SSL certifikáty

Zde popisuji, jak získat, nainstalovat a používat služby, zabezpečené pomocí TLS - cs:TLS. Pokud máš problém s TLS a nevíš proč, tak použij testy! SSLLabs SSLTest pro HTTPS, pripadne CheckTLS pro mailservery.

CertBot

Priklad pro pouziti s apachem, s automatickym restartem sluzeb

V tomto pripade mame funkcni webserver, ktery serviruje overovaci retezce z webrootu `/srv/http`. Pomoci **renew-hook** nastavime prikaz, ktery pokazde kdyz je vystaven nebo obnoven certifikat restartuje sluzby, ktere ho vyuzivaji. Vyhoda je, ze sluzby jsou restartovany jen pokud certifikat opravdu vyprsi a je obnoven.

```
certbot certonly --webroot --webroot-path /srv/http/ \
    --renew-hook "systemctl reload-or-try-restart apache2 postfix dovecot" \
    --rsa-key-size 4096 \
    -d harvie.cz -d www.harvie.cz -d blog.harvie.cz -d wiki.harvie.cz
```

U slozitejsich setupu apache s rewritama muze bejt dulezity presmerovat vsechny pozadavky na well known:

```
RewriteEngine On

#ACME letsencrypt well known
RewriteCond %{REQUEST_URI} ^/.well-known/(.*)
RewriteRule .well-known/(.*) /srv/http/.well-known/$1 [L]
```

Priklad pro pouziti bez soucinnosti stavajiciho webserveru

V tomhle setupu neni potreba stavajici webserver. To se hodi pokud vubec neni nainstalovany, nebo ho nelze nakonfigurovat aby serviroval overovaci retezce. Certbot si spusti vlastni integrovany webserver. Jen je potreba pomoci **pre-hook** zastavit stavajici webserver (pokud na serveru bezi), aby neblokoval HTTP(S) porty. A pomoci **post-hook** ho opet spustit, potom co certbot porty opet uvolni. Sluzby ktere potrebuji certifikat vyuzivat opet restartujeme pomoci **renew-hook**, aby si ho nacetly.

Toto neni preferovana metoda na webserverech, protoze zpusobuje vypadek webserveru v radu jednotek az desitek sekund (tj. po dobu probihajiciho overeni ze strany serveru letsencryptu). Tj. pouzivat jen tam, kde nebezi webserver, nebo bezi nejaky jednoucelovy webserver, ktery neumi servirovat externi soubory.

```
certbot certonly --standalone \
```

```
    --pre-hook "systemctl stop apache2" \
    --post-hook "systemctl start apache2" \
    --renew-hook "systemctl reload-or-try-restart apache2 postfix dovecot" \
    --rsa-key-size 4096 \
    -d harvie.cz -d www.harvie.cz -d blog.harvie.cz -d wiki.harvie.cz
```

Automaticka obnova certifikatu

Obnova certifikatu se provadi prikazem `certbot renew`, nicmene to neni potreba delat, ani davat do cronu. Soucasti instalace certbotu totiz uz je soubor `/etc/cron.d/certbot`, ktery se o vse postara. Jen je potreba mit spravne nastavene hooky, aby se korektne restartovaly sluzby a vzdy po obnoveni nacetly novy certifikat. Hooky se zadavaji/upravuji pri prvotnim vytvareni certifikatu prikazem `certbot certonly …` (viz. vyse). Nastaveni hooku a dalsich parametru procesu obnovy certifikatu lze overit v souborech `/etc/letsencrypt/renewal/*.conf` Napriklad renew hook je tam uveden v sekci `[renewalparams]` jako `renew_hook = systemctl reload-or-try-restart apache2 postfix dovecot`.

Instalace certbota

Debian wheezy/wheezy-lts

- je potřeba přidat balíčky z wheezy-backports. Dále je potřeba stáhnout balíček "certbot 0.8", který je funkční ve wheezy - např. odtud (přiznám se, že jsem zapomněl kde přesně jsem ho stáhnul, proto neuvádím zdroj)

```
apt-get install -t wheezy-backports augeas-lenses libaugeas0
apt-get install dialog
dpkg -i certbot_0.8.1_amd64.deb
```

Debian Jessie

- je potřeba přidat balíčky z jessie-backports (ten jiz certbot obsahuje), ostatní by se mělo nainstalovat přes závislosti.

```
apt-get install -t jessie-backports certbot
```

Nastavení služeb

Apache2

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
```

```
SSLStaplingCache shmcb:/tmp/stapling_cache(2097152)
<VirtualHost *:443>
        SSLEngine on
    #SSLProtocol all
    #SSLCipherSuite HIGH:MEDIUM
    #SSLCipherSuite "HIGH:!aNULL:!MD5:!3DES:!CAMELLIA"
    SSLCipherSuite "HIGH"
    SSLHonorCipherOrder on
    SSLCompression off
    SSLUseStapling on

    SSLCertificateFile /etc/letsencrypt/live/harvie.cz/fullchain.pem
    SSLCertificateKeyFile /etc/letsencrypt/live/harvie.cz/privkey.pem

    #Obsah sajty (spolecny pro HTTP i HTTPS):
    Include /etc/apache2/site.conf
</VirtualHost>
```

Použití SNI

Pozor, SNI nedává dnes až tak moc velký smysl, protože letsencrypt dokáže vystavit společný certifikát až pro 100 různých domén. Takže pokud nejde o velký sdílený hosting, tak se často obejdeme i bez SNI. Ale do budoucna se snad bude podpora SNI zlepšovat, takže nebude nutné se mu vyhybat.

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
TLS_PROTOCOL="TLS1_1:TLS1"
TLS_CIPHER_LIST="TLSv1:!SSLv2:!SSLv3:HIGH:!LOW:!MEDIUM:!EXP:!NULL@STRENGTH"
```

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
smtpd_tls_mandatory_exclude_ciphers = aNULL, MD5
smtpd_tls_mandatory_protocols = !SSLv2, !SSLv3
smtp_tls_mandatory_exclude_ciphers = aNULL, MD5
smtp_tls_mandatory_protocols = !SSLv2, !SSLv3
lmtp_tls_mandatory_exclude_ciphers = aNULL, MD5
lmtp_tls_mandatory_protocols = !SSLv2, !SSLv3
```

Dovecot

```
ssl_cert_file = /etc/ssl/private/vasedomena.pem
ssl_key_file = /etc/ssl/private/vasedomena.pem
#ssl_protocols = !SSLv2 !SSLv3
```

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"
```

Překonané a nepotřebné informace

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
```

Pro komplexní otestování korektního nastavení SSL na serveru, včetně diagnostiky slabých šifer je možné provést pomocí skriptu **testssl.sh**. jeho aktuální verzi naleznete na stejnojmenných stránkách testssl.sh

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 -sha256 -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~~,[1] / V současné době přecházíme u standardních služeb na Let's Encrypt - popis implementace viz. dále. Výše uvedený postup je možné použít pro vygenerování žádosti o certifikát pro jakoukoliv certifikační autoritu, kterou si vyberete - (Let's Encrypt zatím neumí wildcard certifikaty, takže pokud se bez něj neobejdete, je nutné si certifikát u některé z jiných autorit zakoupit)

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
```

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.

Ověření příslušnosti dvojice private a public key k sobě

openssl rsa -noout -modulus -in test.key | openssl md5

openssl x509 -noout -modulus -in test.crt | openssl md5

Chyby v knihovně OpenSSL

Začátkem dubna 2014 se objevila nepříjmená 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

[1)]

K Opuštění StartSSL nás vede jeho postupné vyřazení z důvěryhodných certifikačních autorit - podrobnosti např. zde Mozilla přestane důvěřovat certifikátům od StartSSL