Multi CA authentikáció a weben

1. Szcenário

A munka során sokszor olyan helyezetekben ütközhetünk, ahol a pénzt kreativitással kell pótolni, viszont a biztonságon nem szabad rontani. Ezek a helyzetek roppant kínosak tudnak lenni esetenként egykét jobban szívén viselő IT-Security szakember számára, főleg ha nekik kell elvégezni a munkát és nem mást felügyelni.

Jelen helyzetben a probléma a következő:

- adott egy szerver amin számos weboldal fut
- ehhez a szerverhez igényeltünk egy hitelesített CA által aláírt certificate-et amivel probléma nélkül megjeleníthetőek bármilyen böngészőben ssl/tls-el felvértezett kapcsolaton keresztül
- szükséges üzemeltetni olyan aloldalakat/aldomaineket ahol a titkosított csatorna nem elég, kell "névre szóló" kliens certificate is az oldal megjelenítéséhez.

2. Megvalósítás

A megvalósítás első körben egyszerűnek tűnhet, viszont kellő tapasztalat nélkül, jó adagnyi SSL-howto birtokában is el lehet vele szórakozni, majd egy idő utána ideg bajt kapva falnak rohanni.
Természetesen a szerver, weboldalak konfigurációját és a hiteles CA-tól való aláírt certificate igénylését, beállítását kihagyva rögtön a lényegre is térünk, hiszek ezekről számos, egzaktul használható how-to készült.

Jelen pillanatban egy felkonfigolt apache webserver áll a rendelkezésünkre, ahova beillesztve az aláírt certificate-et a https://www.domain.hu -t gond nélkül elérhetjük. A böngésző egy szót sem szól, szimplán megjelenik a lakat és titkosítva lesz a csatorna.
A certificate-et a *.domain.hu-ra rendeltük meg, így bármilyen aldomaint korlátok nélkül használhatunk vele, anélkül hogy akármilyen hibát dobna a böngészőnk.

Feladatunk konkrétabban annyi, hogy még a www aldomain alatt az adatok hiba, bármilyen külső beavatkozás nélkül titkosított csatornán bárki számára elérhető legyen, mellette szeretnénk egy client.domain.hu aldomaint is üzemeltetni, amit csak azok érhetnek el akik az általunk kiállított kliens certificate-et installálták a böngésző, operációs rendszer konténerébe.

3. Saját CA készítése

```
# mkdir -p ca/certs
# mkdir ca/private
# mkdir ca/requests
# mkdir ca/crl
# mkdir ca/newcerts
# touch ca/index.txt
```

```
# echo 01 > ca/serial
```

openssl.cnf készítése a ca/ könyvtár alá, jó példa található erre itt:
http://www.securityfocus.com/infocus/1823
Minimum kulcsméretet érdemes 2048-ra venni, hiszen nem rég találtak elvi sebezhetőséget a 768-as méretű kulcsokra, így jobb távolra tervezni mint közelre.

CA kulcs generálása:

```
# openssl req -config ca/openssl.cnf -new -x509  -nodes -days 7304 -sha1 -
newkey rsa:2048 -keyout ca/private/ca.key -out ca/ca.crt -subj '/O=Owner/
OU=Owner Unit'
```

-*nodes* paraméternél nem szükséges a certhez jelszót használni, vagyis minden egyes CA műveletnél megspórolunk egy jelszó begépelést. Természetesen automatizálni is lehet ezt, viszont akkor ott vagyunk ahol a gát szakad.

4. Kliens kulcsok generálása

```
# openssl req -new -sha1 -newkey rsa:2048 -nodes -keyout client1.key -out
request1.pem -subj '/O=Owner/CN=client.domain.hu'
# cp request1.pem ca/requests/
# openssl ca -config ca/openssl.cnf -policy policy_anything -extensions
ssl_client -out signed1.pem -infiles ca/requests/request1.pem
# openssl pkcs12 -export -clcerts -in signed1.pem -inkey client1.key -out
client1.p12
```

Ezzel előállítottunk egy kliens certificate-et, amit titkosított csatornán szükséges eljuttatni a felhasználóhoz, így megóvva integritását, authentikusságát. A felhasználónak innentől egy dolga van, ezt importálni a megfelelő tárolóba a későbbi használathoz.

5. Kliens kulcsok visszavonása

```
# openssl ca -config ca/openssl.cnf -revoke ca/newcerts/01.pem
# openssl ca -config ca/openssl.cnf -gencrl -crlexts crl_ext -md sha1 -out
crl.pem
```

01-es ID-vel jelzett kulcs visszavonása, majd lista generálása amit szükséges átmásolni a későbbiekben megadott helyre.

6. Apache beállítása

Debian és hasonló strukturalitást követő operációs rendszeren szerencsése szétpartícionálták a konfigurációs fileokat az apache csomagban, így könnyebben szerkeszthető és található meg bármi.

Első körben az ssl.conf filenak a következőket kell tartalmaznia:

```
<IfModule mod_ssl.c>
Listen 0.0.0.0:443

SSLOptions +StrictRequire

SSLProtocol -all +TLSv1 +SSLv3
SSLCipherSuite ALL:!ADH:!EXPORT56:RC4+RSA:+HIGH:+MEDIUM:+LOW:+SSLv2:+EXP:
+eNULL

 SSLRandomSeed startup builtin
SSLRandomSeed startup file:/dev/urandom 512
SSLRandomSeed connect builtin
SSLRandomSeed connect file:/dev/urandom 512

AddType application/x-x509-ca-cert .crt
AddType application/x-pkcs7-crl    .crl

SSLPassPhraseDialog  builtin

SSLSessionCache        shmcb:/var/run/apache2/ssl_scache(512000)
SSLSessionCacheTimeout  300

SSLMutex  file:/var/run/apache2/ssl_mutex

SSLCertificateFile /usr/local/apache2/conf/ssl.crt/wildcart.signedca.crt
SSLCertificateKeyFile /usr/local/apache2/conf/ssl.key/wildcart.signedca.key

SSLCACertificateFile /usr/local/apache2/conf/ca.crt/ca.crt
SSLCARevonationFile /usr/local/apache2/conf/ssl.crl/crl.pem

SetEnvIf User-Agent ".*MSIE.*" nokeepalive ssl-unclean-shutdown
downgrade-1.0 force-response-1.0

SSLProxyEngine off

</IfModule>
```

SSLCertificateFile és *SSLCertificateKeyFile* a hiteles CA által aláírt kulcsot és certet, *SSLCACertificateFile* direktíva az általunk generált CA cert-jét és a *SSLCARevonationFile* pedig a visszavont certek listáját.

Többi direktívát a vhostoknál szükséges beállítani.
Szimpla SSL vhost:

```
<VirtualHost *:443>
    SSLEngine         On

    ServerName        www.domain.hu
    DocumentRoot      /var/www/www/
    CustomLog         /var/log/access.www.log "%t %h %{HTTPS}x %
{SSL_PROTOCOL}x %{SSL_CIPHER}x %{SSL_CIPHER_USEKEYSIZE}x %
{SSL_CLIENT_VERIFY}x  \"%r\" %b"
```

```
    Errorlog          /var/log/error.www.log
    <Directory /var/www/www/>
        SSLRequireSSL
    </Directory>
</VirtualHost>
```

Kliens certet igénylő vhost:

```
<VirtualHost *:443>
    SSLEngine         On

    ServerName        client.domain.hu
    DocumentRoot      /var/www/client/
    CustomLog         /var/log/access.client.log "%t %h %{HTTPS}x %
{SSL_PROTOCOL}x %{SSL_CIPHER}x %{SSL_CIPHER_USEKEYSIZE}x %
{SSL_CLIENT_VERIFY}x  \"%r\" %b"

    Errorlog          /var/log/error.clientlog
    <Directory /var/www/client/>
        SSLRequireSSL
        SSLRequire    %{SSL_CLIENT_S_DN_O}  eq "Owner" and  %
{SSL_CLIENT_S_DN_CN}  eq "client.domain.hu"

     SSLVerifyClient require
     SSLVerifyDepth  1
    </Directory>
</VirtualHost>
```

SSLRequire-vel tudjuk szabályozni az elérést. Mivel mi szabjuk meg mi legyen a kliens kulcs ezen mezőiben, ezért könnyen tudjuk vele korlátozni a felhasználót is.

7. Fellélegzés

10perces munkával ugyanezt el lehet érni ha tudjuk mit szeretnénk és csináltunk is hasonlót. Sajnálatos módon az apache dokumentációja elégséges de nem több, sok érdekes dolgot tud produkálni használat közben. Ha például a *SSLCACertificateFile* direktívát vhostnál adjuk meg, viszont nem mindegyiknél vagy legalábbis nem a legelsőnek betöltődőnél akkor elfelejti inicializálni, és sajnos nem tudja a kliens kulcsának eredetét igazolni, ezért érdemes az ssl.conf-ba rögtön definiálni. Ezen felül 2006-os kérésként az Apache Software Foundation bugtraq-jában szerepel mint hibabejelentés, hogy a revocation list-et csak restart-al lehet újratölteni, és legalább a dokumentációba meg kellene ezt említeni, de sajnos ez nem került bele.

Bucsay Balázs <bucsay.balazs[]rycon[]hu>