IKE - a kulcs csere alapjai

Az `IKE` protokoll célja, hogy két egymással kommunikálni kívánó fél számára biztonságos módon meghatározzon olyan kriptografikus kulcsokat, amikkel a biztonságos kommunikáció elvégezhető. Az IKE protokoll nem csak a kulcsokat határozza meg, hanem a kulcsokhoz tartozó titkosítási algoritmusokat és egyéb olyan paramétereket is, amik szükségesek a kommunikáció lebonyolításához. Ezeket a dolgokat összefoglaló néven Security Association-nak vagy röviden `SA`-nak nevezzük. Egy SA mindig két konkrét félhez (peer) tartozik. Ha ez első olvasásra nem teljesen világos akkor gondolj úgy az SA-ra, mint egy olyan titkosító kulcsra ("jelszóra"), amit csak az egy mással kommunikáló felek ismernek. Fontos látni, hogy az IKE mindig két fél között egyezteti az algoritmusokat és a kulcsokat. Emiatt az `SA`-k mindig párban jönnek létre (a kommunikációs csatorna két oldalán.) Ennek hiányában az IPSEC kommunikáció nem lehetséges.

Az IKE protokoll egy démont (folyamatosan a háttérben futó programot) használ. Ez a démon az idő nagy részében nem csinál semmit, inaktív. Alapvetően kétféle módon lehet aktiválni:

- Ha van olyan adatforgalom ami illeszkedik egy `ipsec policy`-re, akkor az enkapszuláció elvégzéséhez szükség van egy megfelelő `SA`-ra. Ha ez az `SA` még nem áll rendelkezésre, akkor ez felébreszti az IKE démont. Ez az IKE démon fölveszi a kapcsolatot a másik oldalon levő IKE démonnal.
- A másik oldalon levő IKE démon fogadja ezt a kapcsolatot. Ez a másik mód amin keresztül egy IKE démont aktiválni lehet (távolról egy másik IKE démon által).
- A RouterOS-ben van egy olyan beállítás is (`send-initial-contact`), amivel elő lehet írni, hogy a két oldalon konfigurált peer akkor is fölvegye egymással a kapcsolatot, ha épp nincsen olyan továbbítandó csomag, amit enkapszulálni kellene. Az SA előre kiépítése felgyorsítja a válaszidőt az első csomagok elküldésénél.

Miután felébredtek, a két oldalon levő két démon elkezdi egyeztetni az algoritmusokat és a kulcsokat. Ezt UDP üzenetek segítségével valósítják meg. Az egyeztetés végeredménye (szerencsés esetben) az lesz, hogy mindkét félnél létrejön egy-egy `SA`, amivel már le lehet bonyolítani a kommunikációt. A kulcsok meghatározásához a Diffie-Hellman néven ismert kulcs csere algoritmust használják. Ez az algoritmus képes előállítani ugyan azt a titkos és véletlenszerű kulcsot a kommunikációs csatorna két oldalán úgy, hogy közben a csatornát potenciálisan figyelő többi fél nem képes hozzájutni ezekhez a kulcsokhoz (annak ellenére, hogy a kommunikáció nincs titkosítva). Az algoritmus rövid neve a továbbiakban DH algoritmus, vagy egyszerűen csak DH. A DH algoritmusnak több változata ismert, amelyek különböző bonyolultsági fokkal rendelkeznek. A magasabb bitszámot használó, erősebb titkosítást biztosító algoritmusok magától értetődően magasabb számítási igénnyel rendelkeznek. A különböző változatait a kulcs bitek számának megfelelően szokás csoportosítani. Ezek a csoportok az úgynevezett DH csoportok.

Az IKE démon az UDP/500 -as portot használja a kulcsok egyeztetésére. Ha szeretnél IPSEC/IKEv2 kommunikációt, akkor ezt a portot nyitva kell hagynod. Ez a kommunikáció normál UDP csomagokat használ, amelyek adattartalma nincs titkosítva vagy aláírva. (Ezen felül a 4500-as portot is ha NAT mögött van valamelyik fél, erről alább írok.)

Az IKEv2 protokoll az `SA`-t két fázisban hozza létre.

IKE első fázis

Az első fázisban a két fél megegyezik azokban az algoritmusokban, amiket használni fognak a kommunikációra a következő fázisban. Ezen felül meghatározásra kerül néhány olyan kulcs is, ami majd a második fázis összes kommunikációját védi. Ezeket a kulcsokat a fent említett DH algoritmussal határozzák meg. Ha gyakran és sok alagutat kell fölépíteni, akkor a DH algoritmus egy korlátozó tényező lehet (mivel számításigényes). Ebben az írásban az a célunk, hogy két fix címmel rendelkező router között permanens kapcsolatot alakítsunk ki. Itt nem lehet számítani nagy számú kapcsolat gyakori felépítésére, ezért érdemes magasabb (biztonságosabb) DH csoportot választani. A magas számítási igény miatt az első fázisban meghatározott kulcsok általában hosszú lejáratúak (több óra). Az első fázisban a sikeres egyeztetéshez a következőkben kell megegyezniük a feleknek:

- autentikációs módszer (pl. előre megadott titkos jelszó vagy tanúsítvány)
- DH csoport
- titkosítási algoritmus
- kulcs csere algoritmus
- hash algoritmus
- NAT-T
- DPD és leárati idő (opcionális)

Ezeket alább részletezem.

Authetikációs (azonosítási) módszer

A felek valamilyen módon igazolják azt, hogy tényleg azok, akinek mondják magukat. Ez a felek azonosítása, vagy szakszóval autentikáció. A legegyszerűbb esetben erre egy egymás között megosztott titkos kulcsot használnak. Egy egyszerű jelszó, angolul "pre shared key" vagy röviden `PSK`. Egy másik, nagyon elterjedt módszer az X.509 tanúsítványok használata. Ez utóbbit nehezebb beállítani, de jóval biztonságosabb. Ha nagy biztonságra törekedünk, akkor az előre megosztott kulcsok (PSK) használatát kerülni kell.

> Keress rá arra hogy `psk ike vulnerability` és találni fogsz egy csomó cikket ami arról szól, hogy a PSK azonosítás mennyire sebezhető.

A PSK használata mégis indokolt lehet némely esetben:

- Ha a végfelhasználónak nincs elég szakértelme a tanúsítvány alapú azonosítás beállításához.
- Ha a távoli csomópont nem támogatja a tanúsítvány alapú azonosítást (pl. régi szoftver verzió).
- Ha alacsonyabbak a biztonsággal kapcsolatos elvárásaink.

Vannak egyéb más azonosítási módszerek is, erre most nem térek ki. Mi ebben az írásban alhálózatok összekötését akarjuk megvalósítani. Teljes telephelyek közötti mindenféle kommunikációt kívánunk titkosítani, ezért itt érdemes erősebb, tanúsítvány alapú azonosítást használni.

> RouterOS-ben az azonosításhoz tartozó beállítások a `/ip ipsec identity` menüpont alatt találhatók. Ezen belül az `auth-method` attribútum határozza meg az azonosítási módszert.

DH csoport

Ez a már föntebb említett Diffie-Hellman algoritmus bonyolultsági fokát adja meg. A magasabb bitszámú algoritmus magasabb biztonságot ígér, de számításigényesebb.

Titkosítási algoritmus

Ez egy titkos kulcsú, szimmetrikus titkosítási algoritmus kiválasztását jelenti. Mi ebben a példában az `AES256-CBC` algoritmust választjuk, mert ez hardver támogatással rendelkezik, és elég biztonságos.

Kulcs csere algoritmus

Itt `IKE2`-őt választunk. (Akit érdekel az utánanézhet a többi típusnak, ez a cikk az IKEv2-ről szól.)

Hash algoritmus

Ez az algoritmus egy kriptografikus hasítófüggvény aminek az a legfontosabb jellemzője, hogy az invertálása túl nagy bonyolultságú ahhoz, hogy értelmes keretek között megvalósítható legyen. Mi ebben a példában az `SHA2-256` algoritmust választjuk, mert ez hardver támogatással rendelkezik, és elég biztonságos. Ezt a hash algoritmust használjuk többek között a fent említett ellenőrző összegek számításához. Közvetve ez garantálja az enkapszulált csomagok integritását. (Az ellenőrző összeget általában nem direkt módon a hash függvény határozza meg, hanem az ebből származtatott MAC kód.)

Néhány szó a NAT-T-ről

Az IPSEC protokollhoz külön módszert kellett kidolgozni a NAT mögött levő végpontok elérésére. Az IKE protkoll első és második fázisát nem befolyásolja az, ha a peer-ek közötti útvonalon áthaladó csomagok NAT-on esnek át. Ez azért van, mert az alap IKE protokoll normál UDP csomagokat

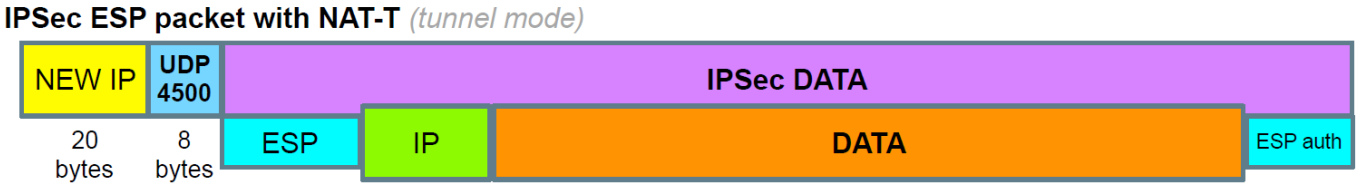
használ, és ezek forrás- és célcíme szabadon módosítható. Azonban az IPSEC által titkosított adatforgalomnál ez már problémát okoz.

Az egyik probléma a csomagok módosításával kapcsolatos:

- `ESP` csomagok esetén azért nem lehet NAT-ot végezni, mert a beágyazott eredeti csomag tartalmazhatja a forrás- és célcímet. Ezekhez a NAT-ot elvégezni kívánó router nem fér hozzá, mert az eredeti csomag titkosítva van.
- `AH` csomagok esetén az eredeti címek hozzáférhetőek, nincsenek titkosítva. Viszont a csomag integritását egy olyan hash érték biztosítja, ami érvénytelenné válik akkor, ha a csomagban levő forrás vagy célcím módosításra kerül. Így tehát az AH csomagokon elvégezhető a NAT, de a célállomásra való beérkezés után a célállomás ezeket a csomagokat eldobja, mert észreveszi hogy a csomagot valaki módosította.

A másik probléma azzal kapcsolatos, hogy a két fél közötti kommunikáció olyan útvonalon haladhat végig, amiben részt vesznek olyan router-ek, amik szintén futtatnak IKE démont és tartalmaznak IPSEC policy-ket. Ha egy ilyen router-re beérkezik egy IPSEC csomag, akkor a router esetleg elkezdheti feldolgozni (megpróbálja kicsomagolni). Valahogy el kell érni azt, hogy ezek a csomagok csak a cél csomóponton legyenek dekapszulálva.

Ezen problémák elkerülésére lefoglalták a 4500-as portot és kialakítottak egy plusz protokollt arra, hogy az IPSEC csomagok át tudjanak haladni NAT-olt hálózatokon. Az eredeti `ESP=50` illetve `AH=51` protokoll mezőt lecserélik `UDP=17`-re, és beszúrnak egy érvényes UDP header-t az IP header és az ESP/AH header közé, valamit módosítják a cél portszámot 4500-ra. Ezekkel a változtatásokkal ezek a csomagok normál UDP csomagként továbbíthatók. A hozzáadott extra UDP header miatt elvégezhető rajtuk a NAT anélkül, hogy az enkapszulált csomag integritása megsérülne.



Mivel ez a módosítás csökkenti a hasznos payload méretét (azonos MTU mellett), ezért ezt csak akkor szokás használni, ha arra lehet számítani hogy a felek között NAT történik. Ha a felek között biztosan nem történik NAT, akkor érdemes kikapcsolni ezt a funkciót, ezzel növelve az egy csomagban kiküldhető (hasznos) adat mennyiségét.

Bár ezek a problémák a kulcs csere (IKE) használatakor nem jelentkeznek, de mégis az IKE protokoll az, ami előre megegyezik abban, hogy a felek használjanak-e NAT-T módot vagy ne. Így amikor a csomagok enkapszulációja szükségessé válik, akkor a router már előre tudja, hogy el kell-e végezni a NAT-T csomag transzformációt.

Az algoritmusok egyeztetésének módja

RouterOS-ben (és általában más szoftverekben is) megadható több algoritmus. A két oldalon levő IKE démon a megadott lehetséges algoritmusok közül olyan választ, ami mindkét oldal számára megfelelő. Ezen belül a sorrend is lényeges - ha több lehetséges algoritmust adunk meg, akkor azoknak magasabb a prioritása, amelyeket előrébb sorolunk a listában.

> RouterOS-ben az ehhez tartozó menüpont az `/ip ipsec profile` és az `/ip ipsec proposal` . A profile az első fázishoz tartozik, a proposal a második fázishoz.

Ha nem található olyan algoritmus amit mindkét fél megadott a listájában, akkor a kulcscsere nem hajtható végre, és az IPSEC kapcsolat nem hozható létre.

Az algoritmusok kiválasztásánál nem csak azt kell figyelembe venni, hogy mennyire biztonságos kapcsolatot szeretnénk. Figyelembe kell venni az elérhető számítási teljesítményt (ami az adatforgalom sebességétől is függ!), valamint a távoli peer lehetőségeit. Ha például megnézzük a Windows 10 kliensek lehetőségeit akkor azt látjuk, hogy a kettes fázisban kizárólag az SHA1 hasító függvényt támogatja. Így például ha azt szeretnénk hogy Windows 10 kliens tudjon csatlakozni, akkor kénytelenek leszünk engedélyezni az SHA1 hasító függvényt. (Megjegyzés: bár maga az SHA1 már rég nem tekinthető biztonságosnak, de az SHA1 alapú MAC kódok igen.)

A mi esetünkben site-to-site kapcsolatot akarunk kialakítani két MikroTik Router között, így előre ismerjük a peer-ek képességeit, és biztonságosnak mondott algoritmusokat tudunk használni.

IKE második fázis

A második fázisban a felek (peers) létrehoznak egy vagy több SA párt. Ezek lesznek később használva az IPSEC csomag enkapszulációhoz (titkosításra és MAC generálásra). Minden IKE démon által előállított SA-nak van egy maximális életkora (lifetime). Az életkor lehet megadva lejárati időben, maximális adatforgalomban, illetve lehet akár mindkettő is. Az életkor elérése után az SA nem használható tovább a kommunikációhoz, új SA-t kell kiépíteni.

Kétféle életkor létezik. egy "soft" és egy "hard". Amikor az SA eléri a "soft" életkort, akkor az IKE démon kap egy értesítést, és újra elkezdi futtatni a második fázist. Ennek az a célja, hogy az életkor végén a kommunikáció folytatásához szükséges "friss" SA azonnal rendelkezésre álljon. Így a kommunikáció nem lesz várakoztatva a kulcsok egyeztetése miatt (az IPSEC forgalomnak nem kell várakoznia az IKE-re).

A második fázisban a feleknek a következőkben kell megegyeznie:

- Mód (alagút vagy transzport)
- IPSEC protokoll (ESP vagy AH)
- Azonosítás (autentikáció) módja
- PFS (DH) csoport
- Életkor

Ezeket alább részletezem.

Mi az a transzport és alagút mód?

A csomagok enkapszulációja során az eredeti IP csomagnak van egy forrás- és egy célcíme. Transzport mód esetében a forrás- és célcím nem kerül beágyazásra. Ezt a módot akkor lehet jól használni, ha egy olyan biztonságos kapcsolatot akarunk kiépíteni, aminél a titkosítandó csomagok forrás- és célcíme ugyan az, mint a peer-ek címe. (A mi esetünkben ez azt jelenti, hogy az egyik router az eredeti csomag küldője, a másik az eredeti csomag végleges címzettje.)

Azokban az esetekben, amikor az eredeti csomagok feladója és/vagy címzettje nem egyezik meg az enkapszulációt/dekapszulációt végző peer-ekkel, alagút módot kell használni. A mi példa feladatunk ilyen! A kapcsolatot két alhálózat között kell megvalósítanunk. Az egyik alhálózatból induló csomagok valahol beérkeznek egy router-be, ami enkapszulálja őket. Ezután az adatok titkosított IPSEC csomagok belsejében haladnak tovább egy másik router-hez. Ez a másik router elvégi a dekapszulációt, és innentől kezdve a csomagok újra titkosítatlan módon haladnak tovább az eredeti célcím irányába. Tehát az eredeti csomagok forrás- és célcíme nem egyezik meg azoknak a router-eknek a címével, amelyek az enkapszulációt/dekapszulációt végzik. Egy ilyen típusú összeköttetés kialakításához a transzport mód nyilván nem használható, mivel az eredeti csomagok forrás- és célcímeit nem lehet használni az összeköttetés kialakításához. Az IPSEC csomag célba juttatásához az kell, hogy a célcíme a távoli peer címe legyen, és ez nem egyezik meg az eredeti címzettel. Az eredeti címek általában olyan belső/foglalt hálózati címek amikhez az interneten nem lehet útvonalat választani. Ugyanakkor az eredeti forrás- és célcímek megőrzése szükséges ahhoz, hogy kicsomagolás után a távoli alhálózaton belül továbbíthatóak legyenek az eredeti csomagok az eredeti (végleges) célcímre. Tehát az IPSEC csomagok forrás- és célcíme a peer-ek címeivel egyezik meg, de valahol el kell tárolni az eredeti forrás- és célcímet is. Tunnel módban pontosan ez történik. Az eredeti csomag teljes egészében, az eredeti forrás- és célcímekkel együtt enkapszulálásra kerül. Az új IPSEC csomag forrás és célcímei eltérhetnek az eredeti csomag forrás- és célcímeitől. Így az IPSEC csomag forrás- és célcíme olyan publikus címeket tartalmazhat, amikhez az interneten lehet útvonalat választani, a kicsomagolás után pedig újra elérhetőek lesznek azok a (belső privát) címek, ami alapján az útvonalválasztás a helyi/belső alhálózaton belül elvégezhető.

A tunnel mód bármikor használható a transzport mód helyett. Azonban fontos megjegyezni azt, hogy a tunnel mód további header-ek hozzáadását igényli, és így csökkenti az egy csomagban továbbítható hasznos adat (payload) maximális méretét, és fölösleges csomag fragmentációt okozat. Ezért ha biztosan nincs rá szükség, akkor ne használjunk tunnel módot.

IPSEC csomag formátumok: AH és ESP

Az AH csomag formátum arra alkalmas, hogy garantálja a küldő fél eredetiségét. (Azt, hogy valóban a másik fél küldte a csomagot.) Más szóval ezt a csomag integritásának nevezik. Az AH nem titkosítja az adatokat! Ez úgy működik, hogy az eredeti csomag egyes részeire kiszámítunk egy ellenőrző összeget. Az összeg kiszámításához felhasználjuk a választott hash algoritmust és az SA-ban található titkos kulcsot. Hogy a csomag mely részeiből számítjuk ki ezt az összeget, az függ

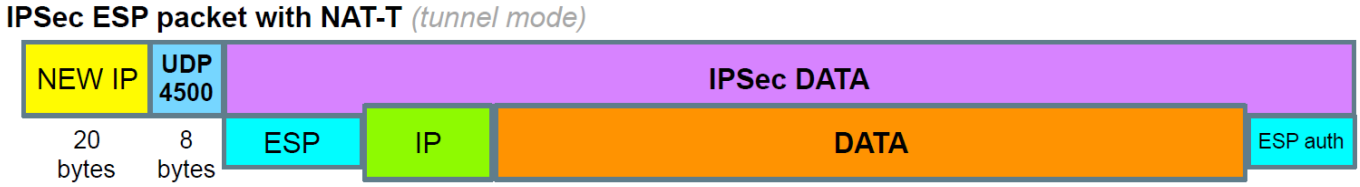
attól is, hogy transzport vagy alagút módot használunk.

- Transzport módnál az AH header-t az IP header után szúrjuk be. Az IP adat és IP header is részt vesz az ellenőrző összeg számításában. Azok az IP mezők amik megváltozhatnak a továbbítás során (pl. TTL) nullára vannak állítva az összeg kiszámítása előtt. Ez csökkenti a header-ek méretét, és növeli a hasznos payload méretét. Ugyanakkor ez azzal jár, hogy az IP header-ben levő címek nem módosíthatók.
- Alagút módnál a teljes IP csomagot enkapszuláljuk. Az eredeti IP csomag teljes egésze (ide értve az forrás és célcímet valamint a TTL mezőket is) részt vesz az ellenőrző összeg számításában. Ez garantálja az eredeti csomag forrás és célcímének integritását, de növeli a header-ek méretét, és csökkenti a hasznos payload méretét. Az IP header-ben levő címek módosíthatók.

Bővebb infó: https://wiki.mikrotik.com/wiki/Manual:IP/IPsec#Authentication_Header_.28AH.29

Az ESP csomag formátum garantálja a csomagok integritását, és titkosítja a bennük levő adatokat. Teljesen más mezőket használ mint az AH. Ezeket a mezőket három csoportba lehet osztani:

- **ESP Header** - ez a titkosított adat előtt van. A helye attól függ, hogy transzport vagy alagút módot használunk.
- **ESP Trailer** - Ez a titkosított adatok után következik. Ez tartalmaz egy padding-ot ami szükséges ahhoz, hogy a titkosítandó adat mérete a titkosító algoritmus blokk méretének egész számú többszöröse legyen.
- **ESP Authentication data** - Ez tartalmaz egy Integrity Check Value (ICV) értéket. Ez az AH protokollhoz hasonló módon van kiszámítva.



Bővebb infó:

https://wiki.mikrotik.com/wiki/Manual:IP/IPsec#Encapsulating_Security_Payload_.28ESP.29

Az AH használata akkor indokolt, ha az átvinni kívánt adatok nem titkosak (nyilvánosak) és csak annyi a célunk, hogy kívülálló/közbeékelődő támadók ne legyenek képesek más nevében adatokat küldeni. A mi példánkban a célok között szerepel az, hogy a két site közötti kommunikációt mások ne tudják lehallgatni. Emiatt ESP-t kell használunk.

Második fázis, azonosítás (autentikáció) módja

Itt pont ugyan azok igazak, mint az első fázisnál. (Az azonosítás módja az első és második fázisnál azonos.)

PFS (DH) csoport

Ennek megértéshez vissza kell térnünk az első fázishoz. Az első fázisban csak azokat a kezdeti kulcsokat határozzuk meg DH algoritmussal, amik a második fázis biztonságos lebonyolításához szükségesek. A második fázisban létrehozott kulcsok nem tartanak örökké. Ahogy azt korábban írtam, a második fázisban létrehozott SA-knak van egy soft és egy hard lifetime limit-je, és időnként meg kell őket újítani. A második fázisban létrehozott SA nem ugyanazokat a kulcsokat tartalmazza, mint amiket az első fázis használt. Minden második fázisban létrehozott SA saját, egyedi titkos kulcsokat tartalmaz, amik előre nem kitalálhatóak. Tehát valójában kétféle SA van: egy szülő SA ami az első fázishoz tartozik, és ehhez kapcsolódó további "gyermek" SA-k, amik a második fázishoz tartoznak. Az első fázis rendszerint egyszer játszódik le, ezért az ehhez tartozó kulcsot szokás *állandó kulcsnak* is nevezni. A második fázis sokszor lejátszódhat, mivel a kommunkikáció hosszú ideig eltarthat, és ezalatt többször szükség lehet a (korlátos életidővel rendelkező), második fázis által generált SA-k megújítására.

Ahogy korábban említettem, a DH algoritmus nagyon számításigényes. Ezért a második fázisban létrehozott SA-k titkos kulcsait alapesetben az első fázisban létrehozott SA kulcsaiból származtatják. Mivel az első fázisban DH algoritmussal létrehozott titkos kulcsok közvetlenül semminek a titkosítására nincsenek fölhasználva, ezért szinte semmi esély nincs arra, hogy ezt a kezdeti kulcsot bárki brute force módszerrel (a kommunikáció rögzítésével és annak próbálkozásos visszafejtésével) meghatározza. Ez a működés `pfs-group=none` beállításnak felel meg RouterOS-ben. Ennél a beállításnál a második fázis egyáltalán nem használ DH algoritmust. Emiatt gyors, kisebb CPU igényű. Mégis viszonylag biztonságos.

Előfordulhat azonban az, hogy az első fázisban meghatározott állandó kulcshoz valaki valamilyen más módon hozzáfér. Ha ez bekövetkezik, akkor az ebből származtott összes második fázisban használt kulcs meghatározása nagyon egyszerűvé válik. Mivel az első fázisban meghatározott kulcs nagyon hosszú ideig használatban lehet (nincs lejárata!), ezért az állandó kulcs feltörése után visszamenőleg megfejthetővé válik az összes kommunikáció, amit korábban a támdó rögzített. Ez különösen akkor veszélyes, ha az első fázis által meghatározott kulcsok megbízhatóan, leállás nélkül működő csomópontok között, határozatlan ideig kiépített alagút kiépítésére vannak használva. Ennek a problémának kivédésre szolgál a Perfect Forward Secrecy vagy röviden `PFS`. Ennek az a lényege, hogy a második fázishoz használt kulcsot nem az első fázisban meghatározott első kulcsból származtatjuk, hanem egy külön DH algoritmussal határozzuk meg. Ez csökkentheti az átviteli sebességet, és a válaszidőket is növelheti. Főleg akkor, ha a második fázis lejárati ideje rövid, és egyetlen eszközön párhuzamosan egyszerre sok VPN alagutat kell működtetni. Ugyanakkor ez a módszer garantálja azt, hogy a korábban rögzített kommunikáció ne legyen teljes egészében visszafejthető akkor, ha valaki megszerzi az első fázisban meghatározott állandó kulcsot. PFS group használata esetén minden IPSEC SA kulcsát egy külön DH algoritmus futtatással határozzuk meg, aminek semmi köze nincsen az IKE első fázisában használt állandó kulcshoz. Ha bármelyik kulcs feltörésre kerül (például brute force módszerrel), akkor csak a kommunikációnak azon része lesz hozzáférhető, aminek titkosításához a feltört/megtalált kulcsot használták.

Van itt erről egy jó kérdés: https://security.stackexchange.com/questions/196832/which-pfs-group-is-recommended-for-ipsec-configuration

Mi ebben a példában `modp2048` beállítást fogunk használni az első és a második fázishoz is. Ez egy viszonylag erőforrás igényes beállítás. A választást azzal indokoljuk, hogy itt egy site-to-site kapcsolatról van szó. Egyetlen alagutat építünk ki két alhálózat összekötésére, és csak ezt az egy alagutat kell működtetnünk. Ezek a peer-ek megbízhatóan, hosszú időn keresztül működnek. A két alhálózatban levő gépek ugyan ezt az egy alagutat használják megosztva. Így a felépítendő SA-t száma alacsony, viszont az egy SA-n keresztül nagyobb adatmennyiség van lebonyolítva. Ez indokolja a magasabb PFS DH csoport használatát.

Revision #2
Created 9 January 2021 11:46:23 by Gandalf
Updated 9 January 2021 12:16:54 by Gandalf