Apache - SSL

Contents

Hogyan m?ködik

Az SSL-el védett weboldalak esetén két célunk van. Az els?, és legfontosabb, hogy titkosított csatornán kommunikáljon a szerver a klienssel, vagyis, hogy senki, semmilyen módon ne tudja lehallgatni a kommunikációt. A második célunk, hogy a kliens böngész? egyértelm?en meg tudjon róla gy?z?dni, hogy a cél szerverrel kommunikál, és nem egy harmadik féllel.

Titkosított csatorna felépítése

Az SSL kapcsolat felépítésekor a szerver és a kliens felépít egy titkosított csatornát, amibe akkor sem lehet belehallgatni, ha valaki kezdettekt?l fogva lehallgatja a kapcsolatot. A kapcsolat felépítésére aszimmetrikus kulcsú titkosítást használ az SSL. A kapcsolat felépítése közben a kliens és a szerver megállapodnak egy olyan szimmetrikus kulcsban, amit csak ?k ketten tudnak. A tényleges kommunikációt már ezzel a szimmetrikus kulccsal titkosítják. A titkos kapcsolat felépítése nagyon leegyszer?sítve a következ?:

1. A kliens küld egy kérést a szervernek, hogy SSL-el kapcsolódni akar hozzá.
2. A szervernek van egy titkos és egy publikus kulcsa párja. Amit a publikus kulccsal valaki titkosít, azt csak a titkos kulccsal elehet dekódolni. A szerver visszaküldi a kliensnek a publikus kulcsát, valamint a titkosítási paramétereket, megállapodásokat. Természetesen ezt bárki láthatja aki hallgatózik a vonalon, de nem baj, nem tud vele semmit kezdeni.
3. A kliens generál egy szimmetrikus kulcsot, amit titkosít a szerver publikus kulcsával. Ezt visszaküldi a szervernek. A szimmetrikus kulcsot kizárólag a szerver tudja kiolvasni, mivel csak nála van meg az aszimmetrikus titkos kulcs. A szerver kititkosítja az üzenetet, kiolvassa bel?le a szimmetrikus kulcsot.
4. A kliens és a szerver innent?l kezdve a kliens által kitalált szimmetrikus kulcsot fogják használni az üzenetek titkosítására. Vagyis az aszimmetrikus szerver kulcsok csak arra kellenek, hogy a kliens eljuttassa a közös szimmetrikus kulcsot a szervernek, anélkül, hogy bárki err?l tudomást szerezhetni.

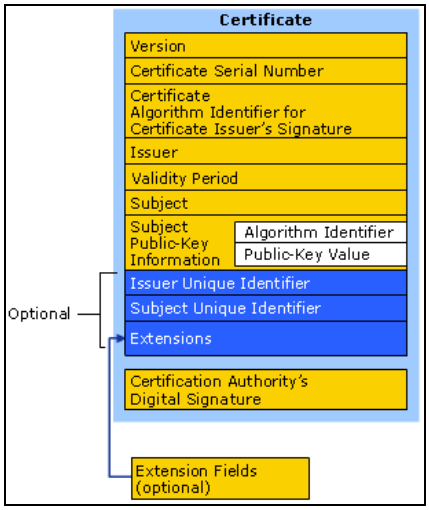
A valóságban a kliens nem a végleges szimmetrikus kulcsot küldi el a szervernek, hanem egy paraméter halmazt, ami alapján a szerver és a kliens is létre tudja hozni ugyan azt a kulcsot. Miután a kliens és a szerver is létrehozták a kulcsot, küldenek egymásnak egy nyugtázó üzenetet az új titkosított kulccsal titkosítva, amivel jelzik egymásnak, hogy a titkos csatorna el?állt.

file:///home/adam/tmp/img1.png

A szerver és kliens azonosítása

Azt kell látni, hogy a kliens vagy a szerver azonosítása csak azon alapszik, hogy a kliensnél, a szervernél és az úgynevezett Hitelesítési szolgáltatónál olyan információ van, ami csak neki lehet meg. Amikor a szerver, a kliens vagy a Hitelesítési szolgáltató ezt ?bebizonyítja?, onnantól megbízunk a másik félben. Klimesként a szerverben, vagy szerverként a kliensben, ha kliens oldali azonosítás is történik.

Szerver azonosítása



Nagyon fontos, hogy a kliens meg tudjon arról gy?z?dni, hogy a szerver által visszaküldött publikus kulcsot tényleg a cél szerver küldte, és nem egy közbeékel?dött támadó (men-in-the-middle). A szerver ezért nem csak a publikus kulcsát küldi el a kliensnek, hanem egy borítékot, amiben benne van a publikus kulcsa is, és a boríték alá van írva. Ezt a borítékot hívják tanúsítványnak. A tanúsítvány tanúsítja a kliens számára, hogy a tanúsítványban lév? kulcs tényleg a szerveré. A tanúsítványt úgynevezett Hitelesítési szolgáltatók (Certificate Authority, CA) bocsájtják ki. A szerver a következ? képen juthat tanúsítványhoz (offline folyamat). Els?ként generál magának egy titkos kulcsot. A titkos kulcsból generál egy úgynevezett tanúsítvány igényl?t. Ebbe belegenerálja magának a titkos kulcshoz tartozó publikus kulcsát, valamit további információkat a szerverr?l (földrajzi hely, vállalat neve..). A legfontosabb információ a tanúsítvány igényl?ben a teljes domain neve a szervernek, amit védeni akar. A tanúsítvány igényl?t a rendszergazda elküldi egy CA-nak, aki az igényl?b?l elkészíti a szerver tanúsítványát, amiben benne van a szerver publikus kulcsa, információk a szerverr?l, a tanúsítványhoz tartozó domain név, a tanúsítvány kibocsájtója, valamint a CA digitális aláírása. A CA által elkészített tanúsítványt és a szerver titkos kulcsát kell beállítani a szervernek. A CA is rendelkezik egy titkos és egy publikus kulccsal, csak pont fordítva használja ?ket mint a szerver. Ugyanis a tanúsítvány igényl?t a titkos kulccsal írja alá. A kliensek pedig a CA publikus kulcsával tudják ellen?rizni a digitális aláírást. A digitális aláírás egyértelm?en bizonyítja, hogy a tanúsítvány tartalma nem volt módosítva, valamit, hogy az aláírást tényleg a CA készítette. A CA digitális aláírásának ellen?rzéséhez a kliens böngész?nek szüksége van a CA publikus kulcsára. A kliens böngész?jébe gyárilag benne vannak a hitelesít? szolgáltatók publikus kulcsai. Azokat a tanúsítványokat fogadja el a böngész? amikhez van publikus kulcsa az aláírás ellen?rzéséhez.

Saját aláírás használata

Ha az SSL-el védett oldal bels? használatra való, akkor felesleges publikus tanúsítványt csináltatni. Készítünk saját használatra egy Hitelesítési szolgáltató titkos és publikus kulcsot. A CA titkos kulccsal aláírjuk a saját tanúsítvány igényl?nket. A Hitelesítési szolgáltatónk publikus kulcsát pedig importálni kell a felhasználók böngész?jébe, mint biztonságos Hitelesítési szolgáltató. Ekkor a böngész? el fogja fogadni a szerver által küldött tanúsítványt, mivel azt hitelesnek fogja gondolni az importált kulcs miatt.

Kliens azonosítása

Különösen védett tartalom esetében szükség lehet rá, hogy a szerver is azonosítani tudja a klienst. Ehhez a kliensnek is szükség van egy saját tanúsítványra. Ekkor a szervernek is be kell állítani egy Hitelesítési szolgáltatót, aki aláírta a kliens tanúsítványát. A szervernek pedig szükség van a Hitelesítési szolgáltató publikus kulcsára, aki aláírta a kliens tanúsítványát, amivel ellen?rizni tudja a tanúsítványt kapcsolódáskor. Ehhez minden képen szükséges hogy létrehozzunk egy saját Hitelesítési szolgáltatót. Ez lehet ugyan az, amit a saját szerver tanúsítványok legyártására használunk. Készítünk ebben az esetben is egy titkos kulcsot a kliensnek, majd abból egy tanúsítvány igényl?t, amit aláíratunk a saját Hitelesítési szolgáltatónkkal, amib?l el?áll a kliens tanúsítványa. A hitelesítési szolgáltató publikus kulcsát beállítjuk a szerverbe, hogy azzal ellen?rizze a hozzá beérkez? kliens tanúsítványokat. A kliens titkos kulcsát és a tanúsítványt pedig össze kell csomagolni egy pkcs12 formátumba, hogy importálni lehessen a böngész?be a saját kulcsok közé. A böngész? a szerver kérésére el fogja küldeni a beállított kliens tanúsítványt. A kliens és a szerver ekkor visszafelé is le tudja zongorázni a titkosított csatorna felépítését. Ekkor a szerver is visszaküld egy a kliens publikus kulcsával titkosított információt. Ha nem saját hitelesítési szolgáltatóval készítenénk el a kliens tanúsítványát, hanem egy mindenki számára elérhet? szolgáltatóval, akkor bárki készíttethetne magának olyan tanúsítványt, amit a szerver elfogadna.

Hogyan támadnak

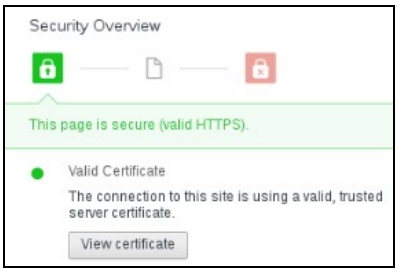
Ahogy azt már láttuk, a titkosított csatornát feltörni semmilyenek módon nem lehet. A támadó viszont ha beékel?dik a kliens és a szerver közé, akkor felépíthet egy SSL kapcsolatot a szerverrel, eljátszva hogy ? a kliens, és egy SSL kapcsolatot a klienssel, mintha ? lenne a szerver. Ilyen esetben a böngész? figyelmeztetni fogja a felhasználót, hogy a tanúsítvány nem megfelel?, mivel valójában nem a szerverrel beszélget a kliens. A másik lehet?ség, hogy a beékel?dött felhasználó kiépíti az SSL kapcsolatot a szerverrel, de a felhasználó számára HTTP-n továbbítja a tartalmat. Ez csak akkor lehetséges, ha a felhasználó nem közvetlenül egy https-es linket ír be a böngész?be, vagy nyit meg könyvjelz?b?l, hanem http oldalon kattint egy linkre, vagy a szerver http-r?l https-re irányít át. Ha a felhasználó nem veszi észre, hogy HTTP-n kommunikál a böngész?, akkor kész a baj. Tehát a felhasználókat meg kell tanítani, hogy mindig ellen?rizzek, hogy https fölött vannak e, és ha a böngész? ssl hibát jelez, akkor nem szabad tovább menni.

Kulcsok legyártása

Warning
Most már a böngész?k nem fogadják el az **SHA-1**-es tanúsítványokat, ezért nagyon fontos, hogy az összes kulcs és aláírás igényl? SHA-2-öt használjon. Ehhez az összes generálásba el kell helyezni a **-sha256** kapcsolót.

SHA-1-es aláírás (Chrome 53)

SHA-2-es aláírás:



Saját Hitelesítési szolgáltató elkészítése

Els?ként létre fogunk hozni egy Hitelesítés szolgáltatót (CA). Ehhez le kell generálnunk a CA publikus és titkos kulcsát. A tanúsítványokat a titkos kulccsal fogjuk el?állítani, míg a publikus kulcsot el kell juttatni a felhasználóink böngész?ibe, mint megbízható hitelesítési szolgáltató, hogy a böngész? elfogadja a szerver tanúsítványát. Ezen felül a szerverbe szintnét be kell állítani a CA publikus kulcsát mint hitelesítési szolgáltató, hogy elfogadja a kliens tanúsítványát.

A titkos kulcs neve berki-ca.key, míg a publikus kulcs neve berki-ca.pem. Ezen felül adjuk meg, hogy a kulcs 3650 napig (10 évig) legyen érvényes. Generálás közben válaszolunk kell pár kérdésre. Ezek közül semelyiknek sincs nagy jelent?sége. Kiemeltem amiket meg kell adni. Az itt megadott értékek látszódni fognak a böngész?ben importált CA adataiként.

```
# openssl req -new -x509 -sha256 -days 3650 -nodes -out berki-ca.pem -keyout berki-ca.key

Generating a 2048 bit RSA private key
.....................................................................+++
...........+++
writing new private key to 'berki-ca.key'
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
```

```
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [XX]:HU
State or Province Name (full name) []:
Locality Name (eg, city) [Default City]:Budapest
Organization Name (eg, company) [Default Company Ltd]:Berki Corpoation
Organizational Unit Name (eg, section) []: Berki-CA Division
Common Name (eg, your name or your server's hostname) []: ca.berki2.org
Email Address []:info@berki2.org
Ezzel létrejött a Hitelesítési Szolgáltató titkos és publikus kulcsa:
berki-ca.key
berki-ca.pem
```

Note
Láthatjuk, hogy hitelesítési szoláltató is SHA-2 -t használ.

Szerver kulcsok el?állítása

Minden egyes domain-hez készíteni kell egy aszimmetrikus kulcs párt. Nem elég csak a f? domain-re létrehozni egy kulcsot, hogy azt használja az összes al domain, ugyanis a tanúsítványnak tartalmaznia kell a teljes domain nevet. A CA többek közözött azt bizonyítja az aláírásával, hogy a tanúsítvány valóban azt a domain-t védi, akivel a kliens beszélget. Els? lépésként készíteni kell egy titkos kulcsot a teljes domain névre. Publikus kulcsot nem kell külön készíteni, a tanúsítvány igényl? generálásával el?áll a publikus kulcs is. A tanúsítvány igényl?be belekerül a szerver publikus kulcsa, valamint a szerver teljes domain neve. Ezt kell majd aláíratni a CA-val, hogy el?álljon bel?le a szerver tanúsítványa, amit majd el tud küldeni a kliensnek a szerver, ha SSL kapcsolatot akar vele kiépíteni. A titkos kulcsot sosem küldjük el a CA-nak. A CA csak a publikus kulcsot is tartalmazó tanúsítványt igényl?t kapja meg. Hozzunk létre egy mappát a root alatt, és ott hozzuk létre az összes kulcsot, majd a végén a helyükre másolhatjuk:

```
# mkdir /root/SSL
# cd /root/SSL
```

A kulcsokat az **openssl** eszközzel fogjuk generálni. Nem ez az egyetlen módja a kulcsok generálásának, nagyon sok egyéb megoldás is létezik.

Titkos kulcs készítése

Els?ként el?állítjuk a szerver 2048 bites titkos RSA kulcsát. Erre nagyon kell vigyázni, ez nem kerülhet rossz kezekbe. A kulcsot nevezzük el a domain név alapján, amihez a kulcs tartozni fog:

```
# openssl genrsa -out admin.berki2.org-server.key 2048

Generating RSA private key, 2048 bit long modulus
.................................+++
.............................+++
e is 65537 (0x10001)
```

Ezzel el?állt a szerver titkos kulcs: admin.berki2.org-server.key

A kulcsot nem titkosítottuk, és nem is adtunk hozzá passphrase-t. Ha belenézünk a fájlba,láthatjuk, hogy nincs titkosítva:

```
-----BEGIN RSA PRIVATE KEY-----
MIIEpAIBAAKCAQEAyZ4ZNz7TbN5p088mRv/JiogOmmS9m8R4gDD7112ayqVCGUzP
.....
QG/XDR6Q/yNoeMbVI6rt5DPYvlyg1KfnnvGHuvnhkbA7Nfc1M+x73A==
-----END RSA PRIVATE KEY-----
```

Lehetséges lenne titkosítani a titkos kulcs fájl tartalmát, de akkor minden olyan alkalmazásba, ami használja majd a titkos kulcsot, induláskor be kéne kézzel írni a jelszót, ami a szervernél nem megoldható, ezért plain text-ként fogjuk tárolni a titkos kulcsot is.

Tanúsítvány igényl? készítése

Készítünk egy tanúsítvány igényl?t a titkos kulcs alapján. A tanúsítvány igényl? generálása automatikusan elhelyezi a publikus kulcsot is a tanúsítvány igényl?be, külön tehát nem kell publikus kulcsot generálni. A tanúsítvány igényl? generálás közben meg kell adni a leend? tanúsítványunk alap adatait. Ezek közül a legfontosabb a Common Name-ben megadott érték, ami a szerver teljes domain neve kell legyen, amihez a kulcs tartozni fog. Amennyiben ez eltér a tényleges domain névt?l, a böngész? akkor sem fogja elfogadni a tanúsítványt, ha ismeri a tanúsítványt aláíró CA-t. A ? A challenge password? mez?nek semmi jelent?sége. Egyes CA-k megkövetelhetik, hogy egy jelszót is küldünk át a tanúsítvány igényl?ben, mely a jöv?beli azonosításunkat tenné lehet?vé. Jelen esetben semmi jelent?sége nem lesz, a tanúsítvány el?állításában nem játszik szerepet.

```
# openssl req -new -sha256 -key admin.berki2.org-server.key -out admin.berki2.org-server.sha2.req

You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [XX]:HU
State or Province Name (full name) []:
Locality Name (eg, city) [Default City]:Budapest
Organization Name (eg, company) [Default Company Ltd]:Berki Ugyvedi Iroda
Organizational Unit Name (eg, section) []: BUI
Common Name (eg, your name or your server's hostname) []:admin.berki2.org
Email Address []:info@berki2.org

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
```

Ezzel el?állt a domain-hez tartozó tanúsítvány igényl?, amit alá kell iratni a CA-val:

```
admin.berki2.org-server.req
```

Tip
Itt is fontos a **-sha256** kapcsoló használata, hogy sha-2 tanúsítványt kapjuk majd.

Tanúsítvány igényl? aláírása

A tanúsítvány igényl?ben benne van a szerver publikus kulcsa, és a teljes domain neve. A CA tanúsítja, hogy a domain név valóban a mi tulajdonunkban van, és hogy a domain név összetartozik a publikus kulccsal. Ehhez a CA a tanúsítvány igényl?ben megkapott tanúsítvány adatokat, a domain publikus kulcsát, valamint CA publikus kulcsát beleteszi egy borítékba, amit aláír a titkos kulcsával. Így áll el? a domain tanúsítványa. Nagyon fontos, hogy adjunk egy egyedi sorszámot a tanúsítványnak a set_serial paraméterrel. Nem lehet két azonos sorszámú tanúsítványa egy CA-nak. Ha több tanúsítványunknak is ugyan azt a sorszámot adjuk, akkor a böngész? egyiket sem fogja elfogadni.

Note
A tanúsítványokhoz több kiegészítés (exctension) is létezik, melyek extra információkat engednek hozzáadni a tanúsítványhoz. A **Google Chrome** az 58-as verziótól kezdve csak olyan tanúsítványt fogad el amiben a **"X509v3 Subject Alternative Name"** kiegészítés tartalmazza a szerver domain nevét. Ha a tanúsítvány igényl?t valódi szolgáltató írja alá, akkor ? ezt bele fogja tenni. Ha magunknak írjuk alá, akkor fontos, hogy ezt megadjuk.

A tanúsítvány legyártásház meg kell adni a CA titkos és publikus kulcsát is, valamint a szerver tanúsítvány igényl?jét. Az -extfile kapcsolóval adhatjuk meg a X509V3 kiegészítéseket tartalmazó fájl nevét. Itt fogjuk megadni a SubjectAlternativeName értékét. Itt vagy IP címet, vagy a DN: el?tag után a domain nevet kell megadni.

```
# openssl x509 -sha256 -req -extfile <(printf "subjectAltName=DNS:admin.berki2.org") -in admin.berki2.org-server.sha2.req -CA berki-ca.pem -C

Signature ok
subject=/C=HU/L=Budapest/O=Berki Ugyvedi Iroda/CN=admin.berki2.org/emailAddress=info@berki2.org
Getting CA Private Key
```

Ezzel el?áll a szerver tanúsítványa:

```
admin.berki2.org-server.cer
```

Tip
Ha itt nem használtuk volna a **-sha256** kapcsolót, akkor az SHA-2-es igényl? ellenére egy SHA-1-es tanúsítványt kaptunk volna.

Listázzuk ki a tanúsítvány attribútumait. Listázni a -text kapcsolóval lehet:

```
# openssl x509 -in admin.berki2.org-server.sha2.cer -text
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number: 104 (0x68)
    Signature Algorithm: sha256WithRSAEncryption
        Issuer: C=HU, ST=Budapest, L=Budapest, O=Berki Ugyvedi Iroda, OU=BUI, CN=admin.berki2.org/emailAddress=info@berki2.org
        Validity
            Not Before: Dec 13 13:06:54 2017 GMT
            Not After : Dec 11 13:06:54 2027 GMT
        Subject: C=HU, ST=Budapest, L=Budapest, O=Berki Ugyvedi Iroda, OU=BUI, CN=admin.berki2.org/emailAddress=info@berki2.org
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                Public-Key: (2048 bit)
                Modulus:
                    ...
                Exponent: 65537 (0x10001)
        X509v3 extensions:
            X509v3 Subject Alternative Name:
                DNS:admin.berki2.org
    Signature Algorithm: sha256WithRSAEncryption
         ...
```

Két kritikus mez? van:

1. CN=admin.berki2.org/.. *(ezt nézi a Firefox és a Chrome els?dlegesen)*
2. X509v3 Subject Alternative Name: DNS:admin.berki2.org *(ezt is nézi a Chrome az 58-as verziótól kezdve)*

Kinél milyen kulcs van

Kulcs neve	Hol van	Funkció
admin.berki2.org-server.key	szerver	A szerver ezzel titkosítja ki a kliens által küldött szimmetrikus kulcsot, amit a session alatt használni fognak.
admin.berki2.org-server.cer	szerver	A szerver tanúsítványa, mai tartalmazza a szerver publikus kulcsát, amivel a kliens titkosítja az általa kitalált szimmetrikus kulcsot, valamint a CA aláírását, ami alapján meggy?z?dhet a kliens, hogy a publikus kulcs tényleg a szerveré.
berki-ca.pem	Kliens böngész?jében/szerverben	A Hitelesítés szolgáltató publikus kulcsa. Ezt a kliens böngész?jébe kell elhelyezni, a megbízható Hitelesítés szolgáltatók listájába. Ezzel a kulccsal tud meggy?z?dni a böngész? arról a szerver által küldött tanúsítvány hiteles.

Kliens kulcsok

A kliens számára is készíthetünk tanúsítványt. Az apache-ba beállíthatjuk, hogy csak olyan klienssel építhet ki kapcsolatot, aki fel tud mutatni egy olyan tanúsítványt, amit az a hitelesítési szolgáltató írt alá, ami be van állítva a szerveren. A kliens tanúsítványát kiállító CA természetesen lehet ugyan az,

amivel a szerver kulcsait aláírtuk. Els? lépésként a kliens számára is generálni kell egy titkos kulcsot, majd abból egy tanúsítvány igényl?t, amivel a kliens publikus kulcsa is el?áll. A kliens tanúsítvány igényl?jében nem számít, hogy milyen domain nevet adunk meg a Common name mez?ben. A CA által el?állított kliens tanúsítványt és a kliens titkos kulcsát be kell rakjuk a kliens böngész?jébe a saját tanúsítványok köz. Ehhez a CA által el?állított tanúsítványt és a kliens titkos kulcsát össze kell majd csomagoljuk pkcs12 formátumba, mert a böngész?k csak így fogadják el. A szerver kulcsával ellentétben nem kell domain-enként egy kliens kulcsot generálni. A szerver nem fogja ellen?rizni, hogy a kliens olyan domain-hez akar e csatlakozni, ami a Common name mez?ben meg van adva.

Kliens kulcsok legyártása

A kliens titkos kulcsa

Els? lépésként készítsük el a kliens 2048 bites titkos kulcsát. (Az RSA az alapértelmezett)

```
# openssl genrsa -out berki-client.key 2048
enerating RSA private key, 2048 bit long modulus
..........+++
.............+++
e is 65537 (0x10001)
```

Ezzel el?áll a kliens titkos kulcsa: berki-client.key

Itt sem használtunk passphrase-t, akárcsak a szerver titkos kulcsa, ez is plain text. Azonban a kliens titkos kulcsát nem fogjuk odaadni a klienseknek, hanem a tanúsítvánnyal együtt össze fogjuk csomagolni egy pkcs12 fájlba, amit viszont már jelszóval fogunk védeni.

Kliens tanúsítvány igényl?je

A titkos kulcs segítségével generáljuk le tanúsítvány igényl?t. A tanúsítványban szerepl? adatoknak a kliens esetében nincs semmi jelent?sége, pusztán a tanúsítvány beazonosítását segítik. Akárcsak a szerver tanúsítvány igényl?jének az esetében, itt sincs funkciója a ?A challenge password? mez?nek. Értéke irreleváns.

```
# openssl req -new -key berki-client.key -out berki-client.req

You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [XX]:HU
State or Province Name (full name) []:
Locality Name (eg, city) [Default City]:Budapest
Organization Name (eg, company) [Default Company Ltd]:Berki Ugyvedi Iroda
Organizational Unit Name (eg, section) []: BUI
Common Name (eg, your name or your server's hostname) []:berki2.org
Email Address []:info@berki.org

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
```

Ezzel el?állt a kliens tanúsítvány igényl?je: berki-client.req

Tanúsítvány el?állítása

Most a CA-val aláíratjuk a kliens tanúsítvány igényl?jét, amib?l el?áll a kliens tanúsítványa. Nagyon fontos, hogy egyedi sorszámot adjunk a tanúsítványnak. Mivel a szerver által küldött tanúsítvány, és a kliens tanúsítványa egyszerre lesz jelen a böngész?ben, és a CA közös, ezért a böngész? nem fogja elfogadni egyik tanúsítványt sem, ha azonos a sorszámuk.

```
#  openssl x509 -req -in berki-client.req -CA berki-ca.pem -CAkey berki-ca.key -set_serial 101 -days 3650 -outform PEM -out berki-client.cer

Signature ok
subject=/C=HU/L=Budapest/O=Berki Ugyvedi Iroda/CN=berki2.org/emailAddress=info@berki.org
Getting CA Private Key
Ezzel el?állt a kliens tanúsítványa: berki-client.cer

Utolsó lépésként a kliens titkos kulcsát és tanúsítványát összecsomagoljuk pkcs12 formátumba, amit majd importálni tudunk a böngész?be. Itt m
# openssl pkcs12 -export -inkey berki-client.key -in berki-client.cer -out berki-client.p12
Enter Export Password: *******
Verifying - Enter Export Password: *******
```

Ezzel el?állt a kliens böngész?be importálható tanúsítványa: berki-client.p12

Kinél milyen kulcs van

Kulcs neve Hol van Funkció berki-client.p12 Kliens böngész?jében A szerver böngész?jébe importált saját tanúsítvány, amiben össze van csomagolva a kliens titkos kulcsa, és a kliens CA által kibocsájtott tanúsítványa. A kapcsolat felépítésekor ezt a tanúsítványt küldi el a szervernek. berki-ca.pem Szerver A Hitelesítés szolgáltató publikus kulcsa. Ezt a szervernek kell megadni, mint megbízható Hitelesítés szolgáltató. Ezzel a kulccsal tud meggy?z?dni a szerver arról a kliens által küldött tanúsítvány hiteles, vagyis, hogy a klines az akinek mondaj magát.

Fontos, hogy a szerver órája jól járjon. Ha késik, akkor lehet hogy nem fogja elfogadni a kliens tanúsítványát, mondván, hogy még nem valid:

```
[client 192.168.10.104] Certificate Verification: Error (9): certificate is not yet valid
```

Importálás a böngész?be

Ahhoz hogy a böngész?nk elfogadja az összes tanúsítványt, amit a saját CA-nkal aláírtunk, a saját CA-nk publikus kulcsát (berki-ca.pem) importálni kell a böngész?be a megbízható Hitelesítés szolgáltatók közé. Firefox:

- Preferences -> Advanced -> Certificates fül -> View Certificates gomb -> Authorities fül -> Import... gomb.

Ha a saját CA-nkat felvettük a megbízható szolgáltatók közé, akkor a böngész?nk az összes tanúsítványunkat el fogja fogadni, a kis lakat zöldre fog változni.

Ahhoz, hogy a szerver felé a böngész? autentikálni tudja magát, a saját titkos kulcsunkból és a saját tanúsítványunkból összecsomagolt pkcs12 fájlt importálni kell szintén a böngész?be a saját tanúsítványaink közé (berki-client.p12). Firefox:

- Preferences -> Advanced -> Certificates fül -> View Certificates gomb -> Your Certificates fül -> Import...

Apache beállítások

Fel kell telepíteni az ssl modult az apache-hoz valamit az openssl programot, ha eddig ezt nem tettük volna meg.

```
# dnf install mod_ssl openssl
```

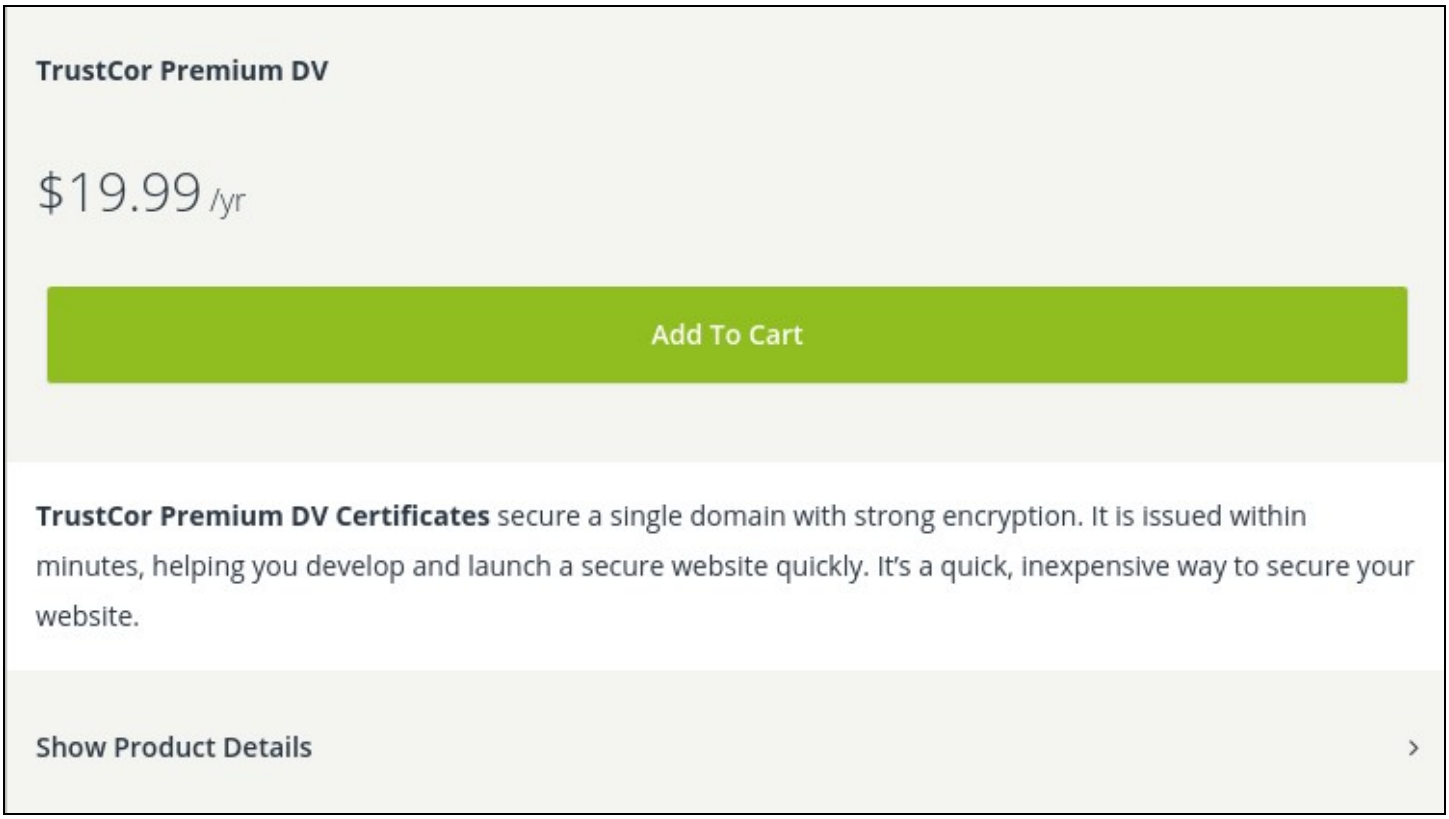
Virtual ssl host elkészítése

```
<VirtualHost *:443>
    ....

    ##########################
    # SSL beállítások
    ##########################

    # SSL parameterek
    ##########################
    SSLEngine on

    # Here, I am allowing only "high" and "medium" security key lengths.
    SSLCipherSuite HIGH:MEDIUM

    # Here I am allowing SSLv3 and TLSv1, I am NOT allowing the old SSLv2.
    SSLProtocol all -SSLv2


    # Kliens autentikaco
    ##########################
    SSLVerifyClient require
    SSLVerifyDepth 1

    # Certificate Authority (CA):
    SSLCACertificateFile /etc/httpd/conf/ssl/berki-ca.pem


    # Server Certificate Chain:
    SSLCertificateChainFile /etc/httpd/conf/ssl/berki-ca.pem


    # Szerver kulcsok
    ##########################

     # Server Certificate:
    SSLCertificateFile /etc/httpd/conf/ssl/admin.berki2.org-server.cer

    # Server Private Key:
    SSLCertificateKeyFile /etc/httpd/conf/ssl/admin-berki2.org-server.key

    ....
</VirtualHost>
```

Ha valódi SSL szolgáltatót használunk, akkor gyakran meg kell adni egy tényleges ...

NO-ip cert generálás

Még miel?tt bármit is generálnánk, venni kell egy 1 éves cert-et a NOip oldalán. Amt még bármiféle generálás el?tt ki kell fizetni.



Most generálni kell egy **Certificate Signing Request** fájlt, amit majd fel kell tölteni a No-ip-re. Ehhez kel a szerver titkos kulcsa, semmi más:

```
$ openssl req -new -sha256 -key wiki.berki.org-server.key -out wiki.berki.org-server-20210829.req

You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [XX]:HU
State or Province Name (full name) []:Pest
Locality Name (eg, city) [Default City]:Budapest
Organization Name (eg, company) [Default Company Ltd]:Berki corp
Organizational Unit Name (eg, section) []:
Common Name (eg, your name or your server's hostname) []:wiki.berki.org
Email Address []:info@berki.org

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
```

A leges legfontosabb a **Common Name** mez?, itt kell megadni a szerver teljes domain nevét.

Most el?állt a **wiki.berki.org-server-20210829.req** fájl. Menjünk fel a no-ip felületére az **SSL Certificates** oldalra, ahol már vár ránk a félig megrendelt cert. https://www.noip.com/support/knowledgebase/get-trustcor-premium-dv-ssl/ Itt kattintsunk az ADD csr gombra

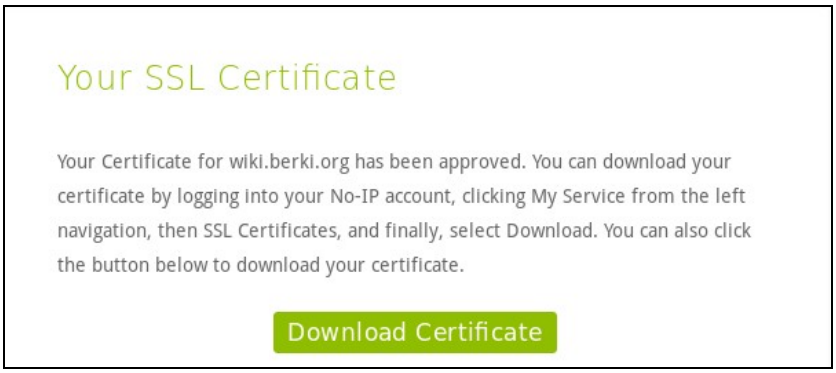
A megnyíló popup-ban másoljuk be a request teljes tartalmát, és a szerver típusa legyen: Apaceh mod ssl.

Ezen a ponton még nem fogja legenerálni a cert-t, most bizonyítani kell hogy tényleg miénk a domain. Ekkor létre fog hozni egy DNS txt rekordot, amit el kell helyezni a DNS beállításokban. De ha NO-ip-s domain nevünk van (ami jelen esetben igaz) akkor ezt ? kb 5 perc alatt elhelyezi a DNS-ben, nekünk ezzel külön nincs dolgunk.

Az SSL beállításokban a NO-ip oldalán a gomb Verify-ra változik:

Ezt kell nyomkodni, addig amíg ki nem írja, hogy a dns verifikáció megtörtént, ekkor fogják csak aláírni a cert-et, amit majd emailben kb 10 perc után küldenek.

Az email-ben lesz egy letölt? gomb:



Ami visszavisz a NO-ip ssl oldalára: