Virtuális Magánhálózatok (VPN)

BME - Távközlési és Médiainformatikai Tanszék
A mérést kidolgozta: Kersch Péter

Tartalomjegyzék

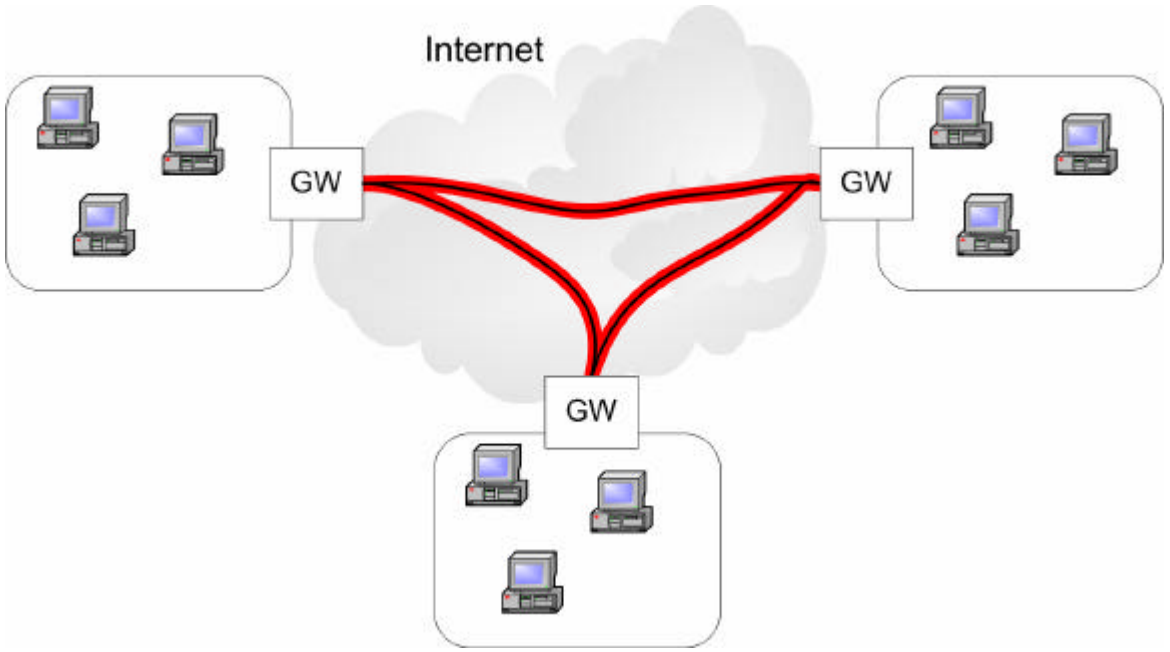
1. A VPN fogalma

A virtuális magánhálózatok (**V**irtual **P**rivate **N**etworks) a nyilvános távközlési infrastruktúrán keresztül biztonságos összeköttetést nyújtanak több földrajzilag elkülönülo hálózat között. Az ilyen Internetre épülo hálózat jóval gazdaságosabb megoldást kínál, mintha egy vállalat saját kommunikációs vonalakat építene ki székhelyei közt, vagy bérelt összeköttetéseket használna. A tranzit hálózat nyilvánossága miatt azonban alapveto követelmény minden VPN-nel szemben egy megbízható biztonsági modell alkalmazása. A VPN-nek védelmet kell biztosítani a hálózatok közti forgalom ellen indított passzív és aktív támadások ellen egyaránt (lehallgatás, az adatforgalom módosítása, megszemélyesítéses támadások, forgalom-elemzés stb..)

2. VPN technológiák

A napjainkban használatos, illetve kifejlesztés alatt álló VPN technikák igen sokfélék, és a kereskedelmi forgalomban is megszámlálhatatlan mennyiségu VPN megoldás kapható. Ezenkívül a VPN-ek esetén kulcsfontosságú biztonsági technikák a hálózati modell számos szintjén megvalósíthatóak. Az alkalmazási rétegben muködik pl. a PGP, szállítási réteg szintu az SSL, de a fizikai és adatkapcsolati rétegben is számos ilyen protokoll létezik (pl. WEP). A felsobb rétegbeli protokollok azonban csak alkalmazások egy csoportjának biztonsági problémáit oldják meg, az alsóbb rétegbeliek pedig csak egy adott hozzáférési közegen használhatók. Minden alkalmazásra kiterjedo, heterogén környezetben is használható megoldást biztosíthatunk viszont az IP alapú hálózati rétegben.

A hálózati réteg biztonsági protokolljai közül a legelterjedtebb és a legnagyobb jelentoséggel bír az IETF által szabványosított **IPsec** [1]. Az IPsec mellett viszonylag elterjedten használnak két másik VNP technológiát is, a PPTP-t [3] (Point to Point Tunneling Protocol), valamint a Cisco által kifejlesztett L2TP-t [2] (Layer 2 Tunnel Protocol), ezekkel azonban nem foglalkozunk részletesebben. A VPN-ek általános hálózati modellje az 1. ábraán látható:



1. ábra - VPN hálózati architektúra

Minden hálózat egy átjárón (GW) keresztül csatlakozik az Internetre. Miután az egyes átjárók hitelesítették egymást, valamennyien kiépítenek egymás közt egy-egy tikosított alagutat, és a VPN egyes hálózatai közti forgalom ezeken az alagutakon keresztül folyik. Ha az 1. hálózat egyik állomása csomagot szeretne küldeni a 2. hálózat egy állomásának, akkor a GW1 átjáró titkosítja a teljes csomagot (az IP fejlécet is beleértve), és ellátja egy új IP fejléccel, amelynek forrás IP címe a GW1, cél IP címe pedig a GW2 átjáró címe. A GW2 átjáró eltávolítja ezt a külso IP fejlécet, dekódolja és továbbítja az eredeti IP csomagot a címzettnek a 2. hálózatban. Ezáltal az Internet egy köztes csomópontja csak egy tikosított adatfolyamot lát két átjáró közt, még azt sem tudja megállapítani, hogy mely állomások kommunikálnak egymással. A megoldás másik elonye, hogy teljesen transzparens a VNP hálózat állomásai számára, a titkosítást és az alagutazást a VPN átjárók végzik. Fontos megemlíteni, hogy az egyes hálózatok állhatnak akár egyetlen mobil állomásból is, amely az Interneten keresztül szeretne biztonságos kapcsolaton keresztül csatlakozni a távoli vállalati hálózathoz. Ilyenkor ez az állomás egy személyben átjáró is.

3. IPsec

Az IPsec egy teljes biztonsági architektúrát definiál az Internet Protokoll szintjén. A szabványt az IPv4 és IPv6 protokollokhoz is kidolgozták; IPv4 esetén opcionális, míg IPv6 esetén kötelezo megvalósítása. Fontos kihangsúlyozni, hogy az IPsec nem egy VPN technológia, hanem egy általános biztonsági megoldás az IP alapú hálózatok számára; a VPN hálózatok létrehozása csupán egy az IPsec lehetséges alkalmazásai közül. Az IPsec a következo biztonsági szolgáltatásokat nyújtja:

- bizalmasság (titkosítás)
- küldo fél azonosítása (hitelesítés)
- adatcsomag integritás (sérülés, módosítás detektálása)
- korlátozott visszajátszás védelem*
- korlátozott adatfolyam bizalmasság (harmadik fél nem mindig tudja megállapítani, ki kivel kommunikál)

Az IPsec biztonsági architektúra a következo biztonsági protokollokból áll össze:

- ESP, Encapsulating Security Payload [4]
- AH, Authentication Header [5]
- IKE, Internet Key Exchange
- Kriptográfiai algoritmusok

3.1. Az ESP és AH protokollok

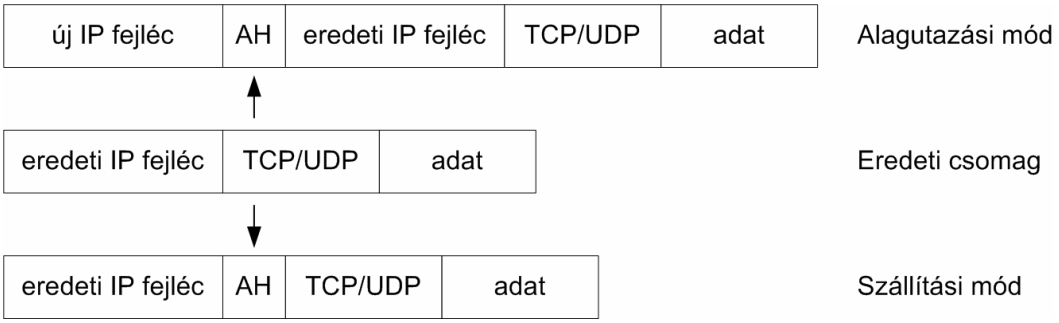
Az ESP és az AH kiegészíto fejléceket (*extension headers*) definiálnak az IP protokollhoz. Az AH fejléc adatcsomag integritás ellenorzést, a küldo fél hitelesítését és opcionálisan korlátozott visszajátszás védelmet biztosít. Az ESP fejléc két különbözo biztonsági szolgáltatás csoportot is nyújthat (vagy az egyiket, vagy a másikat, vagy egyszerre

* Mivel az IP egy kapcsolat nélküli (connectionless) protokoll, ezeket a feladatokat legfeljebb részben képes megoldani. A visszajátszás védelem és a sorrendezés biztosítása a felsobb rétegek feladata.

mindkettot): egyrészt titkosítást, és az adatfolyam korlátozott bizalmasságát, másrészt megvalósítja az AH fejlécnél bemutatott hitelesítési funkciókat is.

A hitelesítés a csomag bizonyos részeibol számított ICV (*Integrity Check Value*) alapján történik. Ennek számítása történhet szimmetrikus kódoláson alapuló MAC (*Message Authentication Code*) algoritmusok vagy egy irányú hash-függvények és aszimmetrikus aláírások kombinációjára alapozva. Az ESP titkosítása szimmetrikus kódoló algoritmusokkal történik. A szabvány nem határozza meg, hogy a hitelesítés és a tikosítás konkrétan milyen algoritmusok szerint történjen, csak azt hogy melyek azok az algoritmusok, amelyeket minden implementációnak minimálisan ismernie kell. A konkrét algoritmus, és annak paramétereinek kiválasztása a biztonsági összerendelés (Security Association - SA) kiépítésekor történik meg, és ennek azonosítója minden AH illetve ESP fejlécben megtalálható az SPI (*Security Parameter Index*) mezoben.

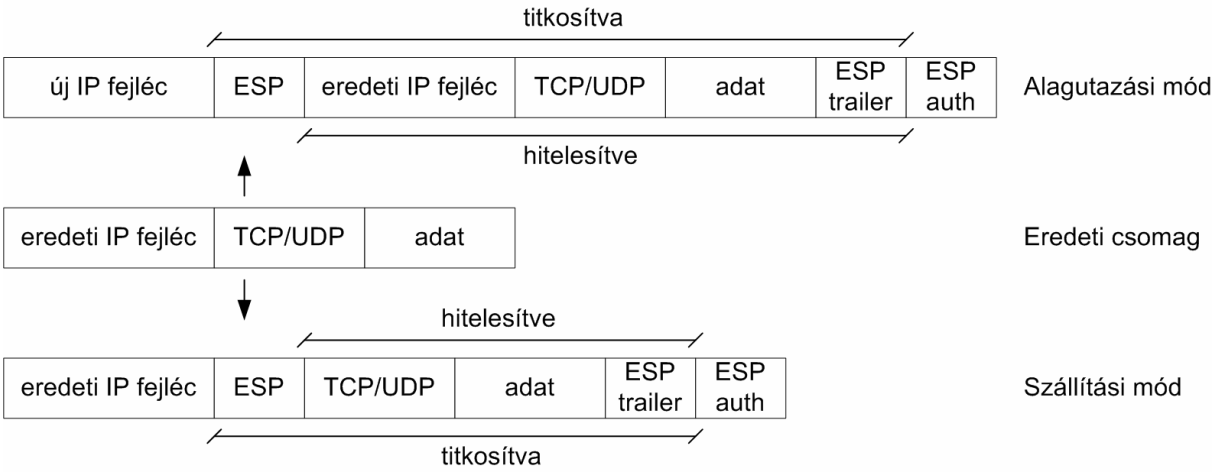
Mindkét protokollnak két felhasználási módja van, szállítási mód (*transport mode*), illetve alagutazási mód (*tunnel mode*). Szállítási módban az AH fejléc az eredeti IP csomag IP fejléce, és a felsobb rétegbeli protokollok közé ékelodik, míg alagutazás módban az eredeti IP csomagot IP fejléccel együtt egy új IP csomagba ágyazzuk.(lásd 2. ábra)



2. ábra - szállítási mód és alagutazási mód AH fejlécnél[*]

ESP esetén ugyanez történik viszont az eredeti adatcsomag végére is kerül egy ESP trailer mezo, illetve az hitelesítési szolgáltatások kiválasztása esetén egy hitelesíto mezo is. Fontos különbség továbbá az AH és az ESP protokollok közt, hogy az ESP protokoll titkosítása illetve a hitelesítése szállítási módban nem terjed ki az eredeti IP fejlécre. (lásd 3. ábra) Az AH kiegészíto fejléc ezzel szemben a teljes IP csomagot védi, beleértve az IP fejlécet is, kivéve azt a néhány mezot, amely a csomagtovábbítás közben is megváltozhat (pl. TTL). Alagutazás módban mindkét protokoll a teljes csomagot védi, az új IP fejlécre viszont ugyanazok a különbségek érvényesek, mint a szállítási módban az eredeti IP fejléc esetén.

[*] Természetesen TCP illetve UDP helyett bármely más IP felett elhelyezkedo protokoll fejlécét is behelyettesíthetnénk



3. ábra - szállítási mód és alagutazási mód ESP fejlécnél

Az alagutazási módot biztonsági átjárók és közönséges IPsec állomások közt is lehet használni, szállítási módot viszont csak két IPsec állomás közt. VPN hálózatok kiépítéshez ezért mindig alagutazás módot használunk.

3.2. IKE

Ahogy azt az elozo fejezetben említettük, ahhoz, hogy az ESP és AH protokollok titkosítási illetve hitelesítési funkciókat láthassanak el, eloször létre kell hoznunk egy biztonsági összerendelést (SA) a két fél között. A két félnek meg kell állapodni a használandó kriptográfiai algoritmusokról, azoknak kulcsáról illetve egyéb paramétereirol. Ez megvalósítható manuális módszerekkel is, azonban nehézkes, rugalmatlan, és rosszul skálázható megoldást eredményezne. (Ha pl. egy 50 hálózatból álló VPN-t szeretnénk létrehozni, amelyben minden egyes hálózat közt biztonságos kapcsolatot építünk ki, akkor ez 50x49/2=1225 kulcs folyamatos karbantartását tenné szükségessé) Ennek kiküszöbölésére az IKE protokoll [6] egy általános hitelesített kulcscsere és paraméteregyeztetés szolgáltatást biztosít az IPsec-en belül. Az automatikus kulcskiosztás további elonye, hogy periodikusan új kulcsokat lehet kiosztani, anélkül, hogy a tikosított kapcsolat megszakadna, nagyban növelve ezáltal a rendszer biztonságát. Az IKE egy hibrid protokoll, felhasználja az ISAKMP keretrendszert, az Oakley kulcscsere mechanizmust és a SKEME kulcscsere módszert. Egy IKE biztonsági összerendelés létrehozása 2 fázisból áll. Elso fázisban a két fél hitelesíti egymást[*], és létrehoz egy biztonságos kommunikációs csatornát egymás közt a SA további paramétereinek egyeztetéséhez. A második fázisban történik meg az AH és ESP protokollok kulcsainak és paramétereinek egyeztetése.

A kulcscsere folyamat során elengedhetetlen a résztvevo felek hitelesítése, hiszen ennek hiányában lehetové válna a *man-in-the-middle* támadás a protokoll ellen. Ez a hitelesítés elore kiosztott közös titok alapján, vagy nyilvános kulcsú titkosítást használva történik. Az elobbi módszer megint csak nehézkes, hiszen ez a manuális kulcskiosztáshoz hasonlóan skálázhatósági problémákba ütközik. Nyilvános kulcsú technikák esetén közös titok párok

[*] A két fél hitelesítése az IKE protokollban nem tévesztendo össze a küldo fél hitelesítésével csomagok esetén az AH illetve ESP protokollban!

helyett elegendo csupán a több n hálózat nyilvános kulcsának ismerete. Visszatérve az elobbi 50 hálózatból álló példánkra, automatikus kulcscsere esetén a hitelesítéshez nyilvános kulcsú technikát használva már csak 50 nyilvános kulcsot kell nyilvántartanunk, titkos kulcsból pedig minden biztonsági átjárón csak egyetlen egyet! A nyilvános kulcsú technikák számításigénye ugyan jóval nagyobb, mint a közös titkos kulcsot használó szimmetrikus kulcsú technikáké, azonban ezt a hitelesítést nem kell elvégezni minden csomag vételekor vagy küldésekor, csak a kapcsolat kiépítésekor, ezért ez nem jelent hátrányt.

A nyilvános kulcsok elosztása megint csak történhet manuálisan, de megoldható X.509 tanúsítványok és PKI (Public Key Infrastructure) segítségével vagy SecureDNS felhasználásával is. PKI felhasználása esetén drasztikusan tovább javul a kulcsmenedzsment feladatok skálázhatósága. Ilyenkor ugyanis elegendo, a saját nyilvános kulcs pár valamint a tanúsítványt kibocsátó CA nyilvános kulcsát eltárolni minden átjárón. A CA nyilvános kulcsa alapján ugyanis meg lehet állapítani a másik fél által átküldött tanúsítványról, hogy azt valóban a CA írta-e alá, és ezáltal meg lehet gyozodni a tanúsítványban átküldött nyilvános kulcs hitelességérol.

Az IKE protokoll UDP csomagokat (500-as port) használ mindkét fázis üzeneteinek átvitelére.

4. IPsec implementációk

A rendelkezésre álló IPSec implementációk skálája nagyon széles: gyakorlatilag minden platfomra létezik kereskedelmi vagy szabadon hozzáférheto implementáció és a hardver megoldások terén is nagyon nagy a választék. Hely hiányában itt csupán egy nyílt forráskódú szoftveres implementációról (Openswan), valamint egy hardveres megoldásról (Linksys BEFVP41) lesz bovebben szó.

4.1. Nyílt forráskódú implementációk

Napjainkban két széles körben elterjed nyílt forráskódú IPSec implementáció hozzáférheto: a KAME [8] stack BSD alapú rendszerekhez, és a FreeS/WAN [9] stack Linux alapú rendszerekhez.

A KAME projektet hat japán vállalat alapította azzal a céllal, hogy nyílt forráskódú IPv6 és IPSec implementációt hozzon létre a különbözo BSD variánsokhoz.

A FreeS/WAN projektnek két fo célkituzése volt: egyrészt olyan Linux alapú IPSec implementáció kifejlesztése volt, amely nem esik az USA kriptográfiai szoftverekre vonatkozó paranoiás export szabályozásai (azaz USA állampolgárok nem vehettek részt a fejlesztésben). A másik fo cél az ún. „opportunistic encryption - OE” megvalósítása és szélesköru elterjedése lett volna. Az OE lényege, hogy minden elozetes konfiguráció nélkül biztonságos kommunikációt biztosítson két tetszoleges állomás közt. Ennek megvalósításához az OE a DNS rendszert használná nyilvános kulcsok tárolására és lekérdezésére. Miután ennek a (több szempontból is megkérdojelezheto) megoldásnak a szélesköru elterjesztése nem sikerült, a projekt 2004 márciusában hivatalosan is befejezodött. Azonban a FreeS/WAN projekt egy fejlesztoi csoportja – akik egyre inkább megelégelték a projekt politikai vonatkozású kérdéseit – Openswan [10] néven tovább folytatják a FreeS/WAN kód fejlesztését. A továbbiakban ennek a Linux alapú implementációnak a részletesebb bemutatása következik

4.2. Openswan

A Openswan két nagy alrendszerbol tevodik össze: egy kernel szinten futó alrendszerbol (KLIPS), amely az AH, illetve ESP protokollokat kezeli, illetve egy felhasználó szintu programból (Pluto daemon), amely az IKE kulcscsere protokollt implementálja. Mivel a 2.6-os Linux kernel már saját kriptográfiai alrendszerrel rendelkezik, és támogatja az AH és ESP protokollokat is, ezért 2.6-os kernel verziótól a KLIPS alrendszer helyett a native Linux IPSec alrendszert is lehet használni. A kernel szintu IPSec alrendszer feladata tehát az IP csomagok titkosítása, hitelesítése, az ESP és AH fejrészek létrehozása az elküldött csomagok esetén és ezeknek a fejrészeknek az értelmezése a bejövo csomagok esetén. A Pluto démon felelossége a biztonsági hozzárendelések létrehozása más biztonsági átjárókkal, vagy állomásokkal, illetve ezeknek az elozetes hitelesítése. A biztonsági hozzárendelések létrehozása után a Pluto démon a megfelelo adatokat (kriptográfiai algoritmus, paraméterek, kulcsok, stb) átadja a kernel szintu IPSec alrendszernek. A pluto démon beállításait a /etc/ipsec.conf fájlban lehet megadni (lásd *man ipsec.conf*), az IPsec muködését pedig az *ipsec* shellscripttel lehet vezérelni (lásd *man ipsec*)

4.2.1. ipsec.conf

Az Openswan IPSec implementáció beállításait néhány kivételtol eltekintve a /etc/ipsec.conf fájl tartalmazza. Ez egy szöveges konfigurációs fájl, amely ún. szakaszokból (*sections*) épül fel. Minden szakasz elso sora

type name

formátumú, ahol a *type* a szakasz típusát határozza meg a *name* pedig egy tetszoleges egyedi azonosító, amellyel a szakaszra hivatkozni lehet. Az elso üres sorig minden további sor ehhez a szakaszhoz tartozik. Ezeknek a soroknak a formátuma a következo:

parameter=value

Minden ilyen sort szóközzel vagy tabulátorral kell kezdeni.

A legfontosabb szakasztípus a *conn*, amellyel a kiépítendo IPsec kapcsolatokat definiálhatjuk. A kapcsolat neve tetszoleges lehet, ezzel a névvel lehet majd hivatkozni az adott összeköttetésre az *ipsec* parancs használata során. A konfigurációs fájlok felesleges szerkesztését elkerülendo a kapcsolat két végpontjára nem úgy hivatkozunk, mint helyi vagy távoli végpont, hanem mint jobb vagy bal. Hogy melyik végpont a jobb illetve a bal, az tetszoleges, ezt az IPsec a szakasz konkrét paraméter értékeibol deríti ki. A *right* kezdetu paraméterek a kapcsolat jobb, a *left* kezdetuek pedig a bal végpontjára vonatkoznak, és természetesen minden *left* kezdetu paraméternek van *right* kezdetu megfeleloje (ezért a továbbiakban mindig csak a left kezdetueket mutatjuk be). Így ugyanazt a kapcsolatleíró szakaszt használhatjuk mindkét végpont esetén. Egy kapcsolat legfontosabb paraméterei a következok:

type: Az IPsec kapcsolat típusa, lehetséges értékek a *tunnel* v. *transport*, attól függoen, hogy IPsec alagutat vagy szállítási módú kapcsolatot szeretnénk kiépíteni. (alapértelmezett az alagút)

left: A bal oldali végpont nyilvános hálózatra csatlakozó hálózati interfészének IP címe. Lehetséges érték még a *%defaultroute* is, ebben az esetben ez az alapértelmezett átjáró felé nézo interfész IP címét jelenti, *%any* értéket megadva pedig bármely címet. (kötelezo paraméter)

leftsubnet: A bal oldali végpont mögött elhelyezkedo magánhálózat alhálózati címe hálózatcím/netmask formátumban. Ha nincs megadva ilyen paraméter, akkor az egyenértéku a

left/32 jelöléssel, azaz azt jelenti, hogy a kapcsolat kizárólag a végpontot érinti, nincs a végpont mögött alhálózat.

leftnexthop: Annak a next-hop routernek az IP címe, amelyen keresztül a végpont a nyilvános hálózatra kapcsolódik. Lehetséges értékek még: *%direct* (alapértelmezett érték), ilyenkor ez nem más mint a *right* paraméter értéke; illetve megadhatunk még *%defaultroute* értéket is. Csak helyi jelentosége van (a másik végpontnak nincs szüksége erre az információra)

leftupdown: Annak a szkriptnek az elérési útvonala, amelyet a kapcsolat kiépítésekor vagy lebontásakor kell meghívni az útvonalválasztó tábla, illetve a tuzfal beállítások módosításához (általában megfelel az alapértelmezett érték, és szintén csak helyi jelentosége van)

A többi paraméter értéke attól függ, hogy manuális, vagy automatikus kulcskiosztást használunk. Mivel a mérés során kizárólag automatikus kulcskiosztást használunk (a manuális kulcskiosztás rendkívül rugalmatlan), a továbbiakban ezeket a paramétereket ismertetem csak:

auto: Annak a muveletnek a neve, amit az IPsec indításakor automatikusan végrehajt a kapcsolat esetén. Lehetséges értékek: *add, route, start, manual, ignore* (lásd még *ipsec auto* parancs)

auth: Lehetséges értéke AH vagy ESP, attól függoen, hogy a hitelesítést az ESP fejléc, vagy egy külön AH fejléc végzi

authby: Meghatározza, hogy milyen módszerrel hitelesíti egymást a két végpont. Lehetséges értékek: *secret* (osztott tikos kulcs) vagy *rsasig* (digitális aláírás RSA-val) Az alapértelmezett ez utóbbi. A hitelesítés X.509 vagy OpenPGP tanúsítványok használata esetén is a tanúsítvány RSA kulcsa alapján történik, ezért ilyenkor is az *rsasig* értéket kell használnunk.

leftid: A végpont azonosítója a hitelesítési folyamatban. Ez egy IP cím vagy egy teljes domain név egy '@' karakterrel az elején. Opcionális paraméter. Ha nincs megadva, akkor a végpont azonosítása a left paraméter alapján történik. X.509 alapú hitelesítés esetén speciális formátumot kell használni, amely a tanúsítvány azonosító mezoit tartalmazza vesszovel elválasztva, pl.: „C=HU, L=BUTE, O=TMIT, CN=gateway1.tmit.bme.hu, E=vpn@gateway1.tmit.bme.hu"

leftrsasigkey: A bal oldali végpont nyilvános RSA kulcsa RSA hitelesítés esetén. Ha az RSA kulcsot X.509 tanúsítvány tartalmazza, akkor ezt a *%cert* érték megadásával jelezhetjük.

leftcert: A hitelesítéshez használt tanúsítvány elérési útvonala.

compress: Lehetséges értékek: *yes, no* (alapértelmezett) Engedélyezi, illetve kikapcsolja az IPComp [*] tömörítést. [7]

Számos paraméter használható még a kulcsok élettartamának, és az újrakiosztásának szabályozására, ezeket most nem mutatom be részletesebben.

[*] A titkosított csomagok esetén az adatkapcsolati rétegbeli tömörítés hatékonysága általában nagyon rossz, még akkor is, ha egyébként az eredeti csomag egyébként jól tömörítheto. Az IPComp lehetoséget biztosít a csomag tömörítésére még a titkosítás elvégzése elott.

Megjegyzés: Az egyes kapcsolatok paramétereinek konfigurációs fájlban történo megadása nem teszi lehetové ezeknek a paramétereknek a dinamikus változtatását. Ezt az *ipsec whack* paranccsal lehet megoldani. Mivel azonban a VPN-ek többségében ilyen dinamikus változtatásra nincsen szükség, ezért ezzel részletesebben nem foglalkozunk.

4.2.2. ipsec.secrets

A /etc/ipsec.secrets fájl egy index – kulcs párokból álló táblázat (az index és a kulcs kettosponttal elválasztva), amely különbözo titkos kulcsokat tartalmaz. Ez a kulcs lehet osztott titkos kulcs PSK (Pre-Shared Key) azaz osztott kulcsos hitelesítés esetén, RSA titkos kulcs RSA alapú hitelesítéshez esetén, valamint titkos kulcsot védo jelszó X.509 hitelesítés esetén. Az index vagy IP címekbol vagy DNS nevekbol álló lista, amely azt azonosítja, hogy a kulcs mely kapcsolatra vonatkozik, de akár teljesen el is hagyható (pl általános érvényu, minden kapcsolatra vonatkozó RSA titkos kulcs esetén).

4.2.3. IPsec parancsok

Az *ipsec* shellszkript egy egységes interfészt biztosít a FreeS/WAN ipsec implementációt vezérlo parancsok meghívására, és egyúttal kiküszöböli azokat a névtérbeli ütközéseket, amelyeket sok különálló ipsec parancs okozhatna más programokkal. A parancs szintaxisa:

ipsec *parancs [paraméterlista]*

A legfontosabb *ipsec* parancsok a következok:

ipsec auto *muvelet_neve kapcsolat_neve:* Az automatikus kulcskiosztású kapcsolatokat indítja el illetve állítja le a paraméterként megadott muvelet szerint. A muvelet pararméter lehetséges értékei: *--add, --delete, --replace, --up, --down, --route, --unroute.* A kapcsolat neve az *ipsec.conf* fájl megfelelo *conn* szakaszának neve

ipsec barf: Megjelenít minden olyan *ipsec*-kel kapcsolatos információt, amely hibakereséshez hasznos lehet.

ipsec setup *parancs:* A teljes ipsec alrendszer vezérelhetjük vele. A megadott parancstól függoen elindítja (*start*), leállítja (*stop*), újraindítja (*restart*) a KLIPS *kernel* alrendszert és a *pluto* kulcskiosztó démont.

ipsec showhostkey [**--key**] [**--left**] [**--right**] [**--txt** gateway] [**--dhclient**] [**--file** secretfile] [**--id** identity]: A paraméterek által meghatározott formátumban megjeleníti az *ipsec.secrets* fájlban tárolt kulcsokat.

ipsec newhostkey parancs: létrehoz és a standard kimenetre kiír egy új RSA kulcspárt a */etc/ipsec.secrets* fájl által eloírt formátumban.

Az összes lehetséges ipsec parancsot az ***ipsec –help*** paranccsal jeleníthetjük meg

4.3. Hardver megoldások

A számtalan hardver megoldás közül csak a mérés során felhasznált Linksys BEFVP41 VPN routerrol lesz bovebben szó. A Linksys BEFVP41 egy kisvállalati illetve házi felhasználásra szánt IPSec alapú VPN router, amely támogatja a DES és 3DES tikosításokat és az IKE kulcscsere protokollt, valamint beépített IPSec társprocesszorral rendelkezik.

A BEFVP41 konfigurálása a modern broadband routerek többségéhez hasonlóan egyszeru és könnyen kezelheto webes felületen keresztül történik. A router üzembe helyezésérol és konfigurálásáról bovebben [11]-ben lehet olvasni. A BEFVP41 – Openswan együttmuködésrol pedig lásd [12].

5. PKI

Bár a nyilvános kulcsú infrastruktúráknak szigorúan véve semmi köze nincsen az IPSec-hez, azonban az elozoekbol kitunt, hogy a kulcskiosztás és kulcsmenedzsment kiemelkedoen fontos szerepet játszik a VPN-ek kiépítése során, ezért fontos röviden szót ejteni a PKI-ról is.

PKI azaz nyilvános kulcsú infrastruktúra alatt egy olyan rendszert értünk, amely digitális tanúsítványokból (Digital Certificates), hitelesítés szolgáltatókból (Certificate Authorities - CA) és egyéb regisztrációs hatóságokból áll, amelyek feladata, hogy ellenorizzék a résztvevo felek hitelességét biztonságos digitális kommunikációs során. A nyilvános kulcsú infrastruktúra végletekig leegyszerusített muködési elve a következo: A kommunikáló felek nyilvános kulcsát és azonosításra szolgáló adatait egy megbízható harmadik fél (CA) egy tanúsítványba foglalja, és ezt a tanúsítványt saját tikos kulcsával aláírja. A tanúsítvány és a tanúsítványt kiállító CA nyilvános kulcsának ismeretében a másik fél meggyozodhet arról, hogy a tanúsítványban átadott nyilvános kulcs valóban a kommunikáló partner nyilvános kulcsa-e, a nyilvános kulcs ismeretében pedig már ellenorizheti, hogy a másik fél valóban rendelkezik-e a nyilvános kulcshoz tartozó titkos kulccsal. A problémát a „megbízható harmadik fél" okozza. Egységes adminisztráció alá eso hálózatban (pl. vállalati hálózatban) ez a probléma viszonylag könnyen megoldható – pl. nagy hálózatok esetén hierarchikus CA rendszer kiépítésével – azonban külön adminisztráció alá eso hálózatok esetén vagy világméretetekben már komoly nehézségeket okoz.

Részben ennek köszönheto, hogy bár nyilvános kulcsú infrastruktúrát illetve annak részeit elterjedten használnak számos területen (gondoljunk csak a https oldalak X.509-es tanúsítványira, vagy a böngészokbe beégetett root CA tanúsítványokra vagy a PGP-re), máig sincs egységes és általánosan elfogadott szabvány ezen a területen. A legelterjedtebb az X.509 alapú PKI [14], azonban ez a rendszer messze nem tökéletes. Érdekes olvasmány ezzel kapcsolatban [15].

5.1. Egyszeru X.509 PKI infrastruktúra kiépítése OpenSSL-el

Az OpenSSL [16] egy nyílt forráskódú program és eljáráskönyvtár, amely megvalósítja az SSL (Secure Sockets Layer) és TSL (Transport Layer Security) protokollokat. Az OpenSSL változatos alkalmazási lehetoségei közül a továbbiakban csupán hitelesítés szolgáltató (CA) létrehozása valamint X.509-es tanúsítványok generálása és aláírása lesz csak részletesebben bemutatva. Ehhez az *openssl* parancsra és *CA.sh* scriptre lesz szükség (ez utóbbi Debian Linux alatt a */usr/lib/ssl/misc/* könyvtárban található – az elérési út disztribúció függo). Megjegyzés: a *CA.sh* script szintén openssl parancsokat használ, azaz kis többlet munkával minden tanúsítványokkal kapcsolatos muveletet el lehet végezni a script nélkül is. Az *openssl* parancsokat kilistázhatjuk az *openssl –help* paranccsal, az egyes parancsok szintaktikáját pedig szintén hasonló módon lehet lekérdezni: *openssl parancs_neve --help*

Új CA létrehozásához a

CA.sh –newca

parancsot kell kiadni. A script bekér egy jelszót a CA titkos kulcsának kódolásához (Ez a *PEM pass phrase*, amit minden alkalommal meg kell adni, amikor a CA használja titkos kulcsát – pl. tanúsítvány aláírásakor), valamint bekéri azokat az azonosításhoz szükséges adatokat, amik a bekerülnek a CA saját tanúsítványába. A script létrehozza a CA saját nyilvános és titkos kulcsát (*demoCA/private/cakey.pem*), valamint saját maga által aláírt X.509 tanúsítványát (*demoCA/cacert.pem*).

Nyilvános kulcsok, titkos kulcsok és tanúsítványok kódolására három különbozo fájl formátum használatos: PEM (OpenSSL és általában Unix világban), DER (java + böngészok),

PKCS12 (Windows világ). A DER fájl formátum ASN.1 DER formátum Base64 kódolással. A PEM gyakorlatilag ugyanaz, mint a DER formátum + egy ascii fejléc. Ezért egy szöveges PEM fájl elso ránézésre nem sok mindent árul el magáról. Az

openssl x509 –text –in demoCA/cacert.pem

paranccsal megjeleníthetjük emberi szem számára is olvasható formátumban a CA tanúsítványának tartalmát. Természetesen nem csak txt formátumba, de a fent említett bármelyik más formátumra is konvertálhatunk az *openssl* parancs felhasználásával.

Tanúsítványok létrehozása ezután a következo lépésekbol áll

- RSA kulcspár generálása
- Tanúsítvány kérelem létrehozása, amely tartalmazza a generált nyilvános kulcsot, és a tanúsítványban szereplo egyéb adatokat
- A tanúsítvány kérelem elküldése a CA-nak
- A CA ellenorzi a megadott adatokat, és aláírja a tanúsítványt
- A CA visszaküldi az aláírt tanúsítványt a kérvényezonek

RSA kulcs generálása az *openssl genrsa* paranccsal lehetséges. Paraméterként opcionálisan megadható a tikos kulcs rejtjelezése (DES, 3DES vagy AES titkosítással), a generált PEM fájl neve, illetve esetleges véletlen szám forrás. Az *openssl rsa* paranccsal pedig megjeleníthetjük a létrehozott kulcs-pár egyes paramétereit.

Tanúsítvány kérelmet a következoképpen hozhatunk létre:

openssl req –new –key key_file_name –out req_file_name

A létrehozott tanúsítványt ember által olvasható formátumban megjeleníthetjük az

openssl req –text –in req_file_name

paranccsal. Megjegyzés: az elobbi két lépés a *CA.sh* script felhasználásával egy lépésben is elvégezheto:

CA.sh –newreq

A tanúsítvány kérelem aláírása szintén elvégezheto openssl parancsokkal, és a *CA.sh* scripttel is:

CA.sh –sign

openssl ca –in certificate_request_to_be_signed –out certificate

5.2. X.509 tanúsítványok használata Openswan alatt

Az Openswan gatewayeknek a következo fájlokra van szüksége X.509 alapú tanúsítványok használatához:

- A hitelesítés szolgáltató(k) saját tanúsítványa (*/etc/ipsec.d/cacerts/*)
- A gateway saját tanúsítványa (*/etc/ipsec.d/certs/*) - elvileg lehet több is, de egy boven elég)
- A gateway titkos kulcsa (*/etc/ipsec.d/private/*). Ha a kulcs jelszóval tiktosítva van, akkor a jelszót a /etc/ipsec.secrets fájlban lehet megadni (: RSA kulcsfájl ”jelszó” formátumban)

- és végül egy üres CRL azaz a visszavont tanúsítványok listája (*/etc/ipsec.d/crls*) – amelyet az *openssl ca –gencrl –out filename* paranccsal lehet létrehozni

Azaz az RSA kulcs-pár megoldással szemben itt nem kell eltárolni a többi gateway tanúsítványát. Ezek a hitelesítési folyamat során lesznek lekérdezve, és a CA tanúsítványát felhasználva meg lehet állapítani, hogy azokat valóban a CA írta-e alá (azaz hitelesek-e).

6. Irodalomjegyzék

[1] S. Kent, R. Atkinson: Security Architecture for the Internet Protocol – rfc2401

[2] Layer 2 Tunnel Protocol http://www.cisco.com/univercd/cc/td/doc/product/software/ios120/120newft/120t/120t1/l2tpt.htm

[3] Kory Hamzeh, Gurdeep Singh Pall: Point to Point Tunneling Protocol – Internet Draft

[4] S. Kent, R. Atkinson: IP Encapsulating Security Payload (ESP) – rfc2406

[5] S. Kent, R. Atkinson: IP Authentication Header – rfc2402

[6] D. Harkins, D. Carell: The Internet Key Exchange (IKE) – rfc2409

[7] A. Shacham, R. Monsour, R. Pereia, M. Thomas: IP Payload Compression Protocol (IPComp) – rfc2393

[8] A KAME projekt honlapja: http://www.kame.net

[9] A FreeS/WAN IPSec implementáció honlapja: http://www.freeswan.org/

[10] Az *Openswan* implementáció honlapja: http://www.openswan.org

[11] Felhasználói útmutató a Linksys BFEVP41 VPN router üzembe helyezéséhez és beállításához: ftp://ftp.linksys.com/pdf/befvp41ug.pdf

[12] Útmutató Linksys BFEVP41 VPN router és Openswan együttmuködéséhez: http://www.freeswan.ca/docs/BEFVP41/

[13] Útmutató X.509 tanúsítványok használatához Openswan alatt: http://www.natecarlson.com/linux/ipsec-x509.php

[14] X.509 alapú PKI: http://www.ietf.org/html.charters/pkix-charter.html

[15] Amit a PKI-val kapcsolatos hátulütokrol érdemes tudni: http://www.cs.auckland.ac.nz/~pgut001/pubs/pkitutorial.pdf

[16] OpenSSL projekt honlapja: http://www.openssl.org