -out test.key 2048
Generating RSA private key, 2048 bit long modulus
.............................+++
.....................................+++
e is 65537 (0 x10001 )
Enter passphrase for test.key:
Verifying - Enter passphrase for test.key:
```

Achtung

Der private Schlüssel ist stets auf einem vertrauenswürdigen Host zu erzeugen. Er sollte niemals im Web auf den Seiten einer CA generiert werden.

Generierung einer Zertifikatsanfrage (CSR)

```
openssl req -batch -sha256 -new -key test.key -out test.csr \
-subj "/C=DE/L=Musterstadt/O=Musterfirma/OU=Musterabteilung/ \
CN=foo.example.com/emailAddress=foo@example.com"
```

Achtung

Auch dieser Schritt sollte auf einem vertrauenswürdigen Host ausgeführt werden. Der private Schlüssel darf den Host nicht verlassen.

Beispiel einer PKI

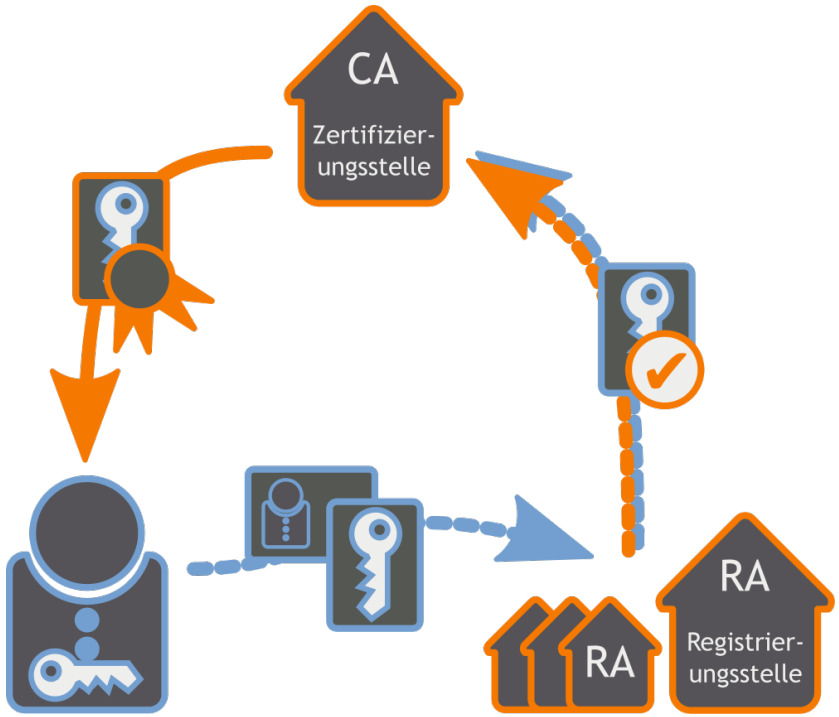

Abbildung: Verarbeitung eines CSR durch RA und CA

Beispiele für Zertifizierungsstellen

- DFN-PKI
- Let's Encrypt
- WoSign
- CAcert
- Symantec
- Hosting-Provider wie z. B. Strato, Hosteurope, 1&1, usw.
- Selbstsignierte Zertifikate (durch eigene CA)

Implementierung auf einem Webserver

- Die Dateien gehören **nicht** ins DocumentRoot

Implementierung auf einem Webserver

- Die Dateien gehören **nicht** ins DocumentRoot
- Zugriffsrechte auf den privaten Schlüssel soweit wie möglich beschränken

Implementierung auf einem Webserver

- Die Dateien gehören **nicht** ins DocumentRoot
- Zugriffsrechte auf den privaten Schlüssel soweit wie möglich beschränken
- Zertifikatskette mitausliefern

Die Zertifikatskette

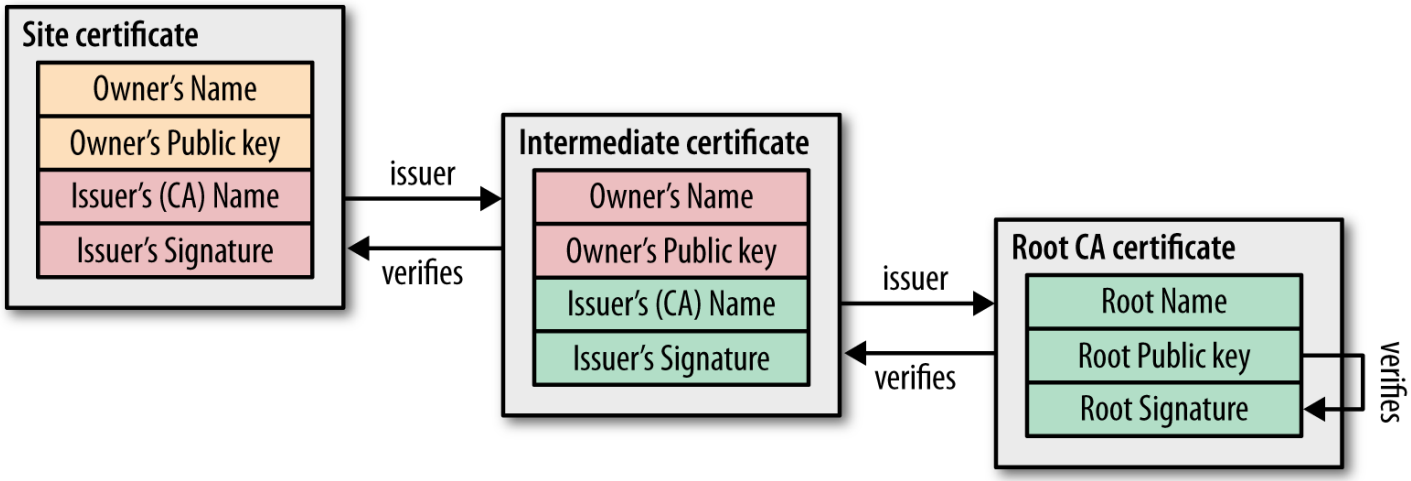

Abbildung: Zertifikatskette vom Leaf Certificate bis zum Root Certificate im Trust Store des Webbrowsers

Konfiguration einer TLS/SSL-Webseite

Ein Beispiel für Apache 2.4

```
LoadModule ssl_module modules/mod_ssl.so

Listen 443
<VirtualHost *:443>
ServerName www.example.com
SSLEngine on
SSLCertificateFile "/path/to/www.example.com.cert"
SSLCertificateChainFile "/usr/local/apache2/conf/ssl.crt/ca.crt"
SSLCertificateKeyFile "/path/to/www.example.com.key"
</VirtualHost>
```

Quelle: `https://httpd.apache.org/docs/2.4/`

Konfiguration einer TLS/SSL-Webseite

Ein Beispiel für NGINX

```
server {
  listen 443 ssl;
  server_name www.example.com
  ssl_certificate /path/to/www.example.com.cert
  ssl_certificate_key /path/to/www.example.com.key
  ...
  }
```

Quelle: `http://nginx.org/en/docs/http/configuring_https_servers.html`

Gefahr durch Man-In-The-Middle-Angriffe

Es ist Vorsicht geboten

Obwohl TLS/SSL-Verschlüsselung implementiert wurde, sind Man-In-The-Middle-Angriffe weiterhin möglich.

Gefahr durch Man-In-The-Middle-Angriffe

Es ist Vorsicht geboten

Obwohl TLS/SSL-Verschlüsselung implementiert wurde, sind Man-In-The-Middle-Angriffe weiterhin möglich.
Denn der Webbrowser weiß nicht, ob TLS/SSL auf einer Webseite implementiert sind oder nicht.

Schwachstelle im Design

- Es gibt hunderte von „vertrauenswürdigen“ CAs, denen Browser und E-Mail-Clients vertrauen
- Jede dieser CAs kann Zertifikate für beliebige Domains ausstellen und die Browser und E-Mail-Clients vertrauen ihnen
- Man-In-The-Middle-Angriffe sind so auch auf HTTPS-Verbindungen möglich

HTTP Strict Transport Security

Welchen Nutzen bietet HSTS?

- Dem Webbrowser wird mitgeteilt, dass eine Seite über HTTPS erreichbar ist
- Die URL wird transparent zu einer `https`-URL umgeschrieben
- Der Webbrowser wird angewiesen, diese Seite ausschließlich über HTTPS aufzurufen

HTTP Strict Transport Security

Wie implementiert man HSTS?

```
<VirtualHost www.example.com:443>
Header always set Strict-Transport-Security "max-age=63072000; \
includeSubdomains;"
</VirtualHost>
```

```
server {
  listen 443 ssl;
  add_header Strict-Transport-Security "max-age=63072000; \
  includeSubdomains; ";
}
```

HTTP Strict Transport Security

Vor- und Nachteile von HSTS

Vorteile

- Leicht zu konfigurieren
- Verhindert MITM-Angriffe auf HTTP-Verbindungen

Nachteile

- Problematisch bei Mixed-Content
- MITM-Angriff mit gefälschtem TLS/SSL-Zertifikat weiterhin möglich

Pinning - Wie funktioniert das?

- Der öffentliche Schlüssel eines Zertifikats wird „festgenagelt“
- Ein Hash-Wert wird an den Browser übermittelt und von diesem gespeichert

Pinning - Wie funktioniert das?

- Der öffentliche Schlüssel eines Zertifikats wird „festgenagelt“
- Ein Hash-Wert wird an den Browser übermittelt und von diesem gespeichert
- Dadurch kann der Browser das Zertifikat wiedererkennen
- Stimmt der berechnete Hash nicht mit dem gespeicherten Wert überein, wird der Zugriff auf die Webseite verweigert

Implementierung von HPKP
Vorüberlegungen

- Welcher öffentliche Schlüssel soll „gepinnt“ werden?

Implementierung von HPKP
Vorüberlegungen

- Welcher öffentliche Schlüssel soll „gepinnt“ werden?
- Tipp: Pinnt das Serverzertifikat!

Implementierung von HPKP
Vorüberlegungen

- Welcher öffentliche Schlüssel soll „gepinnt“ werden?
- Tipp: Pinnt das Serverzertifikat!
- Was tun, wenn das Zertifikat kompromittiert wurde?

Implementierung von HPKP
Vorüberlegungen

- Welcher öffentliche Schlüssel soll „gepinnt“ werden?
- Tipp: Pinnt das Serverzertifikat!
- Was tun, wenn das Zertifikat kompromittiert wurde?
- Einen vorbereiteten Backup-PIN nutzen!

Implementierung von HPKP

PINs berechnen

```
openssl x509 -noout -in certificate.pem -pubkey | openssl asn1parse \
  -noout -inform pem - out public.key
openssl dgst -sha256 -binary public.key | openssl enc -base64
```

```
openssl req -noout -in example.com.csr -pubkey | openssl asn1parse \
  -noout -inform pem -out example.com_csr.key
openssl dgst -sha256 -binary example.com_csr.key | \
  openssl enc -base64
```

Implementierung von HPKP am Beispiel von NGINX

```
server {
  listen 443 ssl;
  server_name www.example.com
  ssl_certificate /path/to/www.example.com.cert
  ssl_certificate_key /path/to/www.example.com.key

  add_header Public-Key-PINs 'pin-sha256="PRIMARY-PIN"; \
pin-sha256="BACKUP-PIN"; max-age=300; includeSubDomains';
}
```

Nutzen der Public Key Pinning Extension for HTTP (HPKP)

Vorteile

- Kann MITM-Angriffe mit gefälschten Zertifikaten aufdecken
- Verhindert den Zugriff auf die betroffene Seite

Nutzen der Public Key Pinning Extension for HTTP (HPKP)

Vorteile

- Kann MITM-Angriffe mit gefälschten Zertifikaten aufdecken
- Verhindert den Zugriff auf die betroffene Seite

Nachteile

- Aufwendig zu implementieren
- Kann eine Webseite unerreichbar machen

Fazit

Mit HSTS und HPKP stehen wirksame Mittel zur Verfügung, um die Angriffsfläche auf HTTPS-Verbindungen zu verkleinern.

Fazit

Mit HSTS und HPKP stehen wirksame Mittel zur Verfügung, um die Angriffsfläche auf HTTPS-Verbindungen zu verkleinern.

Weiterführende Informationen findet ihr im TLS-Kochbuch unter: `https://www.my-it-brain.de/wordpress/mein-tls-kochbuch/`