SSL certifikáty

Zde popisuji, jak získat, nainstalovat a používat služby, zabezpečené pomocí SSL - cs:SSL.

Generování requestu

Aby to celé mělo smysl, musím si svůj certifikát nechat podepsat nějakou důvěryhodnou certifikační autoritou, kterou znají prohlížeče a poštovní klienti.

V krajním případě můžeme použít také vlastní certifikační autoritu nebo si vygenerovat tzv. self-signed certifikát, ale v takovém případě budou prohlížeče zobrazovat nejrůznější varování a nedůvěryhodném certifikátu a to nás bude dost zdržovat a obtěžovat, proto budu dále počítat pouze s podepsaným certifikátem důvěryhodnou autoritou.

Pro získání podepsaného certifikátu je nejprve nutné vygenerovat tajný klíč. Tajný klíč se zpravidla umisťuje do adresáře **/etc/ssl/private**. Doporučuji překontrolovat přístupová práva, k tajnému klíči by měl mít přístup pouze uživatel root a skupina sslcert

> Vygenerované certifikáty je dobré ukládat podle jména domény, např. vasedomena-cz.key Přípona souboru je pak určena dle významu klíče
>
> - key - tajný klíč (necháváme na serveru)
> - crt - veřejný klíč (dostaneme podepsaný od certifikační autority na základě žádosti)
> - csr - žádost o certifikát. Žádost si vygeneruje k již vygenerovánu tajnému klíči a vzniklý soubor odešleme certifikační autoritě
> - pem - tento soubor obsahuje zpravidla jak tajný klíč tak k němu odpovídající veřejný klíč včetně veřejných klíčů a root certifikátu autority, která veřejný klíč vystavila. Tento formát se používá pro služby, které nepodporují oddělené úložiště pro tajný a veřejný klíč např. imap nebo postfix.

1. V adresáři */etc/ssl/private* vygenerujeme tajný klíč

```
openssl genrsa -out nazevdomeny.key 2048
```

2. Nyní ke klíči připravíme žádost o certifikát

```
openssl req -new -key nazevdomeny.key -out nazevdomeny.csr
```

3. Soubor

```
nazevdomeny.csr
```

odešleme certifikační autoritě (dále probíhá podle podmínek příslušné autority)

4. Jakmile je proces u autority hotov, obdržíme podepsaný certifikát. Tento certifikát následně uložíme ve tvaru **nazevdomeny.crt** do adresáře */etc/ssl/certs*. Ze stránek naší certifikační autority si dále musíme stáhnou ChainFile a root certifikát naší autority.

V našem případě využíváme certifikační autoritu StartSSL, ale postupy zde uvedené se dají aplikovat bez problému na ostatní autority.

V našem případě tedy stáhneme do adresáře */etc/ssl/certs*soubory **sub.class2.server.ca.pem** a **ca.pem**. Následně vytvoříme společný soubor se všemi klíči (toto dělat nemusíte, pokud SSL certifikát použijete pouze v Apachi)

```
cat /etc/ssl/private/vasedomena.key /etc/ssl/certs/vasedomena.crt
/etc/ssl/certs/sub.class2.server.ca.pem /etc/ssl/certs/ca.pem >
/etc/ssl/private/vasedomena.pem
```

Nastavení Apache

Zpravidla se používá pro každou doménu, kterou chceme zabezpečit, zvláštní IP adresa. Od verze Apache 2.2.12 a OpenSSL verze 0.9.8f je možné využívat jedinou IP adresu pro více domén s různými SSL certifikáty tzv. SNI (Server Name Indication - http://wiki.apache.org/httpd/NameBasedSSLVHostsWithSNI)

V Apachi je potřeba povolit mod_ssl

```
a2enmod ssl
```

Doporučené nastavení mod_ssl

Upravte nebo přidejte tyto volby v konfiguračním souboru /etc/apache2/mods-available/ssl.conf

```
SSLCipherSuite HIGH:MEDIUM:!aNULL:!MD5:!RC4
SSLHonorCipherOrder on
SSLProtocol -All +TLSv1 +TLSv1.1 +TLSv1.2
```

Konfigurace virtual hostu

Příklad konfigurace Apache virtualhostu. Věnujte pozornost nastavení zejména položek

- VirtualHost - Uvadíme IP adresu, na které chceme SSL provozovat
- ServerName a ServerAlias - Uvádíme doménu, na kterou je vystaven SSL certifikát. Pokud certifikát obsahuje alternativní jména, uvedeme je dále jako ServerAlias
- SSLCertificateFile - podepsaný certifikát
- SSLCertificateKeyFile - náš tajný klíč
- SSLCertificateChainFile a SSLCACertificateFile - ChainFile a Root certifikát, oba soubory ke stažení u příslušné certifikační autority.

/etc/apache2/ports.conf

```
    Listen 443
    NameVirtualHost 123.456.78.90:443
```

/etc/apache2/sites-available/default-ssl

```
<VirtualHost 123.456.78.90:443>
        ServerAdmin admin@vasedomena
        ServerName vasedomena

        DocumentRoot /var/www/ssl
        <Directory /var/www/ssl>
                Options FollowSymLinks MultiViews
                AllowOverride None
                Order allow,deny
                allow from all
        </Directory>

        ErrorLog /var/log/apache2/ssl_error.log
        LogLevel warn
        CustomLog /var/log/apache2/ssl_access.log combined

        #   SSL Engine Switch:
        #   Enable/Disable SSL for this virtual host.
        SSLEngine on

        #   A self-signed (snakeoil) certificate can be created by
installing
        #   the ssl-cert package. See
        #   /usr/share/doc/apache2.2-common/README.Debian.gz for more info.
        #   If both key and certificate are stored in the same file, only
the
        #   SSLCertificateFile directive is needed.
        SSLCertificateFile    /etc/ssl/certs/vasedomena.crt
        SSLCertificateKeyFile /etc/ssl/private/vasedomena.key

        #   Server Certificate Chain:
        #   Point SSLCertificateChainFile at a file containing the
        #   concatenation of PEM encoded CA certificates which form the
        #   certificate chain for the server certificate. Alternatively
        #   the referenced file can be the same as SSLCertificateFile
        #   when the CA certificates are directly appended to the server
        #   certificate for convinience.
        SSLCertificateChainFile /etc/ssl/certs/sub.class2.server.ca.pem

        #   Certificate Authority (CA):
        #   Set the CA certificate verification path where to find CA
        #   certificates for client authentication or alternatively one
        #   huge file containing all of them (file must be PEM encoded)
        #   Note: Inside SSLCACertificatePath you need hash symlinks
```

```
        #         to point to the certificate files. Use the provided
        #         Makefile to update the hash symlinks after changes.
        SSLCACertificatePath /etc/ssl/certs/
        SSLCACertificateFile /etc/ssl/certs/ca.pem
```

Pokud neuvedete SSLCertificateChainFile a SSLCACertificateFile, bude neustále prohlížeč hlásit potíže s certifikátem.

Použití SNI

Pokud např. nemáme dostatek veřejných IP adres pro každou SSL doménu, můžeme využít SNI, tzn. certifikáty se budou brát podle jména domény a všechno poběží s jednou IP adresou. Nevýhoda tohoto řešení je, že starší prohlížeče nebo většina textových klientů bude mít potíže se zobrazením. Podrobnosti o SNI lze získat na cs:SNI

V současné době umí SNI pouze tyto prohlížeče:

- Mozilla Firefox 2.0 nebo novější
- Opera 8.0 nebo novější (musí být povolen protokol TLS)
- Internet Explorer 7 (Vista, ne XP) nebo novější
- Google Chrome
- Safari 3.2.1 Mac OS X 10.5.6

Z hlediska SNI je nastavení virtualhostu uplně stejné, jako v předchozím případě, akorát se na jednu IP adresu na portu 443 binduje libovolný počet virtualů s ruzným ServerName a svojí definicí certifikátů.

POZOR: Pokud klient NEPODPORUJE SNI, pak mu Apache v uvedeném nastavení na portu 443 bude servírovat první virtualhost, který načte v konfiguraci, proto doporučuji dát jako první virtualhost nějakou základní webovkou, přístupnou pro všechny. Pokud máte každý virtualhost ve vlastním souboru, pak konfigurační soubor, který chcete servírovat klientům bez SNI, pojmenujte tak, aby byl v adresáři sites-enabled jako první v pořadí !

Courier IMAP / POP

Courier vyžaduje mít uložené všechny certifikáty spolu s klíčem v jednom souboru. Použijeme tedy náš vytvořený soubor **/etc/ssl/private/vasedomena.pem** a vložíme odkaz do příslušných konfiguráků:

V souboru **imapd-ssl** a **pop3d-ssl** najdete příslušnou volbu a upravíme jí takto:

```
TLS_CERTFILE=/etc/ssl/private/vasedomena.pem
```

Postfix

Postfix sice podporuje uložení certifikátu a klíče v odděleném souboru, ale je v tomto případě problém používat spolu s root certifikátem autority, takže potom poštovní klient při odesílání hlásí nedůvěryhodný certifikát. Doporučuji tedy použít opět náš společný soubor **/etc/ssl/private/vasedomena.pem**

V konfiguraci postfixe je potřeba nastavit v souboru **main.cf** v sekci nastavení TLS parametru následující volby

```
smtpd_tls_cert_file = /etc/ssl/private/vasedomena.pem
smtpd_tls_key_file = $smtpd_tls_cert_file
```

Lighttpd

Editujeme soubor **/etc/lighttpd/conf-available/10-ssl.conf**. Nasledne je potreba udelat alias v adresari conf-enabled

```
$SERVER["socket"] == "0.0.0.0:443" {
        ssl.engine  = "enable"
        ssl.pemfile = "/etc/ssl/private/vasedomena.pem"
```

Testování nainstalovaného certifikátu

Nainstalovaný certifikát můžeme otestovat buď přímo v prohlížeči tj. https://vasedomena a pokud prohlížeč nebude zobrazovat žádná varování a zároveň v adresním řádku bude vidět zelený zameček (chromium), modrý proužek se jménem domény (firefox) apod. je vše v pořádku nainstalováno.

Ostatní služby můžeme testovat pomocí příkazu openssl - OpenSSL. V případě postfixu použijeme tento příkaz

```
openssl s_client -starttls smtp -crlf -connect vasedomena:25
```

Self-Signed certifikát

Někdy nám stačí certifikát, podepsaný sám sebou. Např. u méně důležitých služeb, kde potřebujeme šifrovat, ale nepotřebujeme ověřený certifikát, vydaný autoritou. Vygenerování takového certifikátu probíhá obdobně, jako v předchozím případě:

1. V adresáři */etc/ssl/private* vygenerujeme tajný klíč

```
openssl genrsa -out nazevdomeny.key 2048
```

2. Nyní ke klíči připravíme žádost o certifikát

```
openssl req -new -key nazevdomeny.key -out nazevdomeny.csr
```

3. Následně vytvoříme certifikát, který podepíšeme stejný klíčem. Parametr **days** můžeme nastavit libovolně a určuje délku platnosti certifikátu.

```
openssl x509 -req -days 365 -in nazevdomeny.csr -signkey nazevdomeny.key -
out nazevdomeny.crt
```

A máme hotovo.

Chyby v knihovně OpenSSL

Začátkem dubna 2014 se objevila nepříjemná chyba v knihovně OpenSSL, která umožnuje ve verzi 1.0.1 útocníkům zjistit tajné klíče SSL certifikátů a následně dešifrovat hesla.

Chyba se týká verze OpenSSL 1.0.1 - 1.0.1f a je opravena až od verze 1.0.1g

- OpenSSL 1.0.1 through 1.0.1f (inclusive) are vulnerable
- OpenSSL 1.0.1g is NOT vulnerable
- OpenSSL 1.0.0 branch is NOT vulnerable
- OpenSSL 0.9.8 branch is NOT vulnerable

Tyto verze OS jsou chybou dotčeni:

- Debian Wheezy (stable), OpenSSL 1.0.1e-2+deb7u4 (**opravuje balik 1.0.1e-2+deb7u7**)
- Ubuntu 12.04.4 LTS, OpenSSL 1.0.1-4ubuntu5.11
- CentOS 6.5, OpenSSL 1.0.1e-15 (**opravuje balik 1.0.1e-16**)
- Fedora 18, OpenSSL 1.0.1e-4
- OpenBSD 5.3 (OpenSSL 1.0.1c 10 May 2012) and 5.4 (OpenSSL 1.0.1c 10 May 2012)
- FreeBSD 10.0 - OpenSSL 1.0.1e 11 Feb 2013
- NetBSD 5.0.2 (OpenSSL 1.0.1e)
- OpenSUSE 12.2 (OpenSSL 1.0.1c)

Je nutné o nejdříve provést aktualizaci uvedených systémů a po provedené aktualizaci je nutno provést regenerování všech SSL certifikátů na serveru a změne hesel v aplikacích, které SSL připojení používají !!

Starsi verze OS jsou bezchybne:

- Debian Squeeze (oldstable), OpenSSL 0.9.8o-4squeeze14
- SUSE Linux Enterprise Server
- FreeBSD 8.4 - OpenSSL 0.9.8y 5 Feb 2013
- FreeBSD 9.2 - OpenSSL 0.9.8y 5 Feb 2013
- FreeBSD 10.0p1 - OpenSSL 1.0.1g (At 8 Apr 18:27:46 2014 UTC)
- FreeBSD Ports - OpenSSL 1.0.1g (At 7 Apr 21:46:40 2014 UTC)

Diagnostika

Otestovat je možno aktuální verzí nmap - http://nmap.org/download.html

```
nmap -sV --script=ssl-heartbleed <adresa>
```

Podrobnosti o celé problematice je možné najít zde - http://heartbleed.com/

Další zajímavé informace:

- http://blog.nic.cz/2014/04/14/jak-heartbleed-poukazal-na-slabiny-certifikacnich-autorit/
- http://www.root.cz/clanky/heartbleed-bug-vazna-zranitelnost-v-openssl/#ic=kolotoc-header&icc=heartbleed-bug-vazna-zranitelnost-v-openssl
- http://blog.existentialize.com/diagnosis-of-the-openssl-heartbleed-bug.html