3. RETELE VPN – PRINCIPII SI CONFIGURARE

3.1. Concepte VPN

3.1.1. Ce este si cum functioneaza un VPN

Un VPN reprezinta o modalitate de a realiza un canal de comunicatie privat peste o infrastructura publica sau pasibila de interceptare/modificare a informatiei. Acest lucru presupune doua aspecte:

- tunelarea informatiei – incapsularea ei de asa natura incat capetele de VPN sa nu perceapa existenta infrastructurii WAN care le interconecteaza. In acest fel, statii aflate in doua retele situate in locatii geografice diferite pot comunica ca si cum s-ar afla conectate la acelasi router sau la acelasi switch
- securitatea informatiei (confidentialitate, integritate, autenticitate) – este optionala dar in general de dorit (pentru a justifica P-ul din VPN). Acest aspect este independent de primul: putem realiza o tunelare a informatiei prin internet (de exemplu) fara a o cripta, insa informatia va fi susceptibila de capturare/falsificare.

Un VPN este o alternativa la o linie de comunicatie dedicata (foarte scumpa).

3.1.2. Tipuri de VPN

Clasificarea VPN-urilor se poate alcatui din mai multe puncte de vedere:

- din punct de vedere al scenariului de utilizare, distingem VPN-uri:
  - **site-to-site** – sunt VPN-uri care isi propun sa conecteze doua retele printr-un tunel criptat. Fiecare dintre cele doua retele contine mai multe statii. Exemplul tipic este cel al unei companii cu doua sau mai multe filiale
  - **remote-access** – VPN-uri care isi propun sa asigure accesul unei statii la o retea. Cazul tipic este cel al angajatilor mobili, aflati in afara retelei companiei dar care au nevoie de a o accesa de la distanta, de acasa sau din alte retele
- din punct de vedere al plasarii codului VPN in cadrul sistemului de operare, avem VPN-uri:
  - **kernel-based** – operatiile de incapsulare si criptare sunt realizate de codul de retea din kernel. Implementarile uzuale sunt bazate pe IPSec. Caracteristici:
    - dezavantaje: protocol complex si mai greu de inteles si implementat; necesita modificare de kernel
    - avantaje: IPSec este un standard public, permitand astfel interconectarea intre diverse echipamente de retea (statii, routere ale diferitilor producatori etc)
  - **user-space** – operatiile de incapsulare/criptare sunt realizate de o aplicatie si transmise celuilalt capat prin intermediul unei interfete virtuale de tip tunel. Exemplu: VPN-urile SSL. Caracteristici:
    - avantaje: flexibilitate; usurinta de configurare; nu necesita modificari asupra kernelului
    - dezavantaje: nu pot fi implementate pe echipamente hardware precum routere sau switch-uri
- din punct de vedere al modului de adresare, impartim VPN-urile in:
  - **routed** – adresele aflate de cele doua parti ale tunelului formeaza subnet-uri distincte (domenii de broadcast diferite). Este solutia cea mai flexibila si scalabila
  - **bridged** – retelele aflate de o parte si de alta a tunelului fac parte din aceeasi retea (impart acelasi spatiu de adrese) iar broadcast-urile circula prin tunel. Este o solutie nu foarte scalabila, dar utila atunci cand retelele interconectate folosesc protocoale ce se bazeaza pe broadcast (ex: vechile protocoale de file sharing din Windows)

3.2. Prezentare OpenVPN

3.2.1. Caracteristici generale si principii de functionare

OpenVPN este un soft open source care implementeaza un SSL VPN; el ruleaza in user-space si poate fi folosit pentru implementarea tuturor scenariilor VPN enumerate mai sus (site to site si remote access, atat in mod routed cat si in mod bridged).

OpenVPN foloseste ca protocol de transport TCP sau UDP, portul fiind la alegerea administratorului. Pachetele transmise de capete vor fi astfel incapsulate in segmente TCP/datagrame UDP; putem ajunge astfel sa avem IP incapsulat in TCP

sau chiar TCP incapsulat in TCP. Acest ultim scenariu este in general de evitat, deoarece efortul simultan al ambelor TCP-uri de control al fluxului poate duce la efecte nedorite (performante mult degradate). In general este preferat UDP, deoarece este un protocol best effort si reproduce mai bine caracteristicile IP-ului, peste care este TCP-ul gandit sa functioneze.

Pentru a securiza canalul de comunicatie intre cele doua capete, OpenVPN poate folosi pana la 4 chei:

- *o cheie de criptare a informatiei pentru capatul 1* - folosita de catre capatul 1 atunci cand transmite informatie catre capatul 2, si utilizata de capatul 2 pentru a decripta informatia primita
- *o cheie de criptare a informatiei pentru capatul 2* – folosita de capatul 2 pentru a cripta informatia transmisa capatului 1, si de asemenea de catre capatul 1 pentru a decripta aceasta informatie
- *o cheie HMAC pentru capatul 1* – folosita de capatul 1 pentru a genera informatia de control atasata mesajelor transmise catre capatul 2, si utilizata apoi de capatul 2 in scopul verificarii integritatii mesajului receptionat
- *o cheie HMAC pentru capatul 2* – utilizata de capatul 2 la generarea informatiei de control a mesajelor transmise catre capatul 1, si folosita apoi de catre capatul 1 pentru verificarea integritatii mesajului primit

Cu setarile implicite, cele doua chei de criptare vor fi identice, si la fel si cele doua chei HMAC, insa cu o configurare corecta obtinem 4 chei distincte.

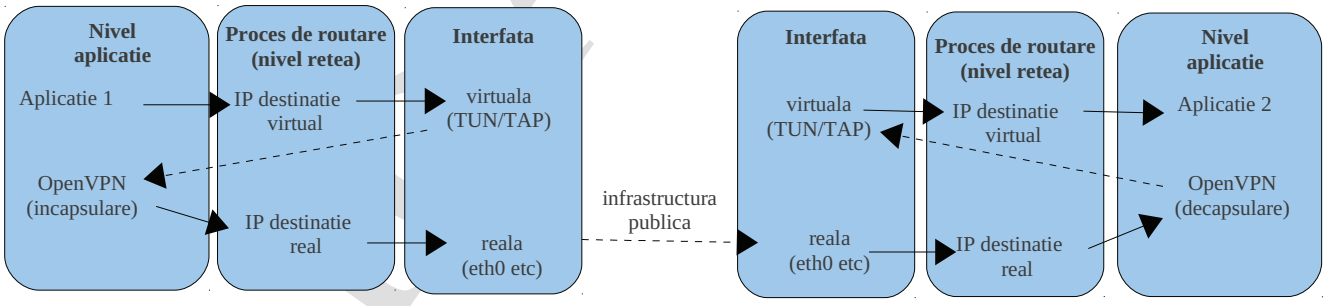
Cele 4 chei se obtin pe baza unui element secret comun, care poate fi obtinut pe doua cai, corespunzatoare celor doua modalitati de securizare a comunicatiei disponibile in OpenVPN:

- cheie statica pre-shared, din care se deriva apoi cele 4 chei necesare
- TLS+Diffie-Hellmann. Cele doua parti se autentifica reciproc folosind certificate digitale X.509, iar apoi folosesc Diffie-Hellmann pentru a genera elementul secret comun

Pentru sporirea securitatii si reducerea sanselor de atac DoS in cazul TLS, OpenVPN mai poate folosi inca o cheie in afara de cele enumerate. Aceasta este un pre-shared key folosit pentru a autentifica clientii in etapa de conectare (inaintea handshake-ului TLS) si care are rol de triere – va fi permisa conectarea numai pentru clientii care cunosc cheia corecta.

3.2.2. Detalii de implementare

Intr-un VPN implementat in userspace, pentru a putea realiza operatiile necesare asupra pachetelor trimise de aplicatiile beneficiare ale VPN-ului, traficul catre VPN trebuie “deturnat” prin aplicatia VPN. Acest lucru se realizeaza in Linux prin intermediul interfetelor virtuale de tip TUN sau TAP. O astfel de interfata se comporta pana la un punct ca una obisnuita din punct de vedere al codului de retea din kernel (poate avea adresa, rute in tabela de routare, reguli de filtrare atasate in firewall etc) insa ea nu corespunde unui dispozitiv fizic, ci trimite toate pachetele primite aplicatiei VPN. Aceasta efectueaza prelucrarile dorite si re-injecteaza pachetele in kernel, urmand ca acestea sa paraseasca statia pe una dintre interfetele fizice.



Interfetele de tip TUN sunt interfete care lucreaza in mod point-to-point si care opereaza la nivel retea (aplicatia pereche trebuie sa genereze – si va primi inapoi – pachete IP). Interfetele TAP sunt interfete ethernet, iar aplicatia care le utilizeaza trebuie sa genereze/prelucreze frame-uri.

O interfata point to point este una care conecteaza direct exact doua host-uri; in aceste conditii nu mai este nevoie ca pe interfata sa functioneze protocolul ARP, deoarece toate pachetele care parasesc interfata au o singura destinatie posibila. Configurarea interfetei presupune doua adrese – cele ale punctelor terminale pe care ea le conecteaza (asa-numitele

endpoint-uri): cel local (adresa configurata pe interfata in cauza) si cel remote (la care ajung pachetele transmise prin aceasta interfata). Iata un exemplu de configurare pentru o interfata virtuala de tip TUN:

```
# ifconfig tun0
tun0      Link encap:UNSPEC  HWaddr 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00
          inet addr:10.10.10.1  P-t-P:10.10.10.2  Mask:255.255.255.255
          UP POINTOPOINT RUNNING NOARP MULTICAST  MTU:1500  Metric:1
[… restul de output a fost omis...]
```

3.2.3. Moduri de lucru OpenVPN

OpenVPN poate fi configurat sa functioneze intr-unul dintre urmatoarele moduri de lucru:

- **p2p** (peer-to-peer) – openvpn interconecteaza doar doua statii (cu sau fara criptare). La o instanta de openvpn care functioneaza in acest mod nu se pot conecta mai multi clienti. Modul p2p este folosit pentru scenarii site to site, sau chiar remote access in cazul particular in care exista un singur client
- **server** – o aceeasi instanta de openvpn permite conectarea mai multor clienti, folosind acelasi port si aceeasi interfata tun/tap si multiplexand fluxurile de date ce circula prin ele. Modul server poate fi folosit atat pentru remote access cat si pentru site to site, oferind o buna scalabilitate, insa presupune aspecte suplimentare de care administratorul trebuie sa fie constient (vezi capitolul dedicat)

3.3. Elemente de configurare OpenVPN

3.3.1. Modalitati de configurare

OpenVPN se porneste prin rularea executabilului *openvpn*. Acesta poate fi configurat in doua moduri:

- prin optiuni pasate in linia de comanda la lansarea in executie. Optiunile sunt de forma --*optiune* <*parametru*> (doar o parte dintre optiuni au si parametru)

```
openvpn --proto udp --port 1234 --remote 1.2.3.4 --secret cheie.key
```

- prin directive de configurare plasate intr-un fisier text. Directivele au acelasi nume ca optiunile corespondente, insa nu mai sunt precedate de --

```
# fisierul openvpn.conf
proto udp
port 1234
remote 1.2.3.4
secret cheie.key
```

OpenVPN nu are un fisier de configurare prestabilit, din cauza faptului ca putem porni mai multe instante de OpenVPN simultan (ruland pe porturi diferite si cu setari diferite) si avand fiecare nevoie de un alt fisier de configurare. Fisierul de configurare dorit este indicat la lansarea in executie a executabilului *openvpn* – fie pasandu-l ca unic argument, fie folosind optiunea –*config*:

```
# atunci cand folosim si fisier de configurare si optiuni separate, --config este obligatoriu:
openvpn --config server.conf --dev tun --port 4321

# atunci cand fisierul de configurare este unicul argument putem omite –config:
openvpn server.conf
```

Atunci cand exista suprapunere intre o optiune din fisierul de configurare si una pasata la apelare, cea din linia de comanda are intaietate.

Fisierele de configurare pot contine comentarii; acestea incep cu # sau cu ; .

3.3.2. Configurare de baza

Exista un set de elemente de configurare necesare, comune tuturor scenariilor in care functioneaza OpenVPN:

- protcolul folosit – TCP sau UDP. Din motive enumerate anterior se prefera UDP ori de cate ori este posibil. Directiva utilizata este *proto*, cu valori posibile *tcp* sau *udp*
- portul pe care openvpn asteapta conexiuni – se stabileste cu ajutorul directivei *port* urmata de numarul portului
- tipul de interfata virtuala de retea pe care o va folosi OpenVPN – se utilizeaza directiva *dev* urmata de cuvintele cheie *tun* sau *tap* (in functie de tipul de VPN realizat – routed sau bridged)
- modul de lucru openvpn (punct-la-punct sau multi-client) – se stabileste cu directiva *mode* urmata de una dintre valorile *p2p* sau *server*. Valoarea default este p2p
- modul de rulare al procesului openvpn (foreground sau background). Implicit openvpn ruleaza in foreground; directiva *daemon* face ca procesul sa se detaseze de la terminalul curent imediat dupa pornire

```
# exemplu de fisier de configurare de baza
proto udp
port 1234
dev tun
mode p2p
daemon

# echivalent cu a porni openvpn astfel:
#  openvpn --proto udp --port 1234 --dev tun --mode p2p --daemon
```

3.3.3. Configurare logging

Atunci cand este rulat din linia de comanda, openvpn ruleaza automat in foreground, trimitandu-si mesajele de diagnostic catre terminalul din care a fost pornit. Informatia poate fi memorata si in fisiere, folosind urmatoarele directive:

- **log** *cale* – stabileste calea catre fisierul in care openvpn consemneaza informatiile legate de pornirea si functionarea sa. **Atentie! Daca exista, fisierul este suprascris la pornirea openvpn!** Asadar nu se pot crea loguri persistente folosind aceasta directiva.
- **log-append** *cale* – are acelasi efect ca log, insa daca fisierul exista va adauga informatia la sfarsitul acestuia
- **status** *cale* [ns] – creeaza un fisier cu informatii de stare, in care openvpn va scrie statistici de functionare odata la *ns* secunde

Indiferent daca logurile sunt afisate doar pe ecran sau si in fisiere, gradul lor de detaliu se controleaza cu ajutorul directivei verb, urmata de nivelul de logging sub forma unui numar intre 0 si 11 (numar mai mare inseamna grad de detaliu superior). Nivelurile de logging au urmatoarele semnificatii:

- 0 – sunt consemnate numai erorile fatale
- 1-4 – nivelurile de logging utile administratorului openvpn
- 5 – se indica pe ecran sub forma unei litere primirea/trimiterea fiecarui pachet
- 6-11 – niveluri de debug, utile mai degraba dezvoltatorilor

```
# exemplu de fisier de configurare pentru logging persistent
log-append /var/log/openvpn
verb 3
```

3.4. Functionarea in mod p2p routed

3.4.1. Principii

Va fi prezentata in continuare modalitatea de configurare a unui tunel routed, cu openvpn functionand in modul peer-to-peer, fara criptare. Aceasta reprezinta o baza pentru scenariile reale, deoarece poate fi apoi usor extinsa prin configurarea criptarii canalului de comunicatie, adaugarea de rute pentru a asigura vizibilitatea retelelor din cele doua capete etc. Aceasta baza prezinta urmatoarele particularitati:

- functionare openvpn in modul peer-to-peer. In acest mod de lucru, o instanta de openvpn se conecteaza la o singura alta instanta, creand un tunel punct la punct. Cu acest setup putem implementa scenarii site-to-site, sau chiar remote access in cazul particular in care avem un singur client. Cazul multi-client (modul de functionare server) va fi prezentat intr-un capitol separat deoarece implica multiple aspecte suplimentare importante (alocare dinamica a adreselor si rutelor conform cu diverse topologii etc)
- tunelul nu beneficiaza de criptare, asadar avem pe moment un VN (P-ul din VPN lipsind). Abordarea este utila atat in etapa de invatare, pentru a gandi si urmari separat routarea si criptarea, cat si in cazul real, pentru diagnosticarea pe componente a unui sistem VPN. Criptarea beneficiaza si ea de propriul sau capitol in cadrul acestui material

3.4.2. Etape

Etapele necesare presupun configurarea celor doua capete de asa natura incat unul sa initieze conexiunea la celalalt. Vom numi capatul care initiaza *client* si celalalt capat *server*, desi ambele instante de OpenVPN deschid un port si sunt pregatite sa primeasca cereri de conexiune.

Operatii necesare:

- server:
  - configurare generala (protocol, port etc)
  - configurarea adreselor de pe interfata virtuala (local si remote endpoint). Ca urmare vor fi adaugate automat rutele corespunzatoare in tabela de routare
- client
  - configurare generala
  - configurarea adreselor de pe interfata virtuala. Adresele endpoint-urilor vor fi in oglinda fata de server (ceea ce reprezenta endpoint-ul local pentru server este endpoint-ul local pentru client si invers)
  - adresa serverului

Nota: *am putea folosi adrese complet separate pentru endpoint-urile de pe server si de pe client, insa 1) am face risipa de adrese si 2) ar trebui sa adaugam rute suplimentare in cele doua parti pentru a asigura interconectivitatea.*

3.4.3. Implementare

Pentru configurarea interfetei virtuale TUN se fololseste, atat pe server cat si pe client, directiva **ifconfig**, urmata de doua adrese: local endpoint-ul si remote endpoint-ul. Adresa de local endpoint a serverului constituie remote endpoint pentru client si invers. In plus, una dintre statii trebuie sa initieze conexiunea: statia client va folosi directiva **remote** pentru a stabili adresa serverului si a se conecta la aceasta:

```
# server
proto udp
port 1234
dev tun
ifconfig      10.10.10.1          10.10.10.2
```

```
# client
proto udp
port 1234
dev tun
ifconfig      10.10.10.2          10.10.10.1
remote 1.2.3.4
```

Nota: *desi multe dintre operatiile de configurare de adrese si de rute ar putea fi efectuate manual de catre administrator folosind comenzile Linux* ifconfig *si* route *(mai ales in cazul configurarii statice, cum este cazul acesteia), OpenVPN dispune de directivele* ifconfig *si* route *pentru a oferi o interfata unitara indiferent de sistemul de operare pe care ruleaza (exista si versiuni de OpenVPN de Windows).*

3.4.4. Controlul accesibilitatii retelelor terminale

Cu configurarea anterioara avem un tunel punct-la-punct intre doua statii; acesta poate fi usor extins in doua moduri:

- daca clientul are acces la statiile aflate in reteaua din spatele serverului, avem un scenariu remote-access
- daca, in plus, si reteaua din spatele serverului are acces la cea din spatele clientului, ne gasim intr-un scenariu site-to-site

Ambele extensii ale scenariului de baza sunt realizabile prin adaugarea de rute pe server, pe client sau pe ambele, dupa cum urmeaza:

- pentru ca clientul sa trimita pachetele catre reteaua serverului prin tunel este necesar ca el sa dispuna de o ruta catre acea retea asociata interfetei de tip TUN. Cu aceasta am asigurat sensul “dus” al pachetelor
- pentru a asigura intoarcerea pachetelor trebuie ca statiile din reteaua serverului fie sa aiba ruta explicita catre client, fie sa aiba serverul pe post de ruta default, caz in care pachetele, odata ajunse pe server, au deja ruta definita catre client
- prin analogie (dar facand configurarile “in oglinda”) se asigura si accesul serverului la statiile din reteaua clientului

Adaugarea rutelor se realizeaza cu ajutorul directivei **route** cu urmatoarea sintaxa:

```
route retea/IP [netmask] [gateway] [metrica]
```

Parametrul *retea/IP* poate fi fie o adresa de retea, atunci cand ruta trimite catre un intreg subnet, fie un IP individual atunci cand se defineste un host route (o ruta catre o statie). Directiva route poate fi utilizata pentru a adauga orice fel de rute, inclusiv catre retele care nu sunt direct conectate, de unde si necesitatea prezentei parametrului *gateway*. In toate cazurile, ruta va fi atasata interfetei virtuale de tip TUN (pachetele vor fi trimise prin tunel).

Exemplu: fie un server si un client OpenVPN conectate prin internet, fiindc fiecare default gateway pentru reteaua proprie si avand urmatoarele configurari:

- server: adresa internet 1.2.3.4, adresa LAN 10.0.0.1, adresa virtuala 10.10.10.1
- client: adresa internet 5.6.7.8, adresa LAN 192.168.0.1, adresa virtuala 10.10.10.2

Rutele care ar trebui adaugate pentru a realiza conectivitate site-to-site (statiile din reteaua 10.0.0.0/24 sa poata comunica cu statiile din reteaua 192.168.0.0/24) sunt urmatoarele:

```
# in fisierul de configurare al serverului
route 192.168.0.0 255.255.255.0

# in fisierul de configurare al clientului
route 10.0.0.0/24 255.255.255.0
```

Sa urmarim traseul unei perechi de pachete echo request-reply intre o statie din reteaua clientului si una din reteaua serverului: pachetul echo request cu sursa 192.168.0.100 si destinatia 10.0.0.100 generat pe statia de origine este directionat catre default gateway, care este statia ce tine capatul client al tunelului; aici exista ruta explicita catre reteaua 10.0.0.0/24 prin tunel. Astfel pachetul ajunge pe statia server, unde exista ruta catre reteaua (de data aceasta direct conectata) 10.0.0.0/24 in virtutea adresei de pe interfata sa de LAN. Pachetul ajunge la statia 10.0.0.100 si genereaza un echo reply cu destinatia 192.168.0.100 care urmeaza un traseu asemanator: este pasat default gateway-ului (serverul), acesta are ruta catre 192.168.0.0/24 prin tunel si astfel pachetul ajunge pe client, unde gratie rutei catre reteaua direct conectata 192.168.0.0/24 va fi livrat statiei destinatie.

3.5. Securizarea comunicatiei

3.5.1. Principii

Dupa cum s-a explicat anterior, securizarea este o operatie independenta de routare si incapsulare – un serviciu suplimentar care asigura confidentialitatea, integritatea si autenticitatea datelor. Pentru criptare este nevoie de o cheie

simetrica sigura (care sa ajunga in posesia ambelor parti fara a fi transmisa ca atare pe mediul de transmisiune). Cheia poate fi:

- statica (pre-shared) – este o solutie simplu de configurat pentru capete de VPN aflate sub administrare comuna, insa prezinta urmatoarele dezavantaje:
  - cheile nu se schimba automat de-a lungul timpului, ceea ce deschide calea spargerii lor daca raman constante suficient de mult timp
  - compromiterea cheii statice duce la compromiterea comunicatiilor prezente si trecute, deoarece cheile secrete sunt generate pe baza celei statice si vor fi de fiecare data aceleasi
  - solutia nu poate fi folosita in scenarii multi-client (modul de lucru *server* al OpenVPN impune utilizarea TLS)
- generata cu Diffie-Hellmann printr-o conexiune criptata TLS, care asigura autentificarea initiala a celor doua parti. Solutia este scalabila si potrivita pentru statii care nu au mai comunicat niciodata si nici nu se afla sub administrare comuna, deoarece autentificarea se realizeaza pe baza de certificate digitale X.509 emise de un CA comun. In plus, avand in vedere ca elementul secret este generat pe loc la nivel de fiecare sesiune, compromiterea uneia dintre cheile private ale capetelor nu poate afecta sesiunile trecute

Directivele explicate in continuare pentru securizarea cu TLS se aplica in acelasi fel pentru ambele moduri de lucru ale OpenVPN (p2p si server).

3.5.2. Solutia cu chei statice pre-shared

3.5.2.1. Etape

Aceasta solutie este posibila numai in modul de functionare p2p al OpenVPN. Pentru a o implementa sunt necesare urmatoarele operatii:

- generarea cheii simetrice
- plasarea ei pe ambele statii ce vor constitui capetele VPN-ului
- configurarea OpenVPN in nambele capete sa foloseasca aceasta cheie

3.5.2.2. Implementare

Pentru generarea cheii se foloseste chiar executabilul *openvpn*, pasandu-i optiunile *--genkey* si *--secret.* Optiunea *--secret* este obligatorie si stabileste calea catre fisierul cheie generat:

```
$ openvpn --genkey --secret cheie.key
...iar fisierul generat va debuta astfel:
#
# 2048 bit OpenVPN static key
#
-----BEGIN OpenVPN Static key V1-----
5cd946d0a72d27fe27cce8d4c27b7707
[...restul fisierului a fost omis...]
```

Distribuirea cheii pe cele doua statii se realizeaza prin orice mijloace considerate sigure de catre administratorul VPN (ssh, scp etc).

Pentru a configura cheia pe cele doua statii se utilizeaza directiva secret, cu sintaxa:

```
secret fisier [directie]
```

Pe baza cheii cuprinse in fisier se genereaza cheile necesare OpenVPN. Implicit vor fi folosite 2 chei - una pentru criptare (aceeasi in ambele parti) si una HMAC (din nou identica in ambele parti). Pentru a activa utilizarea a 4 chei diferite este necesara folosirea parametrului *directie*, care poate avea valoarea 0 sau 1. **Atentie! Valoarea trebuie sa fie diferita pe server si pe client!** Explicatia este ca, pe baza cheii simetrice din fisier, OpenVPN va genera in ambele capete aceleasi 4 chei, si trebuie reglementat care capat ce chei foloseste.

3.5.3. Solutia cu autentificare TLS

3.5.3.1. Principii

Autentificarea TLS presupune prezenta unor certificate digitale in cele doua capete. Acestea trebuie sa fie emise de acelasi CA, declarat ca trusted in ambele fisiere de configurare.

Pasii si directivele prezentate in continuare se aplica in ambele moduri de lucru ale OpenVPN (p2p si server).

3.5.3.2. Etape

Setul de operatii necesar este urmatorul:

- crearea CA-ului comun
- crearea cheii+certificatului pentru server
- crearea cheilor+certificatelor pentru clienti. Certficatele vor fi emise de CA-ul creat anterior
- generarea parametrilor Diffie-Hellmann pe server si clienti
- configurarea parametrilor DH, cheilor si certificatelor pe server si clienti
- pornire si testare

3.5.3.3. Pachetul de scripturi easyrsa

Pentru managementul CA-ului si al cheilor/certificatelor se poate folosi orice solutie, inclusiv openssl in linia de comanda. OpenVPN pune insa la dispozitia administratorului un set de scripturi care automatizeaza aceste operatii si care creeaza certificatele cu setarile corecte; intregul ansamblu se gaseste intr-un director numit *easy-rsa*, aflat de obicei in undeva in /usr/share, locatia diferind usor de la o distributie la alta (/usr/share/openvpn, /usr/share/doc/openvpn/examples etc). Printre scripturile de interes se numara:

- **vars -** contine setarile comune tuturor scripturilor ce compun easyrsa. Dupa editare trebuie executat cu source inaintea utilizarii oricarui alt script
- **pkitool** - scriptul principal, folosit pentru crearea CA-ului si semnarea certificatelor. Poate primi urmatoarele optiuni/argumente:
  - --initca - creeaza un root CA
  - --server - folosit pentru a construi un certificat de server (in care keyUsage si extendedKeyUsage reflecta statutul de server)
  - --sign - folosit pentru semnare de CSR
  - --pass - la crearea de chei/certificate le protejeaza prin parola
  - *numeclient* - creeaza un certificat de client semnat de catre CA-ul curent
- **build-dh** - genereaza parametrii Diffie-Hellmann. Generarea acestora consuma in general mult timp (e vorba de niste numere mari cu anumite caracteristici particulare)
- **clean-all** - initializeaza intregul sistem (sterge CA-ul si toate certificatele emise de acesta)

Toate cheile si certificatele (inclusiv cele ale CA-ului) sunt create in subdirectorul *keys/* al directorului easyrsa.

3.5.3.4. Creare CA si generarea certificatelor pentru server si clienti

Setul de pasi necesar pentru a obtine un certificat de server si unul de client este prezentat in continuare (toate comenzile pornesc de la premiza ca directorul curent este cel care contine scripturile):

```
# intai de toate, editare vars daca este nevoie, si apoi preluarea setarilor din el
$ vim vars
$ source vars

# curatarea intergului ansamblu
$ ./clean-all
```

```
# generarea parametrilor Diffie-Hellmann (creeaza fisierul dh1024.pem in subdirectorul keys/ )
$ ./build-dh

# creare CA (ca urmare, se genereaza ca.key si ca.crt in subdirectorul keys)
$ ./pkitool --initca

# crearea certificatului de server (se foloseste ca argument DN-ul serverului)
$./pkitool --server DNserver
# efect: se creeaza DNserver.key, DNserver.csr si DNserver.crt in subdirectorul keys/

# crearea certificatelor de clienti (argumentele sunt DN-urile clientilor)
$ ./pkitool client1
$ ./pkitool client2
# efect: sunt create fisierele client1.key, client1.csr si client1.crt in subdirectorul keys/
```

In scenariile multi-client, ultima comanda se aplica de oricate ori este nevoie, pentru fiecare DN de client in parte.

In continuare se distribuie fisierele:

- certificatul CA-ului pe toate masinile (cheia privata ramane pe loc!)
- parametrii DH pe toate masinile
- cheia privata si certificatul serverului pe masina server
- cheile private si certificatele pentru clienti pe fiecare masina client in parte

3.5.3.5. Configurarea serverului si a clientilor

Configurarea statiilor implicate presupune doua aspecte:

- folosirea directivelor *ca, dh, key, cert* pentru configurarea parametrilor criptografici, dupa cum urmeaza:

```
# exemplu de configurare pentru server
ca /etc/openvpn/ca/crt
dh /etc/openvpn/dh1024.pem
key /etc/openvpn/DNserver.key
cert /etc/openvpn/DNserver.crt

# configurarile de client folosesc aceleasi directive
```

- folosirea directivelor **tls-server** si **tls-client** pentru a indica felul in care este gestionat dialogul TLS de catre diversele statii implicate (clientii initiaza, serverul doar asteapta cereri)

```
# pe server
tls-server

# pe clienti
tls-client
```

Nota: tls-server *nu implica modul de lucru server (multi-client), ci doar indica faptul ca partenerul de dialog este cel care initiaza negocierea in vederea stabilirii canalului de comunicatie criptat.*

3.6. Functionarea in mod server (multi-client)

3.6.1. Principii si particularitati

Initial, OpenVPN functiona numai in mod punct la punct. Modul server a fost introdus odata cu versiunea 2.0 si presupune ca o singura instanta OpenVPN sa poata avea mai multi clienti conectati simultan, utilizand acelasi port si aceeasi interfata virtuala TUN/TAP.

Atentie! Modul de lucru server impune TLS ca solutie de criptare a comunicatiei! Aceasta deoarece OpenVPN suporta configurari per-client, si pentru a identifica clientul (in scopul de a-i aplica configurarea atribuita) foloseste DN-ul acestuia.

Functionarea in modul server ridica doua probleme noi:

- **alocarea adreselor virtuale catre clienti** (plus rutele aferente) – exista doua abordari posibile:
  - alocare dinamica de adrese + alte setari de catre server. Serverul dispune de un lot de adrese (un asa-numit *pool*) pe care le aloca clientilor pe masura ce acestia se conecteaza, configurand in acelasi timp automat si rutele catre adresele nou-alocate
  - alocare statica a adreselor de client. Se poate defini pe server un director de fisiere de configurare per-client; fiecare fisier poate contine o adresa statica dedicata clientului in cauza
- **crearea unei tabele interne de routare openvpn**. Deoarece toate pachetele catre clienti ii parvin lui OpenVPN prin aceeasi interfata virtuala TUN/TAP, acesta trebuie sa poata decide carui client ii sunt adresate. Sa consideram exemplul unei aplicatii care comunica prin OpenVPN si care ruleaza pe statia server; succesiunea de operatii care i se aplica unui pachet este urmatoarea:
  - aplicatia genereaza pachetul si i-l “inmaneaza” kernelului
  - kernelul decide, pe baza tabelei de routare, ca pachetul trebuie trimis prin interfata tun0
  - pachetul este trimis prin interfata tun0 care il paseaza catre instanta server de OpenVPN
  - OpenVPN se afla in fata unui pachet pe care trebuie sa il incapsuleze/cripteze/etc si sa il trimita unui anumit client. Problema este ca la aceeasi instanta server de OpenVPN sunt conectati mai multi clienti, si OpenVPN este singurul in masura sa decida caruia dintre ei sa trimita pachetul (tabela de rutare externa – a sistemului de operare – deja “si-a facut treaba”, nemaifiind de folos in acest punct). In plus, pachetul poate fi destinat nu numai unuia dintre clientii directi, ci unei statii aflate intr-o retea din spatele unui client. Din acest motiv, OpenVPN trebuie sa mentina o tabela de rute interne pentru a determina destinatia corecta a pachetelor care parasesc serverul.

Nota: *lista de rute interne poate fi vizualizata din consola de management OpenVPN (vezi capitolul dedicat).*

3.6.2. Alocarea dinamica a adreselor de client

3.6.2.1. Directive

OpenVPN ofer urmatoarele directive care tin de configurarea dinamica a clientilor in modul de functionare server:

- **ifconfig-pool** *adresa_start adresa_stop* - configurarea lotului de adrese IP alocate in mod dinamic clientilor (se va aloca de la adresa_start pana la adresa_stop)
- **ifconfig-pool-persist** *fisier N* - determina pastrarea (pe cat posibil) a corespondentelor client-IP. Corespondentele vor fi scrise in fisierul specificat odata la N secunde
- **ifconfig** *IPlocal IPremote_sau_netmask* - desi deja prezentata, aceasta directiva care configureaza adresele de pe interfetele virtuale are un al doilea argument care depinde de topologia folosita in modul de lucru server (vezi mai jos prezentarea topologiilor)
- **topology** *topologie* - determina strategia de alocare a adreselor virtuale pentru server si clienti (vezi mai jos)

3.6.2.2. Despre topologii si implicatiile acestora

Felul in care serverul aloca adrese IP pentru clientii nou-conectati depinde de valoarea directivei de configurare *topology*. Valori posibile:

- **subnet** – adresa serverului si adresele clientilor fac parte din acelasi subnet, care are masca diferita de 255.255.255.255. Fiecare nou client conectat primeste urmatoarea adresa libera din pool
- **p2p** – adresele de server si de clienti sunt adrese de host (masca 255.255.255.0); remote endpoint-ul de pe client este local endpoint-ul de pe server
- **net30** – creeaza o topologie punct la punct, insa complicata in mod artificial pentru a trece de limitarile impuse de unele versiuni mai vechi de drivere TUN/TAP de Windows: fiecare client primeste un subnet /30 – asadar doua adrese utilizabile – pe care le va folosi ca local/remote endpoint pentru interfata sa virtuala tun. Este o topologie ce duce la o mare risipa de adrese si care trebuie evitata pe cat posibil

In toate topologiile, interfetele virtuale TUN sunt configurate in mod point-to-point, insa adresele si mastile de retea sunt cele care difera, dupa cum se va vedea in continuare.

Atentie! *Valoarea directivei topology trebuie sa fie identica pe server si pe clienti! Acest lucru se poate realiza fie prin configurarea manuala a tuturor statiilor implicate, fie prin transmiterea acestui parametru catre clienti folosind directiva push.*

3.6.2.3. Topologia subnet

In topologia subnet, interfetele virtuale de pe server si clienti sunt configurate in mod point-to-point, insa cu netmask diferit de 255.255.255.255 (cel obisnuit in cazul p2p). In aceste conditii, pe server si clienti se adauga automat in tabela de routare o singura ruta (indiferent de numarul de clienti), care trimite intregul subnet corespunzator endpoint-ului remote prin tunel. In acest fel serverul si clientii au conectivitate inca de la stabilirea tunelului fara a mai fi nevoie de interventie suplimentara din partea administratorului.

Atentie! La configurarea adreselor virtuale ale serverului in aceasta topologie, al doilea argument al directivei ifconfig este masca de retea dorita!

Exemplu de configurare:

```
# pe server
topology subnet
push "topology subnet"
ifconfig 10.10.10.1 255.255.255.0
ifconfig-pool 10.10.10.2 10.10.10.254
```

3.6.2.4. Topologia p2p

In aceasta topologie, interfetele virtuale de pe server si clienti au urmatoarele caracteristici:

- au mask 255.255.255.255, deci nu mai fac parte din acelasi subnet
- interfata TUN de pe server are ca remote endpoint o adresa oarecare (existand mai multi clienti, nici nu ar avea rost sa se refere la unul anume)
- interfata TUN de pe client are ca remte endpoint IP-ul virtual de pe server (endpoint-ul local al interfetei TUN a serverului)

Al doilea argument al lui ifconfig este un IP (remote endpoint-ul serverului).

Fiecare dintre cele doua parti (server/client) adauga rute automat numai pentru remote endpoint-ul sau; in consecinta, serverul va avea ruta numai catre remote endpoint-ul virtual si nu catre vreun client. De aceea este necesara adaugarea de rute pe server pentru clienti - fie rute individuale, fie o singura ruta catre intregul subnet al clientilor (daca acestia fac parte din acelasi subnet) - atasate interfetei TUN.

3.6.2.5. Topologia net30

Aceasta este topologia implicita OpenVPN deoarece ofera compatibilitate maxima cu clientii. Dupa cum s-a explicat, alocarea adreselor este departe de a fi optima, de aceea aceasta topologie trebuie evitata daca natura clientilor nu impune folosirea ei.

Al doilea argument al directivei ifconfig de pe server este un IP (remote endpoint) care insa nu se regaseste pe clienti.

In aceasta topologie, toate cele 4 endpoint-uri sunt diferite, motiv pentru care, by default, nici serverul, nici clientul nu au rute unul catre altul si nici catre retelele din spatele lor; toate aceste rute trebuie adaugate - fie din OpenVPN, fie prin mijloace externe.

Concret: daca clientul primeste adresa 10.10.10.130 pe tun0, cu endpoint-ul remote 10.10.10.129, iar serverul are local endpoint 10.10.10.13 si remote 10.10.10.4, atunci rutele vor arata astfel:

```
# pe client:
10.10.10.129    0.0.0.0         255.255.255.255 UH    0      0        0 tun0

# pe server:
10.10.10.14     0.0.0.0         255.255.255.255 UH    0      0        0 tun0
```

3.6.2.6. *Configurarea rapida cu noile directive server si client*

Directivele server si client sunt directive-umbrela care incearca sa automatizeze configurarile necesare pentru server si clienti:

- directiva **client** este echivalenta cu a folosi directivele *tls-client* si *pull* pe client
- directiva server are un echivalent mai complex, explicat mai jos

Exemplu: directiva *server 10.8.0.0 255.255.255.0* este echivalenta cu (urmeaza logica de expandare a directivei server sub forma de pseudocod):

```
mode server # implica modul de lucru multi-client
tls-server
push "topology [topologie_curenta]"

if dev tun AND (topology == net30 OR topology == p2p):
  ifconfig 10.8.0.1 10.8.0.2    # adresele virtuale de pe interfata TUN a serverului
  if !nopool:
    ifconfig-pool 10.8.0.4 10.8.0.251  # adresele alocate clientilor
  route 10.8.0.0 255.255.255.0         # pachetele catre clienti sa plece prin interfata TUN
  if client-to-client:   # daca se doreste ca clientii sa aiba conectivitate unul cu altul
    push "route 10.8.0.0 255.255.255.0" # pt ca pachetele clientului sa poata ajunge la server
  else if topology == net30:
    push "route 10.8.0.1" # pt ca clientul sa ajunga la server prin IP-ul remote de P2P
                          # pe care l-a primit
if dev tap OR (dev tun AND topology == subnet):
  ifconfig 10.8.0.1 255.255.255.0
  if !nopool:
    ifconfig-pool 10.8.0.2 10.8.0.254 255.255.255.0
  push "route-gateway 10.8.0.1"
```

3.6.3. Configurari particularizate per-client; aplicare pentru alocare statica

Un server OpenVPN are posibilitatea de a diferentia clientii care se conecteaza si de a le oferi configurari distincte. Clientii sunt identificati dupa DN-ul pe care il prezinta serverului in certificatul digital cu care se autentifica. Administratorul OpenVPN poate configura pe server un director dedicat ce contine fisiere de configurare per-client; odata autentificarea TLS incheiata si clientul conectat, serverul va cauta in acel director un fisier care are nume identic cu DN-ul clientului si, in cazul existentei acestui fisier, ii va aplica clientului in cauza directivele cuprinse in acesta. Daca nu exista un astfel de fisier, serverul va cauta un fisier numit DEFAULT si va aplica eventualele directive aflate acolo.

Calea catre directorul de configurari per-client se stabileste cu directiva **client-config-dir**:

```
# pe server
client-config-dir /ec/openvpn/clients
```

Fisierele de configurare per-client pot contine urmatoarele directive: *push, push-reset, iroute, ifconfig-push, config.*

Spre exemplu, daca dorim sa atribuim unui anume client o adresa statica - aceeasi de fiecare data - putem scrie in fisierul sau de configurare aflat pe server:

```
# client1.conf
ifconfig-push 10.10.10.100 10.10.10.1
```

Nota: *fisierele de configurare per-client au avantajul ca sunt analizate abia in momentul conectarii unui nou client, si deci pot fi modificate fara a fi necesar restart de server.*

3.6.4. Controlul accesibilitatii retelelor terminale

Pentru a controla conectivitatea clientilor cu reteaua serverului, sau a retelei serverului cu retelele din spatele clientilor, se procedeaza pentru inceput la fel ca in cazul modului p2p, prin adaugare de rute. In modul server apare insa posibilitatea suplimentara a conectivitatii inter-clienti sau a retelelor din spatele acestora. Desigur, aceasta conectivitate nu va fi directa, ci intermediata de server; ea trebuie activata explicit pe server folosind directiva client-to-client:

```
# in fisierul de configurare al serverului
client-to-client
```

3.7. BIBLIOGRAFIE

- OpenVPN:
  - concepte & arhitectura
    - http://openvpn.net/papers/BLUG-talk/index.html
    - http://www.sans.org/rr/whitepapers/vpns/1459.php
  - doumentatie:
    - http://openvpn.net/index.php/open-source/documentation.html
    - manpage: http://openvpn.net/index.php/open-source/documentation/manuals/69-openvpn-21.html
  - configurare:
    - Beginning OpenVPN 2.0.9 – Cap 9 (The Command openVPN and its Configuration File)
    - OpenVPN in Ubuntu 10.10: https://help.ubuntu.com/10.10/serverguide/C/openvpn.html
  - carti: http://openvpn.net/index.php/open-source/books.html