

# Site-to-site VPN beállítása (gyakorlat)

## Előkészületek

### NTP beállítása

Az IKE és az IPSEC protokoll nagyon érzékeny a rendszeridő beállításaira. Mielőtt hozzákezdenél a VPN beállításához, győződj meg róla, hogy a router-ek NTP kliense jól van beállítva, és pontos rajtuk a rendszeridő. A saját országodnak megfelelő nyilvános NTP szervereket [például itt is megtalálhatod](#).

```
/system clock set time-zone-autodetect=yes time-zone-name=Europe/Budapest  
/system ntp client set server-dns-name=0.hu.pool.ntp.org,1.hu.pool.ntp.org primary-ntp=[/resolve 2.hu.pool.ntp.org] secondary-ntp=[/resolve 3.hu.pool.ntp.org]
```

### FQDN (teljes host nevek) beállítása

A továbbiakban sok kényelmetlenségtől megkímélheted magadat akkor, ha rendes host neveket hozol létre a `peer`-ekhez. Ennek különösen akkor lesz jelentősége, ha később nem csak site-to-site, hanem road warrior típusú VPN klienseket is támogatni szeretnél. Ebben a példában az office-hoz az office.myserver.hu, a branch01-hez a branch01.myserver.hu host neveket fogjuk használni.

Ezen felül elvégezzünk pár beállítást amivel elérjük, hogy ez a két név a megfelelő IP címekre oldódjon föl.

Az "ISP" internet router-én adjuk meg az IP címeket a host nevekhez, statikus DNS bejegyzések formájában:

```
/ip dns static  
add address=100.3.3.10 name=office.myserver.hu  
add address=100.2.2.10 name=branch01.myserver.hu  
# Ez már be volt állítva korábban, csak a teljesség kedvéért  
/ip dns set allow-remote-requests=yes
```

Az office router-en beállítjuk az "ISP" által adott DNS szerveret és a router teljes nevét.

```
/ip dns set servers=100.3.3.3
/system identity set name=office.myserver.hu
```

Ugyan ez a branch01 router-en:

```
/ip dns set servers=100.2.2.2
/system identity set name=branch01.myserver.hu
```

Mindkettőn teszteld le, hogy működik-e:

```
[admin@branch01.myserver.hu] > /ping office.myserver.hu count=1
SEQ HOST                                SIZE TTL TIME STATUS
0 100.3.3.10                            56 63 0ms
sent=1 received=1 packet-loss=0% min-rtt=0ms avg-rtt=0ms max-rtt=0ms

[admin@branch01.myserver.hu] > /ping branch01.myserver.hu count=1
SEQ HOST                                SIZE TTL TIME STATUS
0 100.2.2.10                            56 64 0ms
sent=1 received=1 packet-loss=0% min-rtt=0ms avg-rtt=0ms max-rtt=0ms
```

## Tanúsítványok generálása

Ahogy korábban jeleztem, tanúsítványokat fogunk használni az azonosításra. Arra készülünk, hogy a fő irodához (office) tovább telephelyeket fogunk csatlakoztatni. Ezért az azonosításhoz szükséges tanúsítványokat és kulcsokat az office router-jén fogjuk előállítani.

Ennél sokkal egyszerűbb megoldás a szimmetrikus kulcsok (pre shared key) használata. Ha egyszerűsíteni szeretnéd a dolgokat, akkor ugorj a következő részre.

Három tanúsítványt fogunk előállítani:

- [Certificate Authority](#)
- Központi iroda (office)
- branch01

Főbb különbségek:

- A CA lejárat ideje nagyon hosszú. A key-usage részben jó sok dolog szerepel.
- A szervernél a key-usage részben tls-server szerepel.
- A kliensnél a key-usage részben tls-client szerepel.

Ha érdekel hogy ezek pontosan mit jelentenek, akkor az [X.509 szabványt](#) kell tanulmányoznod. Ennek bemutatása bőven túlmutat egy ilyen egyszerű iromány keretein.

# CA tanúsítvány generálása

```
/certificate add name=ca.myserver.hu \  
[c]ountry=HU state=Heves locality=Eger \  
    organization=myserver.hu \  
    common-name=ca.myserver.hu \  
    subject-alt-name=DNS: ca.myserver.hu \  
    key-size=4096 days-valid=3650 trusted=yes \  
    key-usage=digital-signature, key-encipherment, data-encipherment, key-cert-sign, crl-sign
```

Nagyon fontos, hogy a name, common-name és subject-alt-name ugyan azt a nevet tartalmazza, és ez az egyértelműség kedvéért valami "ca." kezdetű legyen. Ennek nem feltétlenül kell létező fqdn-nek lenni, mert a CA cert-et megbízható CA cert-ként fogjuk importálni mindenhová.

Hozzárendelünk egy privát kulcsot, és aláírjuk saját magával. Ez elég sokáig eltarthat, számításigényes.

```
/certificate sign ca.myserver.hu
```

Végül exportáljuk egy állományba. Erre azért van szükség, hogy a másik peer-en importálni tudjuk.

```
/certificate export-certificate ca.myserver.hu
```

## office peer kulcs és tanúsítvány generálása

```
/certificate add \  
[n]ame=office.myserver.hu \  
    country=HU state=Heves locality=Eger \  
    organization=myserver.hu \  
    common-name=office.myserver.hu \  
    subject-alt-name=DNS: office.myserver.hu \  
    key-size=4096 days-valid=1095 \  
    trusted=yes key-usage=tlc-server
```

Nagyon fontos, hogy a name, common-name és subject-alt-name részben azonos fqdn-ek legyenek. És persze nem árt, ha ez egy valós létező fqdn.

Ezt is a CA-val írjuk alá. Eltarthat egy ideig...

```
/certificate sign office.myserver.hu ca=ca.myserver.hu
```

Ezt a tanúsítványt nem exportáljuk, mert már azon a router-en van, ahol szükség lesz rá.

## branch01 peer kulcs és tanúsítvány generálása

```
/certificate add \  
[name=branch01.myserver.hu \  
    country=HU state=Heves locality=Eger \  
    organization=myserver.hu \  
    common-name=branch01.myserver.hu \  
    subject-alt-name=DNS: branch01.myserver.hu \  
    key-size=4096 days-valid=1095 trusted=yes key-usage=tlc-client
```

Ezt is a CA-val írjuk alá:

```
/certificate sign branch01.myserver.hu ca=ca.myserver.hu
```

A branch01-nél szükséges exportálnunk a tanúsítványt a titkos kulccsal együtt. Erre majd a branch01 router-en lesz szükség.

```
/certificate export-certificate branch01.myserver.hu type=pkcs12 export-passphrase=abcd1234
```

## Export állományok másolása branch01-re és importálása

A két exportált állományt át kell másolnod a branch01 router-re. A másolást én a példában úgy oldottam meg, hogy WinBox -szal csatlakoztam az office router-hez, és a files menüpont alól letöltöttem őket a saját gépemre. Ezután csatlakoztam a branch01 router-hez is, és ott feltöltöttem őket (szintén WinBox-szal).

Ezután a branch01 router-en így importáltam őket:

```
/certificate import file-name=cert_export_ca.myserver.hu.crt  
/certificate import file-name=cert_export_branch01.myserver.hu.p12
```

Ezután még át kellett írni a neveiket, hogy ne "cert\_export" nevűek legyenek. Ennek nincs túl nagy jelentősége. Amikor majd az azonosításhoz meg kell adnod őket, akkor "cert\_export\_branch01.myserver.hu.p12\_0" helyett egyszerűen "branch01.myserver.hu" néven lehet rá hivatkozni. Valahogy így:

```
[admin@branch01.myserver.hu] /certificate> print detail  
Flags: K - private-key, L - crl, C - smart-card-key, A - authority, I - issued, R -  
revoked, E - expired,  
T - trusted  
0    A T name="cert_export_ca.myserver.hu.crt_0"
```

```
issuer=C=HU, ST=Heves, L=Eger, O=myserver.hu, CN=myserver.hu digest-algorithm=sha256 key-type=rsa
country="HU" state="Heves" locality="Eger" organization="myserver.hu" common-
name="myserver.hu" key-size=4096 subject-alt-name=DNS: ca.myserver.hu days-valid=3650
trusted=yes key-usage=digital-signature, key-encipherment, data-encipherment, key-cert-sign, crl-
sign serial-number="4723FDDFBA4C0EE7"
fingerprint="d478806e903a39ab9c247fee055913d7b359bf599519edc59ca32ebf10764b3d" invalid-
before=dec/29/2020 20:12:56 invalid-after=dec/27/2030 20:12:56 expires-after=521w2d9h51m36s
```

```
1 K      T name="cert_export_branch01.myserver.hu.p12_0"
issuer=C=HU, ST=Heves, L=Eger, O=myserver.hu, CN=myserver.hu digest-algorithm=sha256 key-type=rsa
country="HU" state="Heves" locality="Eger" organization="myserver.hu" common-
name="branch01.myserver.hu" key-size=4096 subject-alt-name=DNS: branch01.myserver.hu days-
valid=1095 trusted=yes key-usage=tls-client serial-number="6E4BF36124495358"
fingerprint="623f9943980a8bd320d1b8c250d6061fb8fee8de0c6d924144e5df6f4e1398b4" invalid-
before=dec/29/2020 20:18:21 invalid-after=dec/29/2023 20:18:21 expires-after=156w2d9h57m1s
```

```
[admin@branch01.myserver.hu] /certificate> set 0 name="ca.myserver.hu"
```

```
[admin@branch01.myserver.hu] /certificate> set 1 name="branch01.myserver.hu"
```

Végül töröltem őket a branch01 router-ről és az office routerről is:

```
/file remove [find where name=cert_export_ca.myserver.hu.crt]
/file remove [find where name=cert_export_branch01.myserver.hu.p12]
```

Ezen felül a saját gépemről is.

**Figyelem!** A fentieket csak a példa kedvéért csináltam ennyire egyszerű módon. A valóságban sokkal erősebb jelszót kellene használnod, és biztonságos módon megoldani az export állományok átmásolását. Sőt ha nagyon szigorúak akarunk lenni, akkor a branch01 privát kulcsát a branch01 router-en kellene előállítanod, és onnan soha nem szabadna kikerülnie. A példa kedvéért (és általános felhasználásra) a fent leírt módszer, a megfelelő elővigyázatosság mellett megfelelő lehet.

## Phase 1 proposal beállítás (ipsec profile)

Ezt mindkét router-en meg kell tenni. A mi konkrét példánkban előre meghatároztuk azokat a konkrét algoritmusokat, amiket használni fogunk. Ezért itt nem fogunk algoritmus listákat megadni, csak konkrét algoritmusokat. Ezt azért tehetjük meg, mert előre ismerjük az összes peer minden képességét, és tudjuk hogy mit akarunk.

Az IKEv2 első fázisához tartozó algoritmusokat [egy külön menüpontban](#), a `/ip ipsec profile` alatt kell fölvennünk. Ezt mindkét router-en megteesszük:

```
/ip ipsec profile
add dh-group=modp2048 enc-algorithm=aes-256 hash-algorithm=sha256 name="myserver.hu" nat-traversal=no proposal-check=strict
```

Természetesen, ha a Te esetében valamelyik peer egy NAT-olt hálózaton van, akkor állítsd be a `nat-traversal=yes` attribútumot. A mi példánkban erre nincs szükség.

Érdekes az elnevezés. Bár ez is egy proposal, de ennek profile a neve. Ennek megvan a maga magyarázata. Elméletileg minden egyes peer -hez külön algoritmus beállításokat lehet megadni. A gyakorlatban azonban ez egyáltalán nem jellemző. A gyakorlatban az összes lehetséges peer-re, vagy legalábbis a peer-ek nagy számosságú csoportjaira azonos algoritmusokat szoktunk használni. Emiatt az algoritmusok beállítása egy külön `ipsec profile` entitásba került, és névvel van ellátva. A peer-ek létrehozásakor nem a konkrét algoritmus neveket adjuk meg, hanem csak hivatkozunk arra a profile-ra ami meghatározza az algoritmusokat. Ez egyébként azért is jó, mert a profile módosításával egy lépésben át lehet konfigurálni az összes peer-t, ami hozzá van rendelve.

Természetesen nincs annak semmi akadálya, hogy minden egyes peer-hez egy külön profile-t hozz létre, és így peer-enként egyedileg tudd beállítani az algoritmusokat. De ez általában kontraproduktív. A mi konkrét példánkban egyetlen egy peer lesz, de mivel RouterOS-ben a konfigurációnak ilyen a szerkezete, ezért ehhez kell igazodnunk.

A `proposal-check=strict` beállítással azt írjuk elő, hogy a második fázisban létrehozott SA-k milyen maximális életkorral rendelkezhetnek. (Az első fázisban az életkort is egyeztetjük.) A `strict` beállítás szigorú, a responder csak akkor fogadja el az initiator által kínált max. életkort, ha az nem nagyobb, mint ami a responder oldalon alapértelmezésként meg van adva.

## Phase 2 proposal beállítás (Ipsec proposal)

A második fázishoz külön be kell állítanunk a lehetséges algoritmusokat (amiket korábban tárgyaltunk). [Ez RouterOS-ben az `/ip ipsec proposal` menüpont alatt található.](#)

Ezt mindkét router-en futtatni kell:

```
/ip ipsec proposal
add auth-algorithms=sha256 enc-algorithms=aes-256-cbc pfs-group=modp2048 name="proposal-myserver.hu" lifetime=30m
```

Az alapértelmezett lejáratási idő 30 perc, ezt az igényeknek megfelelően módosítani lehet.

## Peer-ek létrehozása

Először a teljes verziót írom le, amikor mindkét oldalon mindkét címet megadod.

Az office router-en:

```
/ip ipsec peer
add exchange-mode=ike2 address=100.2.2.10/32 local-address=100.3.3.10 name="peer-branch01"
passive=yes send-initial-contact=yes profile="myserver.hu"
```

A branch01 router-en:

```
/ip ipsec peer
add exchange-mode=ike2 address=100.3.3.10/32 local-address=100.2.2.10 name="peer-office"
passive=no send-initial-contact=yes profile="myserver.hu"
```

### Egyszerűsített verzió

A responder oldalon (`passive=yes`) nem szükséges megadni az `address` paramétert. Ha ezt elhagyod, akkor az `0.0.0.0/0` -nak felel meg. A responder oldalon a peer `address` paramétere arra való, hogy ezen keresztül korlátozni és/vagy azonosítani tudjuk az initiator-okat az IP címük alapján. Azoknál a konfigurációknál, ahol nem ismered előre az initiator IP címét, az `address` paramétert elhagyhatod. Ezen felül a `local-address` -t se kötelező megadni. De ha ez nincs megadva, akkor az adott peer minden interface-en és minden címen keresztül használható lesz. A responder oldalon az `address` paraméter kizárólag ip cím prefix lehet. (Konkrét IP cím /32 prefixnek felel meg.) A responder oldalon általában nem szükséges minden klienshez külön peer-t létrehozni.

Az initiator oldalon (`passive=no`) az `address` paramétert minden esetben kötelező megadni. Ez mondja meg, hogy hol van a responder, amivel föl kell venni a kapcsolatot. Az initiator oldalon az `address` paraméter értéke nem csak IP cím lehet, hanem fqdn is. Ha a kapcsolat megszakad, akkor az újrapcsolódás előtt a RouterOS `resolve`-ozza az ott megadott host nevet. (A helyi DNS cache TTL figyelmen kívül hagyásával.) Ez lehetőséget ad arra, hogy olyan VPN szerverrel alakíts ki kapcsolatot, aminek a címe dinamikusan változik.

A problémák akkor jelentkeznek, amikor különböző `phase1 proposal`-t (`/ip ipsec profile`) szeretnél megadni a különböző peer-ekhez a responder oldalon. Ilyenkor muszáj megadnod az `address` paramétert. (Mert ha azt nem adod meg, akkor a RouterOS számára az első illeszkedő peer-t fogja felhasználni, a többi elérhetetlen lesz.) Ez a probléma csak akkor fordulhat elő, ha a responder oldalon több peer is van. Erről a problémáról egy későbbi fejezetben lesz szó. (Lásd még:

<https://forum.mikrotik.com/viewtopic.php?f=2&t=172945&p=845929#p845929> )

### Érdekességek:

- A `local-address` azt a címet adja meg, amelyiken az IKE démon várja a bejövő kapcsolatokat. (Ezt nem kötelező megadni, ha nem adod meg, akkor a peer bármely címre beérkező kulcs-csere kéréshez használható.)

- A `passive` azt jelenti, hogy az IKE démon másik oldalon levő peer-re vár a kapcsolat felépüléséhez. Ha a passzív mód ki van kapcsolva, akkor a peer megpróbálja automatikusan végrehajtani nem csak az első fázist, hanem a másodikat is (feltéve hogy az első fázis után vannak létrehozva a peer-nek megfelelő policy-k). RouterOS-ben mindkét oldal lehet initator és responder is.
- A `send-initial-contact` beállítás hatására a egy "initial contact" csomagot küld a peer-nek. Ennek hatására a (távoli) peer kitörli az összes korábban létrehozott SA-t, ami a helyi peer forráscíméhez tartozik.

Bővebb leírás itt: <https://wiki.mikrotik.com/wiki/Manual:IP/IPsec#Peers>

## Identity hozzáadása

Ahogy korábban írtuk, ez szolgál a peer-ek kölcsönös azonosítására. Az azonosítást a korábban létrehozott kulcsokkal és tanúsítványokkal valósítjuk meg.

### x509 tanúsítvány alapú identity hozzáadása

Az office router-en:

```
/ip ipsec identity
add auth-method=digital-signature certificate=office.myserver.hu match-by=certificate my-id=fqdn: office.myserver.hu peer=peer-branch01 remote-certificate=branch01.myserver.hu remote-id=fqdn: branch01.myserver.hu
```

A branch01 router-en:

```
/ip ipsec identity
add auth-method=digital-signature certificate=branch01.myserver.hu mode-config=modeconf-branch01 my-id=fqdn: branch01.myserver.hu peer=peer-office remote-id=fqdn: office.myserver.hu
```

Egy kis magyarázat hozzá:

- Az office router esetében megvan mindkét certificate. Az office.myserver.hu és a branch01.myserver.hu nevű is. Mivel a branch01 tanúsítványait a jövőben is az office router-en kívánjuk kiállítani, ezért ez a jövőben is mindig rendelkezésre fog állni. Emiatt az azonosításnál a `match-by=certificate` értéket adtuk meg. Ez azt jelenti, hogy az azonosítást a teljes certificate pontos egyezése alapján végzi el. Ha ezt megadod, akkor persze kötelező megadni a `remote-certificate` értékét is (tudnia kell, hogy a távoli peer tanúsítványát mivel hasonlítsa össze).
- A branch01 router esetében nem rendelkezünk az office tanúsítványával. Bár elméletileg ezt az office routeren kiexportálhattuk volna (privát kulcs nélkül!), és a branch01 routeren importálhattuk volna. Ebben az esetben ott is használhatnánk a `match-by=certificate` beállítást. De ha így tennénk, akkor ez azt jelentené, hogy minden alkalommal amikor az



office gép tanúsítványa lejár, azt újra kellene exportálni és újra föl kellene másolni a branch01 routerre is. Ez ebben a példában nem tűnik problémásnak. De képzeljük el azt a helyzetet, amikor az office router-hez már 20 különböző branch-et csatlakoztattunk. Könnyen abban a helyzetben találhatjuk magunkat, hogy a VPN responder tanúsítványának megújítása után 20 különböző gépre kell fölmásolni és beállítani az új tanúsítványt. Ez elég kényelmetlen lenne. Ehelyett használhatjuk a `match-by=remote-id` és a `remote-id=fqdn:` beállítást. Ez a beállítás a következőt teszi:

- Először természetesen (mint mindig) ellenőrzi a távoli peer tanúsítványának digitális aláírását és lejáratát. Itt használjuk ki a PKI nyújtotta előnyt: ellenőrizni tudjuk a távoli peer megbízhatóságát egy harmadik fél (a CA) segítségével. (Bár a mi példánkban a CA -t is mi valósítjuk meg, de a CA cert lejáratát 10 év.)
- Ha ezt rendben találja, akkor a `common-name` mezőjében megkeresi, hogy milyen `fqdn`-re szól a tanúsítvány
- Ezt hasonlítja össze a `remote-id: fqdn:` mezőben megadott fqdn értékkel.

Az egyeztetést sok más módon el lehet végezni. Ezeket nem sorolom föl itt, mert a mi példánkhoz nem szükségesek, és tényleg nagyon sok lehetőség van. Bővebb leírást itt találsz:

<https://wiki.mikrotik.com/wiki/Manual:IP/IPsec#Identities>

## Előre megosztott kulcs alapú identity hozzáadása

office oldalon:

```
/ip ipsec identity
add auth-method=pre-shared-key secret="****your_shared_key*****" my-
id=fqdn: office.myserver.hu remote-id=fqdn: branch01.myserver.hu peer=peer-branch01
```

branch01 oldalon:

```
/ip ipsec identity
add auth-method=pre-shared-key secret="****your_shared_key*****" my-
id=fqdn: branch01.myserver.hu remote-id=fqdn: office.myserver.hu peer=peer-office
```

## IKE működésének ellenőrzése (első fázis)

Ezen a ponton a két router IKE démonjainak tudni kell azonosítani egymást, és SA-kat készíteni.

Ha megnézed a naplót (`/log print`) akkor az office oldalon valami ilyesmit kell látnod:

```
10: 58: 30 ipsec,info new ike2 SA (R): 100.3.3.10[4500]-100.2.2.10[4500]
spi: 7d0560af36a26da3: 4caa053a4ee2453e
10: 58: 30 ipsec,info,account peer authorized: 100.3.3.10[4500]-100.2.2.10[4500]
spi: 7d0560af36a26da3: 4caa053a4ee2453e
```

```
10: 58: 30 ipsec, error no policy found/generated
```

A branch01 oldalon pedig ilyet:

```
10: 58: 31 ipsec, info new ike2 SA ( I): 100. 2. 2. 10[ 4500] - 100. 3. 3. 10[ 4500]
spi: 4caa053a4ee2453e: 7d0560af36a26da3
10: 58: 31 ipsec, info, account peer authorized: 10. 2. 2. 10[ 4500] - 100. 3. 3. 10[ 4500]
spi: 4caa053a4ee2453e: 7d0560af36a26da3
```

Ha nem ezt látod, akkor ne menj tovább. Nincs értelme további dolgokat beállítani addig, amíg ez nem működik.

## Policy-k létrehozása manuálisan

A responder oldalon a napló utolsó sorában látható egy hibaüzenet: `no policy found/generated`. Ez azért számít hibának, mert policy megadása nélkül az IKE démon csak az első fázist tudja befejezni. A második fázishoz azért szükséges a policy-k megadása, mert az IKE démon minden policy-hoz külön SA-t generál. Mivel mi az `identity` -nél mindkét router-nél a `generate-policy=no` beállítást adtuk meg, ezért a rendszer nem generál dinamikusan policy-eket. Helyette csak azokat tudja használni, amiket előre (kézzel) fölveltünk. De mivel ilyeneket még nem vettünk föl, ezért nincsen egyetlen egy policy se, amihez a második fázis lefuttatásával SA-t tudna generálni. Így tehát egyetlen egy SA se jön létre. Ez pedig nyilvánvalóan hiba (hiszen SA-k hiányában semmiféle IPSEC kommunikáció nem lehetséges).

Az office oldalon a következő policy-t vesszük föl:

```
/ip ipsec policy
add dst-address=172.16.1.0/24 peer=peer-branch01 proposal=proposal-myserver.hu src-
address=172.16.2.0/24 tunnel=yes
```

A branch01 oldalon ennek a párját:

```
/ip ipsec policy
add dst-address=172.16.2.0/24 peer=peer-office proposal=proposal-myserver.hu src-
address=172.16.1.0/24
```

Egy kis magyarázat a beállításokhoz:

- Az `src-address` és a `dst-address` mondja meg azt, hogy milyen csomagokat szeretnénk enkapszulálni. Ez az eredeti csomag forrás- és célcímére vonatkozó feltétel. Itt nem csak konkrét címeket, hanem címtartományokat is meg lehet adni.
- Lehetne megadni feltételt `src-port` és `dst-port` -ra is, illetve `protocol` -ra (tcp, udp) is. Ezt mi most nem tesszük meg, mert a két telephely közötti kapcsolatot nem szeretnénk

korlátozni. De jó ha tudod: a megfelelő policy-k és feltételek megadásával el tudod érni például azt, a titkosítást szelektíven, csak bizonyos szolgáltatásokhoz tartozó adatforgalomra korlátozod.

- A `peer` mondja meg, hogy melyik peer felé kell küldeni az enkapszuláció után létrejött új IPSEC csomagot.
- A `tunnel=yes` beállítással írjuk elő az alagút módot.

Ha beírod hogy `/ip ipsec policy export` akkor azt fogod látni, hogy az exportba beleírja az `sa-src-address` és `sa-dst-address` attribútumokat is.

```
[admin@branch01.myserver.hu] /ip ipsec policy> export
# jan/09/2021 14:34:57 by RouterOS 6.46.8
# software id =
#
#
#
/ip ipsec policy group
add name=office.myserver.hu
/ip ipsec policy
add dst-address=172.16.2.0/24 peer=peer-office proposal=proposal-myserver.hu sa-dst-address=100.3.3.10 sa-src-address=100.2.2.10 src-address=172.16.1.0/24 tunnel=yes
[admin@branch01.myserver.hu] /ip ipsec policy>
```

Ezek valójában ready-only attribútumok, és a policy-hez rendelt peer alapján automatikusan vannak meghatározva. Az egy RouterOS bug, hogy ezt is beleteszi az exportba. Amikor a policy-k módosításán dolgozol, akkor ez ne tévesszen meg.

## IKE működésének ellenőrzése (második fázis)

A fenti két policy létrehozása után ezeknél a policy-knél meg kell hogy jelenjen az `A` flag, ami azt jelzi hogy aktív.

Az office router-en:

```
[admin@office.myserver.hu] /ip ipsec policy> print detail
Flags: T - template, X - disabled, D - dynamic, I - invalid, A - active, * - default
0 T * group=default src-address=::/0 dst-address=::/0 protocol=all proposal=default
template=yes

1 A peer=peer-branch01 tunnel=yes src-address=172.16.2.0/24 src-port=any dst-
address=172.16.1.0/24 dst-port=any protocol=all action=encrypt level=require ipsec-
protocols=esp sa-src-address=10.3.3.10 sa-dst-address=10.2.2.10 proposal=proposal-
myserver.hu ph2-count=1
[admin@office.myserver.hu] /ip ipsec policy>
```

A branch01 router-en:

```
[admin@branch01.myserver.hu] /ip ipsec policy> print detail
Flags: T - template, X - disabled, D - dynamic, I - invalid, A - active, * - default
```

```
0 T * group=default src-address=::/0 dst-address=::/0 protocol=all proposal=default
template=yes

1 A peer=peer-office tunnel=yes src-address=172.16.1.0/24 src-port=any dst-
address=172.16.2.0/24 dst-port=any protocol=all action=encrypt level=require ipsec-
protocols=esp sa-src-address=10.2.2.10 sa-dst-address=10.3.3.10 proposal=proposal-
myserver.hu ph2-count=1

[admin@branch01.myserver.hu] /ip ipsec policy>
```

A `default` nevű nullás policy template mindkét oldalon megjelenik, ezzel most nem kell foglalkozni.

Ezen felül, a `/ip ipsec installed-sa` menüpont alatt meg tudod nézni az IKE második fázisában generált SA-kat. Ez az office router-en például valahogy így néz ki:

```
[admin@office.myserver.hu] /ip ipsec installed-sa> print
Flags: H - hw-aead, A - AH, E - ESP

0 E spi=0x798ACAD src-address=100.2.2.10 dst-address=100.3.3.10 state=mature auth-
algorithm=sha256 enc-algorithm=aes-cbc enc-key-size=256 auth-
key="c2c879d6e7db7221df2ddb3e14b139e7a614ea54a85cabbe672cc0f8a6942d67" enc-
key="3f87e644e9d5085bb2f19c55cb111f4662ab4140187d898d6eddedd5989211b6" add-
lifetime=24m6s/30m8s replay=128

1 E spi=0x40A1E9 src-address=100.3.3.10 dst-address=100.2.2.10 state=mature auth-
algorithm=sha256 enc-algorithm=aes-cbc enc-key-size=256 auth-
key="72819bd20e06f814859e44bd35e47b848e05c9230ca0d0006e56962bf89e71e5" enc-
key="6900d551d05d522263402c7231e7de154c3c08d58bca850d9aa31f75cbf5f333" add-
lifetime=24m6s/30m8s replay=128

[admin@office.myserver.hu] /ip ipsec installed-sa>
```

Itt látható, hogy a két peer között két SA jött létre (a küldésre és a fogadásra). Ezen felül látszódnak a választott algoritmusok, a titkosító kulcsok, és az életkor.

Ha megfelelő hardvert és megfelelő algoritmus beállításokat használtál, akkor itt az E betű mellett megjelenik egy H betű is. Ez jelzi azt, hogy az SA-hoz elérhető a hardveres gyorsítás. A mi példánkban ez azért nincs kiírva, mert a teszt során egy virtuális gépen futtattam egy demo router-t, és abban nincsen ilyen támogatás.

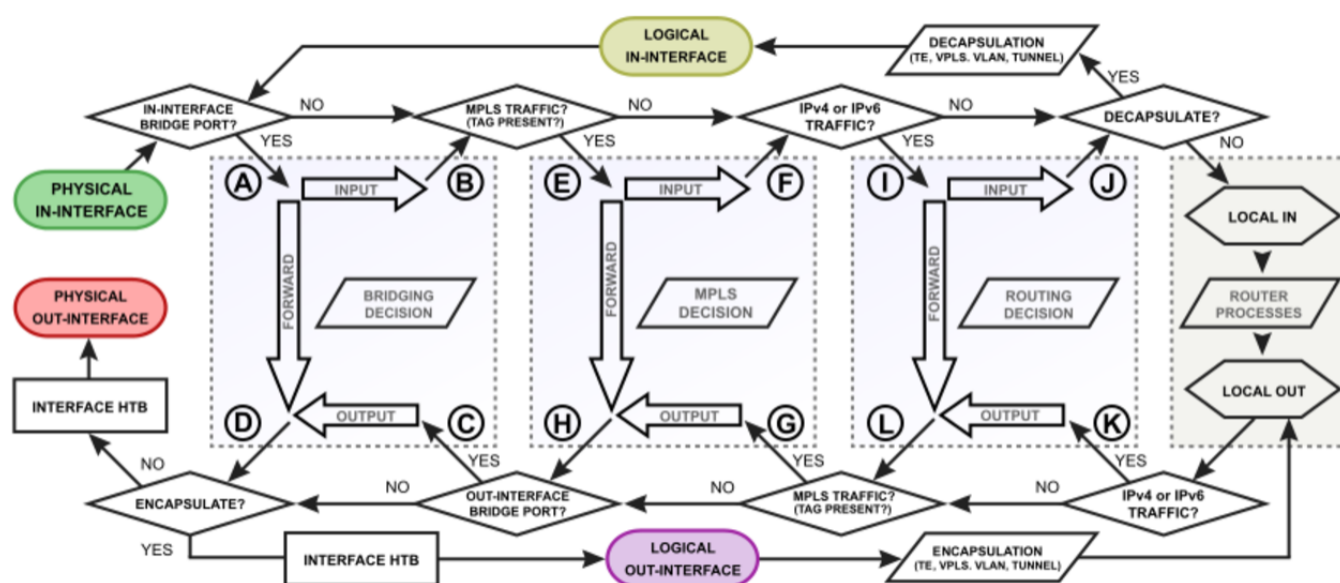
## Hiányzó NAT bypass szabályok

Ha ezek után megpróbáljuk mondjuk az office telephelyen levő PC2 gépről elérni a branch01 gépen levő PC1 gépet, akkor a következőt fogjuk látni:

```
PC2> ping 172.16.1.10 -c 3
*100.3.3.3 icmp_seq=1 ttl=63 time=0.747 ms (ICMP type:3, code:0, Destination network unreachable)
*100.3.3.3 icmp_seq=2 ttl=63 time=0.752 ms (ICMP type:3, code:0, Destination network unreachable)
*100.3.3.3 icmp_seq=3 ttl=63 time=0.702 ms (ICMP type:3, code:0, Destination network unreachable)
```

Ez elsőre érthetetlennek tűnhet. Az az ICMP csomag ami a 172.16.2.10 gépről indult el a 172.16.1.10 irányába, elvileg illeszkedik az általunk létrehozott policy-ra, és becsomagolás után az alagúton kellene hogy átmenjen. De láthatóan nem ez történik. Helyette az office router ezt a csomagot továbbküldte az "internet" irányába, és az "ISP" fő routere (10.3.3.3) azt a választ küldte rá, hogy ezt erre a privát címre ő nem tud útvonalat választani. Ha az ICMP hiba a 10.3.3.3 -tól jött vissza akkor a csomag egyértelműen kiment az internetre, méghozzá titkosítatlanul.

Mi történik itt? A magyarázathoz ismét a packet flow diagramhoz kell visszatérnünk.



Ahogy korábban írtuk, a csomagok enkapszulációja (a policy szabályok futtatása) az egyik legutolsó lépés. A switching és a routing jóval hamarabb fut le, mint az enkapszuláció. Az office router tűzfalában be volt állítva egy ilyen NAT szabály:

```
/ip firewall nat
add action=masquerade chain=srcnat out-interface=ether1-internet
```

Ez a **címfordítás** szükséges ahhoz, hogy az office és branch01-en belül található, **privát címmel** rendelkező gépek el tudják érni az interneten található nyilvános szolgáltatásokat. De mi történik akkor, amikor nem az internetet, hanem a másik privát LAN-on belül található címet akarjuk elérni? Az az ICMP ping csomag aminek a forráscíme 172.16.2.10 és a célcíme 172.16.1.10, illeszkedik erre a NAT szabályra. Ez azért van, mert az illeszkedés egyetlen feltétele az, hogy a kimenő interface az ether1-internet interface legyen.

Hogy pontosan miért illeszkedik, azt alább még bővebben kifejtem. A **layer 3 routing** akkor kezdődik el, amikor a csomag áthalad a fenti ábrán az **I** vagy **K** betűvel jelölt csomópontok valamelyikén. Ez belül így néz ki:



Az IPSec policy-k jóval az útvonalválasztás után futnak le. A mi példánkban a 172.16.1.0/24 célcímhez keresünk útvonalat. Az office router saját routing táblázatában a 172.16.1.0/24 alhálózathoz nincsen megadva külön route. A layer 3 routing szempontjából nézve az a megfelelő route amin keresztül ez a célcím elérhető, a default gateway. Ehhez az ether1-internet interface tartozik. Ez egy WAN interface! A masquerade NAT szabály pedig erre illeszkedik:

```
/ip firewall nat
add action=masquerade chain=srcnat out-interface=ether1-internet
```

Mivel ez a NAT szabály illeszkedik, ezért a router lecseréli a forráscímét arra a címre, ami az ether1-internet interface-en van. Mire a csomag eléri az IPSEC policy-kat, addigra a forráscíme 10.3.3.10 lett. Erre nem illeszkedik egyik policy sem, ezért akadály nélkül távozik az ether1-internet interface-en. Beérkezik az ISP routeréhez, és ez egyből ICMP hibát küld vissza (nincs hozzá útvonal).

Ezt azért magyaráztam el ennyire részletesen, hogy belássuk: a probléma nem csak annyiból áll, hogy a csomag nem jut el a céljához. A csomag távozik az internet irányába, ráadásul titkosítatlanul! Ez TCP kapcsolatok esetén még talán nem túl nagy probléma, mert ott csak TCP SYN csomagok tudnak kimenni (a kapcsolat nem épül föl). De UDP csomagok esetében ez azt jelenti, hogy a rosszul beállított routing és NAT szabályok miatt pont az ellenkezőjét értük el annak, amit akartunk: önként küldjük a potenciálisan bizalmas adatokat a nagyközönség felé. Ahogy azt később látni fogjuk, ez a probléma fokozottan jelentkezhet azokban az esetekben, amikor a VPN (kliens) oldalon nem kézzel fölvelt, statikus policy-kat használunk, hanem egy template-ből generáljuk őket. Ha egy hiba miatt a VPN kapcsolat nem épül föl, akkor a dinamikus policy-k nem jönnek létre, és ennek hatására az internet irányába titkosítatlan csomagok hagyhatják el a csomópontot. Ezt esetleg nehezebb lehet észrevenni, mert csak VPN kapcsolat hiba esetén jelenik meg.

Azt a jelenséget, melynek során a router a hiányzó vagy hibás beállításából fakadóan rossz helyre küldi a csomagokat, szokás úgy nevezni hogy `package leaking`.

Ezen problémák megszüntetéséhez két dolgot kell tennünk.

## NAT bypass

Az egyik megoldás az, hogy úgy módosítjuk az src-nat szabályokat, hogy ezeknek a csomagoknak a forrás címét nem írjuk át. Ezt persze mindkét telephelyen meg kell tenni ahhoz, hogy oda- és vissza is illeszkedjenek az ipsec policy-k.

Például az office route-en ezt megtehetjük így:

```
/ip firewall nat
add chain=srcnat src-address=172.16.2.0/24 dst-address=172.16.1.0/24 action=accept place-
before=0
```

Ennek az eredménye egy ilyen NAT táblázat:

```
[admin@office.myserver.hu] /ip firewall nat> print detail
Flags: X - disabled, I - invalid, D - dynamic
 0 chain=srcnat action=accept src-address=172.16.2.0/24 dst-address=172.16.1.0/24
 1 chain=srcnat action=masquerade out-interface=ether1-internet
```

Azok a csomagok amikre később illeszkedik az ipsec policy, illeszkednek a legelső NAT szabályra is. Mivel ott `action=accept` van, ezért ezekre a csomagokra a NAT feldolgozás befejeződik, a második `action=masquerade` szabály sosem fut le rájuk.

Ha ezt így próbáljuk meg kezelni, akkor előre ismernünk kell, hogy milyen cél alhálózatok esetében nem akarunk masquerade/srcnat címfordítást végezni. (A mi site-to-site példánkban ez a helyzet.)

Ha sok ilyen cél alhálózat van, akkor az összes ilyen szabály felsorolása és a policy-kkal való szinkronban tartása fáradságos munka lehet, ráadásul könnyű hibázni. Ha az összes IPSEC policy-re illeszkedő csomagnál el akarjuk kerülni a címfordítást, akkor egyszerűbb ha egy plusz feltételt adunk hozzá, az alábbi módon:

```
[admin@branch01.myserver.hu] /ip firewall nat> print
Flags: X - disabled, I - invalid, D - dynamic
 0 chain=srcnat action=masquerade out-interface=ether1-internet log=yes
[admin@branch01.myserver.hu] /ip firewall nat> set 0 ipsec-policy=out,none
[admin@branch01.myserver.hu] /ip firewall nat> print
Flags: X - disabled, I - invalid, D - dynamic
 0 chain=srcnat action=masquerade out-interface=ether1-internet log=yes ipsec-policy=out,none
[admin@branch01.myserver.hu]
```

Ez a plusz feltétel az ipsec-policy, ami mindig két, vesszővel elválasztott értéket tartalmaz: irány és policy.

Az irány (első) érték lehet `in` vagy `out`. Az `in` érték csak a prerouting, input és forward chain-ekben használható. Az `out` érték csak a postrouting, output és forward chain-ekben használható. Azt már korábban bemutattam, hogy az srcnat chain a postrouting-ban van. Tehát kizárásos alapon itt az `out` -ot kell használnunk. (Az egyedüli chain ahol az `in` és az `out` érték is használható, az a forward chain.)

A policy (második) érték lehet `ipsec` vagy `none`. Az ipsec érték azt jelenti, hogy a van olyan ipsec policy, ami illeszkedik a csomagra, ezért ez alapján a csomaggal később ipsec műveletet (enkapszuláció vagy dekapszuláció) kell végezni. A none pedig ennek az ellenkezőjét jelenti: nincs olyan ipsec policy, ami illeszkedik a csomagra.

Tehát az `ipsec-policy=out,none` feltétel hozzáadásával azt írjuk elő, hogy az action=masquerade művelet csak azokra a csomagokra fusson le, amikhez nincsen megfelelő ipsec policy.



Ha ezek után megpróbáljuk az egyik alhálózatban levő gépről elérni a másik alhálózatban levő gépet akkor azt látjuk, hogy a VPN alagút működik, és a két alhálózat gépei látják egymást.

```
PC2> ping 172.16.1.10 -c 3
84 bytes from 172.16.1.10 icmp_seq=1 ttl=62 time=2.108 ms
84 bytes from 172.16.1.10 icmp_seq=2 ttl=62 time=1.432 ms
84 bytes from 172.16.1.10 icmp_seq=3 ttl=62 time=1.262 ms
```

Ezzel azonban csak a probléma egyik felét oldottuk meg. Egyrészt elképzelhető, hogy a VPN peer vagy policy kikapcsolása/törlése után titkosítatlan csomagok szöknek ki az internetre. Másrészt ha megpróbáljuk magáról a router-ről elérni a távoli telephely egy gépét, akkor azt fogjuk látni, hogy ez nem működik:

```
[admin@branch01.myserver.hu] > /ping 172.16.2.10 count=1
SEQ HOST                                SIZE TTL TIME STATUS
  0 100.2.2.2                            84  64 0ms  net unreachable
sent=1 received=0 packet-loss=100%
```

Ennek a magyarázata az következő. Az egyetlen olyan route, ami alapján a 172.16.2.0/24 célhálózat elérhető, a default route (ISP-n keresztül). Ne feledjük: az ipsec policy egy külön réteg. Van policy-nk a a 172.16.2.0/24 célhálózathoz, de route-unk nincsen! A RouterOS ping parancsa ezért olyan ICMP csomagokat hoz létre, aminek a forráscíme megegyezik a default route-hoz tartozó címmel. Tehát a ping csomagok forráscíme a 10.2.2.10 cím lesz, és nem a 172.16.1.1 cím.

Ez a probléma akkor is megjelenik, amikor traceroute-ot akarsz futtatni a telephelyek közötti, alagúton keresztül haladó útvonalra:

```
PC1> trace 172.16.2.10 -P 1
trace to 172.16.2.10, 8 hops max (ICMP), press Ctrl+C to stop
 1  172.16.1.1    0.314 ms   0.252 ms   0.213 ms
 2      *      *      *
 3  172.16.2.10   1.288 ms   0.926 ms   0.907 ms

PC1> █
```

A második hop címe azért nem látszódik, mert a VPN alagúton való átjutáshoz az enkapszuláció után a RouterOS egy "logical out-interface" nevű logikai (belső) interface-t használ. Ez látszódik a packet flow ábrán is. Ez a belső interface nem érhető el a RouterOS `/interface` menüjéből. Nincsen neve és nincsen címe sem. Bár az igaz, hogy amikor ezt az interface-t eléri az ICMP ping csomag, akkor csökkentjük a TTL értékét, és ha nullára csökkent akkor eldobjuk a csomagot. Azonban a RouterOS nem küld vissza "ICMP Time Exceeded" üzenetet, mivel ehhez szükség lenne a forráscím megadására. Tehát a VPN alagúton való áthaladás csökkenti a TTL értékét, de az ICMP alapú traceroute-ban nem látszódik ennek az állomásnak a címe, mert az a belső interface ami a TTL-t csökkenti az alagúton való áthaladáskor, nem rendelkezik olyan forrás címmel, amit föl lehetne használni az ICMP válasz visszaküldésére.

A traceroute-ban így csak annyit látunk, hogy interneten való (potenciálisan sok router-en keresztüli) áthaladás helyett az alagút egyetlen (ismeretlen című) hop-ként látszódik.

## Csomag szökések megakadályozása

A fenti példák szemléltetik, hogy szükséges megakadályozni a privát hálózati címekhez tartozó csomagok "szökését" az internet felé. Ezt több lépésben tesszük meg.

Az első lépésben explicit "elérhetetlen" route-okat hozunk létre az [összes privát címtartományhoz](#).

```
/ip route
add comment="Prevent package leak RFC1918 class A" distance=1 dst-address=10.0.0.0/8
type=unreachable
add comment="Prevent package leak RFC1918 class B" distance=1 dst-address=172.16.0.0/12
type=unreachable
add comment="Prevent package leak RFC1918 class C" distance=1 dst-address=192.168.0.0/16
type=unreachable
```

Ez biztosan megakadályozza a csomagok megszökését, mert ezek a route-ok mindig specifikusabbak, mint a 0.0.0.0/0 default route. De az IPSEC szempontjából ez problémás. Ahhoz, hogy az ipsec policy működőképes maradjon, szükség van valamilyen aktív, elérhető route-ra a cél címek irányában.

Az office router-en ezt így adjuk hozzá:

```
/interface bridge
add name=ipsec protocol-mode=none
/ip route
add dst-address=172.16.1.0/24 gateway=ipsec pref-src=172.16.2.1 comment="VPN to branch01"
```

A branch01 router-en pedig így:

```
/interface bridge
add name=ipsec protocol-mode=none
/ip route
add dst-address=172.16.2.0/24 gateway=ipsec pref-src=172.16.1.1 comment="VPN to office"
```

A `pref-src` megadása miatt az összes router-ről a távoli privát hálózatba indított forgalom (például ping) automatikusan az ott megadott címet fogja használni. Ezen felül, a pref-src hatására a traceroute is elkezd "jól" működni:

```
PC1> trace 172.16.2.10
trace to 172.16.2.10, 8 hops max, press Ctrl+C to stop
 1  172.16.1.1    0.386 ms  0.266 ms  0.276 ms
 2  172.16.2.1    0.877 ms  0.833 ms  0.802 ms
 3  *172.16.2.10  0.939 ms (ICMP type:3, code:3, Destination port unreachable)
```

Amikor a TTL értéket nullára csökkenti a távoli router, akkor képes lesz visszaküldeni ICMP "time exceeded" üzenetet, mert a pref-src beállítás alapján képes lesz forráscímet meghatározni az ICMP válaszhhoz.

Ha teljesen perfekcionista akarsz lenni, akkor még ezt a tűzfal szabályt is hozzáadhatod **a megfelelő pozícióba**:

```
/ip firewall filter
add action=reject chain=forward out-interface=ipsec reject-with=icmp-network-unreachable
comment="Reply with network-unreachable when IPSEC tunnel is down"
```

Ha valami miatt a VPN nem működik (pl. letiltottad a policy-t), akkor a router az alapértelmezett "host is unreachable" helyett egy kicsit korrektebb "network is unreachable" ICMP választ küld majd vissza.

## MTU, TCP MSS Clamping

Az elméleti fejezetben [már részletesen beszámoltam](#) arról, hogy mi az az MTU, és mi az a TCP MSS Clamping. Most csak a gyakorlati beállítást mutatom be. Egy olyan szabályt veszünk föl, ami képes lecsökkenteni a `TCP SYN` csomagok 1360 byte-nál nagyobb `MSS` értékét 1360-ra. Nagyon fontos hogy olyan szabályt vegyünk föl, ami nem képes növelni az MSS értékét. Ha a csomag már eleve 1360 -nál kisebb MSS értékkel érkezik be, és ezt felülírjuk 1360-ra, akkor ezzel **megnöveljük azt**, és így végül pont az ellenkezőjét érjük el annak, mint amit szeretnénk (növeljük a fragmentációt ahelyett hogy csökkentenénk).

A megfelelő szabály az office gépen így néz ki:

```
/ip firewall mangle add action=change-mss chain=forward new-mss=1360 src-address=172.16.1.0/24
protocol=tcp tcp-flags=syn tcp-mss=! 0-1360 ipsec-policy=in,ipsec passthrough=yes
comment="IKE2: Clamp TCP MSS from 172.16.1.0/24 to ANY"
```

A branch01 gépen pedig így:

```
/ip firewall mangle add action=change-mss chain=forward new-mss=1360 src-address=172.16.2.0/24
protocol=tcp tcp-flags=syn tcp-mss=! 0-1360 ipsec-policy=in,ipsec passthrough=yes
comment="IKE2: Clamp TCP MSS from 172.16.2.0/24 to ANY"
```

Egy kis magyarázat hozzá:

- Az `src-address`, `ipsec-policy`, `protocol=tcp` és `tcp-flags=syn` együttes használatával kiszűrjük azokat a `TCP SYN` csomagokat, amik **már átjöttek** az alagúton. (Az `src-address` szűrőben az office router szabályánál ezért szerepel a branch oldal alhálózata.) Ez a szabály így azért jó, mert garantáltan csak azokra a TCP kapcsolatokra módosítja az MSS-t, amik már átjöttek az alagúton. (Azokra nem szeretnénk módosítani, ami nem megy át alagúton!)
- A `tcp-mss=! 0-1360` szabály szó szerint azt jelenti, hogy "a tcp mss értéke nem nulla és 1360 között van". Ez könnyebben értelmezhető formában annyit tesz, hogy nagyobb mint 1360.
- A `passthrough=yes` azért került oda, mert az MSS megváltoztatása után nem szeretnénk átugrani a többi mangle szabályt. (Bár a jelenlegi példában nincsen több mangle szabály, de ha lennének akkor valószínűleg nem akarnánk átugrani őket.)

---

Revision #23

Created 9 January 2021 14:17:40 by Gandalf

Updated 27 February 2021 18:32:32 by Gandalf