9. SERVERUL E-MAIL – SECURIZARE SI CONFIGURARE AVANSATA

9.1. Protocolul STMP

9.1.1. Anatomia mesajului si a dialogului SMTP

Ca si in cazul HTTP, protocolul STMP este unul in care informatia circula clear-text, cu urmatoarele caracteristici:

- schimbul de mesaje presupune cereri din partea clientului si raspunsuri corespunzatoare din partea serverului
- raspunsurile serverului contin un cod de raspuns si un mesaj
- mesajul e-mail transmis intre un server si un client sau intre doua servere cuprinde headere si eventual continut

Pentru o scurta familiarizare cu componenta mesajului SMTP si etapele dialogului, fie urmatorul exemplu in care clientul se autentifica si apoi transmite serverului un mesaj ce include headere si continut. Comenzile clientului sunt evidentiate in bold:

220-s01.rohost.com ESMTP Exim 4.69 #1 Wed, 04 Aug 2010 12:14:25 +0300 220-We do not authorize the use of this system to transport unsolicited, 220 and/or bulk e-mail.	Mesajul de intampinare al serverului (asa-numitul "greeting")
EHLO [192.168.1.3]	Clientul se prezinta, indicand faptul ca suporta ESMTP. Serverul raspunde cu setul de extensii SMTP suportate
250-s01.rohost.com Hello [192.168.1.3] [188.26.235.63] 250-SIZE 52428800 250-PIPELINING 250-AUTH PLAIN LOGIN 250-STARTTLS 250 HELP	
AUTH PLAIN yrrwfHBKLJIOy54543wsxgbPO546f5edgfcnmWScnmnklUR=	Clientul se autentifica pentru a i se permite relay-ul
235 Authentication succeeded	
MAIL FROM:<ionut@infoacademy.net> SIZE=442	Aceasta parte a dialogului formeaza asa-numitul "**envelope**" (plic), deoarece specifica expeditorul si destinatarul.
250 OK	
RCPT TO:<studentlinux@yahoo.com>	
250 Accepted	
DATA	Clientul indica faptul ca va incepe sa transmita mesajul
354 Enter message, ending with "." on a line by itself	
Message-ID: <4C592F81.9020301@infoacademy.net> **Date: Wed, 04 Aug 2010 12:14:41 +0300** **From: Ionut Morar <ionut@infoacademy.net>** **User-Agent: Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.9.1.8)** **Gecko/20100227 Thunderbird/3.0.3** **MIME-Version: 1.0** **To: Student Linux <studentlinux@yahoo.com>** **Subject: Test** **Content-Type: text/plain; charset=ISO-8859-1; format=flowed** **Content-Transfer-Encoding: 7bit**	Mesajul debuteaza cu o serie de headere, de forma *Nume-Header: valoare*. Headerele sunt urmate de o linie goala care le separa de continut
Mesaj de exemplificare a dialogului SMTP. **Enjoy :-)** .	Continutul mesajului (asa-numitul "body"). Caracterul . aflat singur pe linie reprezinta semnalizatorul de final de mesaj.
250 OK id=1Oga3N-0003X5-QE	
QUIT	Clientul inchide sesiunea
221 s01.rohost.com closing connection	

Distingem urmatoarele etape ale dialogului:

- mesajul de intampinare al serverului, in care acesta poate sa prezinte sau nu numele si versiunea softului de server folosit (configurabil)
- prezentarea clientului prin intermediul comenzii EHLO, care are doua roluri:
  - argumentul sau reprezinta numele DNS sau adresa clientului. Serverul are optiunea de a aplica validari numelui/adresei prezentate si de a respinge clientul daca nu corespunde criteriilor dorite
  - comanda indica ca clientul suporta ESMTP, adica poate folosi una sau mai multe extensii ale protocolului SMTP daca serverul le ofera

  Nota: *in versiunile mai vechi de SMTP clientul se prezenta cu comanda HELO; aceasta este inca suportata in ziua de astazi, dar declasata in favoarea lui EHLO. Ca urmare a unei comenzi HELO, serverul deduce ca clientul nu are capabilitati ESMTP si nu ii va prezenta lista sa de extensii SMTP.*

- constatand ca clientul are capabilitati ESMTP, serverul raspunde cu setul de extensii SMTP suportate
- autentificarea clientului – etapa optionala; reprezinta o extensie a protocolului SMTP (anuntata de altfel de catre server in etapa anterioara)
- envelope – reprezinta etapa transmiterii de catre client a adresei expeditorului si destinatarului. Serverul are optiunea de a aplica validari acestor informatii si de a respinge clientul in caz de invaliditate. Cele doua informatii sunt folosite astfel:
  - expeditorul este cel catre care sunt trimise mesajele de eroare in caz de imposibilitate a livrarii (bounce-urile)
  - destinatarul este cel catre care va fi livrat mesajul. **Atentie! Chiar daca mesajul include un header To: , mesajul va fi livrat catre destinatarul din envelope!**
- transmiterea mesajului, a carui structura presupune urmatoarele elemente:
  - header-e – se afla cate unul pe linie si au acelasi format ca in HTTP: *Nume-Header: valoare*. Serverul poate de asemenea aplica filtrari in functie de valorile headerelor
  - linie goala, care separa headerele de continut
  - body (corpul mesajului), care contine datele scrise/atasate de catre utilizator

9.1.2. Coduri de raspuns SMTP

S-a observat in exemplul de mai sus faptul ca orice raspuns al serverului consta dintr-un cod si un mesaj. Codurile se incadreaza in urmatoarele categorii:

- **2xx** – coduri care indica succesul operatiei cerute de client. Amintim aici 220 (serverul pregatit de a primi comenzi), 250 (cererea clientului a fost onorata)
- **3xx** – confirmare intermediara. Folosit in general in operatiile care presupun succesiuni de comenzi din partea clientului. Spre exemplu, transmiterea unui mesaj presupune mai intai comanda DATA si apoi continutul mesajului; dupa primirea comenzii DATA serverul raspunde cu un cod 354 (“start mail input”).
- **4xx** – eroare temporara. Clientul poate reincerca mai tarziu operatia solicitata. Amintim 450 (casuta postala temporar indisponibila), 451 (eroare locala la procesarea comenzii)
- **5xx** – eroare permanenta. Clientul nu trebuie sa incerce din nou aceeasi operatie/succesiune de operatii. Amintim 550 (casuta postala permanent indisponibila), 552 (epuizare spatiu de stocare pe server)

Administratorul serverului SMTP poate configura codurile SMTP transmise clientului pentru diferite situatii (spre exemplu, in cazul filtrarii mesajelor in functie de continutul lor).

9.1.3. Header-e SMTP

Headerele SMTP ofera meta-informatii despre mesaj care sunt accesibile MUA-ului si in functie de care acesta poate lua decizii. Ele se impart in urmatoarele categorii:

- headere standard – sunt cele specificate in standardele ce guverneaza SMTP (cel mai recent este RFC5321 la momentul scrierii acestui material). O parte dintre ele sunt obligatorii si restul sunt optionale
- headere non-standard – sunt headere care nu fac parte din standardele IETF. Numele lor incepe cu **X-** (de unde si numele de “X-Headers”) si contin informatii aditionale, interpretate de catre servere/clienti numai in masura in care software-ul din cele doua parti recunoaste acel header. Amintim aici X-Mailer, care descrie clientul de mail folosit pentru crearea mesajului (rol asemanator cu User-Agent din HTTP)

Dintre headerele standard amintim cateva mai importante:

- **Date** (obligatoriu) – memoreaza data expedierii mesajului SMTP de catre client
- **From** (obligatoriu) – memoreaza adresa e-mail a expeditorului. Atentie! Ceea ce afiseaza MUA-ul este continutul acestui header, NU expeditorul din envelope! Cele doua pot diferi!
- **To** – contine destinatarul mesajului. Ca si in cazul lui From, MUA-ul afiseaza continutul acestui header si nu destinatarul din envelope
- **Cc** – contine lista destinatarilor aditionali. Fiecare destinatar va primi headerul intreg, ceea ce inseamna ca poate vedea care au fost ceilalti
- **Bcc** – contine lista destinatarilor aditionali privati. Acestia nu vor fi vizibili destinatarilor principali (cei definiti cu To sau Cc) si in general nu vor receptiona headerul Bcc, ceea ce inseamna ca nu se pot "vedea" nici intre ei
- **Subject** – contine subiectul mesajului e-mail
- **Received** – fiecare server care primeste un mesaj va adauga propriul sau header Received (pastrandu-le pe eventualele altele). Headerul contine informatii despre sursa mesajului (adresa clientului, numele EHLO cu care s-a prezentat, daca clientul suporta ESMTP etc). Succesiunea headerelor Received ale unui mesaj da posibilitatea determinarii traseului acelui mesaj prin internet

9.1.4. Extensii SMTP

Protocolul SMTP nu continea initial niciun fel de facilitati de autentificare sau criptare. Odata ce s-a constatat ca sunt necesare adaugiri la protocolul de baza, a fost gandita o extensie SMTP care face dezvoltarea protocolului scalabila: ESMTP (Extended SMTP).

ESMTP (descris initial in RFC2821) reprezinta o modificare adusa specificatiei originale a protocolului, care permite adaugarea de extensii si defineste modalitatea in care serverele pot comunica clientilor setul de extensii SMTP suportate. In acest scop a fost introdusa comanda EHLO (care inlocuieste HELO) prin care clientul isi arata disponibilitatea de a folosi extensii SMTP. Serverul raspunde la aceasta comanda cu setul de extensii SMTP suportate.

Nota: *ESMTP a fost initial specificat in RFC1869, fiind apoi integrat in standardul de baza; ultima sa forma la momentul scrierii acestui material este cea din RFC5321.*

Extensiile SMTP definesc noi facilitati ale serverului, care in dese cazuri se materializeaza in noi comenzi SMTP acceptate de catre acesta. Amintim aici cateva dintre extensiile SMTP folosite pe larg:

- **8BITMIME** — initial, SMTP permitea doar transmiterea de caractere ASCII, care foloseau doar 7 biti din cei 8 ai unui octet. Extensia 8-Bit MIME, definita in RFC 1652, permite folosirea caracterelor non-ASCII
- **SMTP-AUTH** — permite autentificarea SMTP in scopul permiterii selective a relay-ului. Extensia este definita in RFC 4954
- **SIZE** — permite anuntarea de catre server a dimensiunii maxime de mesaj admise sau, in lipsa anuntarii ei, posibilitatea ca clientul sa anunte dimensiunea estimata a mesajului inaintea trimiterii acestuia. In acest fel serverul poate respinge mesajul de la bun inceput, fara a mai prelucra trafic inutil. Extensia a fost specificata in RFC 1870
- **STARTTLS** — adauga unui server capabilitatea de a comuta conexiunea cu clientul in mod SSL/TLS la solicitarea clientului. Extensia a fost specificata in RFC 3207 si introduce comanda SMTP cu acelasi nume
- **ENHANCEDSTATUSCODES** – introduce o forma mai granulara a codurilor de raspuns SMTP, sub forma tip.subiect.detaliu (ex: 2.1.43). Extensia a fost specificata in RFC 3463
- **PIPELINING** — extensia imbunatateste eficienta SMTP permitand unui server sa accepte mai multe comenzi SMTP in cadrul aceluiasi pachet TCP. Extensia a fost definita in RFC 2920

9.2. Arhitectura postfix

7.2.1. Programe componente

Ceea ce numim "serverul postfix" este de fapt un ansamblu de executabile care conlucreaza. Administratorul de server trebuie sa fie constient de felul in care se imbina acestea, atat pentru o mai usoara diagnosticare a serverului, cat si pentru corecta sa configurare in scenarii complexe.

Delimitam urmatoarele componente majore ale postfix-ului:

- modulul de server SMTP (**smtpd**) – este cel care implementeaza partea de server din protocolul SMTP, deschizand portul 25 si comunicand cu clientii. Modulul implementeaza si importante functii de filtrare, care permit administratorului sa respinga mesaje in functie de continutul lor in timpul dialogului cu clientul
- modulul care verifica validitatea mesajelor si le corecteaza (**cleanup**) – este cel care se asigura de validitatea unui mesaj inaintea introducerii lui in miezul sensibil al serverului, care este coada de mesaje
- modulul care gestioneaza coada de mesaje (**qmgr**) – este piesa centrala a serverului postfix. Orice mesaj primit, dupa o verificare atenta din partea cleanup, este introdus in coada, de unde serverul va decide daca il livreaza local sau catre un alt server SMTP
- modulul de client SMTP (**smtp**) – este cel folosit atunci cand postfix trebuie sa se conecteze la alte servere SMTP in scopul de a le livra mesaje care nu-i sunt destinate
- modulele de tip MDA (**local**, **virtual**) – sunt cele responsabile cu livrarea mesajelor in casutele postale

Dintre modulele prezentate distingem doua categorii:

- module rezidente in memorie – sunt cele de a caror prezenta depinde corecta functionare a serverului. Exemple sunt *smtpd* si *qmgr*
- module apelate in functie de necesitati – sunt apelate in general de catre cele rezidente pentru anumite operatii necesare asupra mesajului (ex: *cleanup*, *smtp*, *local* etc.)

Pornirea si coordonarea modulelor rezidente in memorie este realizata de catre un executabil "dispecer" numit **master**, la randul sau rezident in memorie si pornit automat la lansarea in executie a serverului. Acesta este configurat folosind fisierul */etc/postfix/master.cf*, unde se specifica ce servicii postfix trebuie pornite si care sunt caracteristicile lor.

7.2.2. "Curgerea" mesajelor prin postfix

Serverul postfix poate primi mesaje din doua surse:

- mesaje provenite din retea, care sunt primite de catre *smtpd*, preluate de catre *cleanup* si, in cazul in care sunt valide si li se permite accesul, pasate mai departe lui *qmgr* care le introduce in coada de mesaje
- mesaje injectate local – administratorul serverului are posibilitatea ca, in scopuri de diagnostic (si nu numai), sa paseze mesaje serverului prin alt mecanism decat SMTP – si anume apelarea unui executabil care va transmite mai apoi mesajul lui *cleanup* si in final lui *qmgr*

Odata ajuns in coada, un mesaj poate pleca catre doua destinatii, in functie de decizia pe care o ia serverul in privinta sa:

- un alt server SMTP, prin intermediul modulului de client smtp al postfix
- o casuta locala, prin intermediul unuia dintre modulele MDA

Mesajele pot fi filtrate in diferite puncte ale curgerii lor prin serverul postfix: la primire - de catre *smtpd*, inaintea introducerii in coada – de catre *cleanup*, dupa iesirea din coada folosind filtre externe etc. Pentru un corect design al filtrelor este necesara intelegerea ordinii operatiilor in cazul fiecarui tip de mesaj prelucrat de server.

9.3. Lucrul cu tabele de informatie externa

9.3.1. Principii

Multi parametri de configurare postfix au ca valoare o baza de date externa pe care postfix o interogheaza pentru a afla informatiile necesare. Sa luam ca exemplu lista de alias-uri, care contine corespondente intre adrese de mail secundare si casute deja existente: pentru fiecare mesaj primit, serverul trebuie sa verifice daca adresa sa destinatie corespunde cu vreunul dintre numele de alias-uri si, in caz afirmativ, sa reorienteze mesajul catre casuta destinatie specificata. Fiecare inregistrare din baza de date cu alias-uri contine de fapt doua campuri – unul care constituie adresa de mail "aparenta" (numele alias-ului) si unul care reprezinta destinatia finala a mesajului. Specificarea acestor perechi mail-casuta ar fi fost incomod de realizat din cadrul fisierului de configurare principal.

Pentru parametri de configurare care au ca valoare astfel de seturi de corespondente (dar nu numai) postfix foloseste asa-numitele “lookup tables” - tabele de informatie externa care se manifesta ca niste baze de date, in sensul in care postfix le poate interoga pentru a extrage informatiile necesare. Tabelele contin inregistrari cu doua campuri – cheie si valoare. Postfix foloseste cheia pentru a afla valoarea sau poate verifica simpla prezenta a cheii. Detaliind, postfix poate efectua urmatoarele operatii:

- folosirea cheii pentru a obtine valoarea corespondenta. Spre exemplu, atunci cand incearca sa livreze mesaje catre o anumita adresa, postfix trebuie sa determine care este locatia casutei din sistemul de fisiere (valoarea) corespunzatoare adresei email destinatie (cheia) pe care o cunoaste deja
- confruntarea unei valori aflate in posesia postfix cu un pattern specificat pe post de cheie; in caz de potrivire se aplica actiunea specificata pe post de valoare. Spre exemplu, putem pune diferite conditii de filtrare a mesajelor in functie de valorile headerelor sau continutului lor (cheie), luand in functie de ele decizia de a le accepta sau respinge (valoare)
- verificarea prezentei unei chei. Spre exemplu, putem memora intr-o tabela externa lista de domenii pentru care serverul accepta email. Tehnic avem de-a face cu o simpla lista (care putea fi specificata de data aceasta in cadrul fisierului principal), insa putem folosi infrastructura de tabele externe pentru a separa listele lungi de fisierul principal. In aceste conditii, al doilea camp al fiecarei inregistrari din tabela poate avea orice valoare, el nemaifiind folosit (ex: serverul trebuie sa afle numai daca un domeniu exista in lista de domenii permise)

Folosirea de tabele externe are avantaje multiple, chiar si atunci cand ele sunt folosite pentru a memora simple liste:

- formatul tabelelor poate fi optimizat pentru o cautare rapida a informatiei
- listele lungi nu mai polueaza fisierul de configurare principal
- informatia din tabelele externe poate fi manipulata si de utilizatori neprivilegiati, degrevand administratorul (sa ne gandim la cazul stocarii alias-urilor, domeniilor, filtrelor etc. intr-o baza de date externa SQL)

Atunci cand dorim ca un parametru de configurare sa foloseasca o tabela de informatie externa vom folosi o sintaxa de forma urmatoare:

```
parametru = tip_tabela:locatie
```

Serverul postfix suporta o mare varietate de tipuri de tabele. Le listam mai jos pe cele mai des folosite:

- **hash** – informatia este memorata intr-un fisier binar sub forma unei tabele de dispersie (hash table), ceea ce face cautarea foarte rapida. Este un tip de tabela potrivit pentru cazul in care cheia este un string obisnuit (nu un regex) si este necesara gasirea rapida a valorii
- **mysql** – informatia este memorata intr-o baza de date externa stocata pe un server MySQL. In cazul unei astfel de tabele este necesara configurarea accesului la server si a interogarilor necesare obtinerii informatiei
- **pgsql** – idem, dar interfateaza cu un server de baze de date postgresql
- **pcre** – informatia este stocata intr-un fisier text; prima “coloana” este un regex, iar cea de-a doua specifica actiunea aplicata mesajului in cazul in care continutul sau corespunde regex-ului. Sunt folosite in general in definirea de filtre sau de reguli de acces

Locatia tabelei este o cale care poate pointa catre:

- fisierul care contine datele (cazul tabelelor de tip hash sau pcre)
- un fisier de configurare care contine informatiile necesare conectarii la baza de date (cazul MySQL, postgresql sau LDAP)

9.3.2. Lucrul cu tabele de tip hash

Construirea unei tabele externe de tip hash presupune doi pasi:

- editarea unui fisier text si introducerea continutului
- generarea unui fisier binar (forma finala a bazei de date, asa cum va fi ea folosita de catre postfix) pe baza fisierului text, folosind utilitarul postmap

Fisierul text va contine cate doua campuri pentru fiecare inregistrare – cheia si valoarea – separate prin unul sau mai multe spatii sau tab-uri. Sa luam exemplul unei baze de date care specifica locatia casutei postale in functie de adresa de mail; fie locatia fisierului */etc/postfix/mailbox_maps*:

```
student@domeniu.ro          domeniu.ro/student/
postmaster@domeniu.ro       domeniu.ro/postmaster
```

Pe baza acestui fisier generam baza de date binara folosind utilitarul *postmap*:

```
[ root@localhost ~]$ postmap hash:/etc/postfix/mailbox_maps
```

Rezultatul este crearea fisierului */etc/postfix/mailbox_maps.db*.

Atentie! *Daca dorim preluarea rapida a modificarilor efectuate intr-o astfel de tabela externa, este necesara fortarea recitirii configurarii folosind* postfix reload*!*

Ce ramane de facut este configurarea noii baze de date ca valoare pentru o directiva de configurare din main.cf. **Atentie! Desi fisierul binar are extensia .db, locatia fisierului - asa cum apare ea in main.cf – nu include extensia!**:

```
virtual_mailbox_maps = hash:/etc/postfix/mailbox_maps
```

9.3.3. Lucrul cu tabele mysql

O tabela externa de tip mysql este de fapt o interfata catre un server de baze de date MySQL. Un parametru de configurare postfix care are ca valoare o tabela mysql va avea forma:

```
parametru = mysql:/cale/catre/fisier.sql.cf
```

Nota: *extensia .sql.cf este alegera autorului; fisierul poate fi denumit in orice mod doreste administratorul de server.*

Fisierul referit contine datele necesare conectarii la serverul de baze de date – adresa acestuia, port, username, parola – si interogarea necesara extragerii datelor. Iata un exemplu care realizeaza aceeasi functie ca cel din paragraful anterior – stabilirea locatiei casutei in functie de adresa de mail – insa de data aceasta toata informatia este memorata in MySQL:

```
hosts = 127.0.0.1                                  # adresa serverului/serverelor interogate
user = user1                                       # username-ul necesar conectarii la MySQL
password = pass1                                   # parola corespunzatoare username-ului
dbname = email                                     # numele bazei de date ce contine tabela
query = SELECT cale FROM casute WHERE email='%s'     # interogarea pentru extragerea caii casutei
```

Pe ultima linie din exemplu, *%s* este secventa care desemneaza valoarea cheii cautate de server (in cazul nostru ea se inlocuieste automat, in momentul interogarii, cu email-ul userului). Interogarea trebuie sa produca o singura coloana, ce reprezinta valoarea corespunzatoare cheii. Tabela MySQL care produce datele necesare exemplului ar putea fi creata cu instructiunea SQL:

```
CREATE TABLE casute (email VARCHAR(200), cale VARCHAR(500));
```

Nota: *in cazul tabelelor MySQL nu se mai genereaza un fisier binar pe baza celui text, deoarece fisierul nu mai contine datele, ci doar detaliile de conectare.*

9.3.4. Lucrul cu tabele pcre

O tabela PCRE contine pe prima sa coloana (cheia) un regex, iar pe cea de-a doua o valoare. Daca input-ul furnizat de postfix corespunde regex-ului, este returnata valoarea corespondenta. Acest tip de tabela este folosita pe scara larga in filtrarea mesajelor, unde input-ul comparat cu regex-ul este chiar continutul mesajului iar valoarea este actiunea aplicata mesajului in caz de potrivire.

O tabela de tip pcre se prezinta sub forma unui fisier text in care fiecare rand contine cele doua campuri:

- cheia, care este un regex perl-compatible. Regex-ul este incadrat intre /..../
- valoarea este dependenta de scopul tabelei

Iata un exemplu de filtru aplicat mesajelor:

```
# in fisierul main.cf:
header_checks = pcre:/etc/postfix/header_checks
```

iar fisierul *header_checks* ar putea arata astfel:

```
# fisierul header_checks
/^From: *bill\.gates@microsoft\.com$/     DISCARD Nu sunteti ATAT de popular...
```

Nota: *nici in cazul tabelelor de tip PCRE nu este necesara generarea unui fisier .db.*

9.3.5. Diagnosticarea tabelelor externe

Putem verifica corecta functionare a unei tabele externe folosind acelasi utilitar *postmap* pentru a o interoga. Cu optiunea *-q* specificam valoarea cheii cautate, iar locatia bazei de date se paseaza ca argument, in aceeasi forma in care apare in fisierul de configurare *main.cf*:

```
[root@localhost ~]# postmap -q student@domeniu.ro hash:/etc/postfix/mailbox_maps
domeniu.ro/student/

[root@localhost ~]# postmap -q student@domeniu.ro mysql:/etc/postfix/mboxmaps.sql.cf
domeniu.ro/student/

[root@localhost ~]# postmap -q "From: bill.gates@microsoft.com" pcre:/etc/postfix/header_checks
DISCARD Nu sunteti ATAT de popular...
```

9.4. Lucrul cu useri virtuali

9.4.1. Principii

Deseori cand configuram un server SMTP dorim sa evitam lucrul cu useri de sistem, din diferite motive:

- adresele de mail au useri de sistem corespondenti. Un atacator care cunoaste acest lucru poate usor deduce conturi existente in vederea unui potential atac
- o configurare nefericita a conturilor si a serverului SSH poate oferi acces de la distanta pe statia in cauza folosind unul dintre conturile create pentru email
- in cazul in care serverul SMTP gazduieste mai multe domenii este probabil sa apara adrese cu acelasi username; in */etc/passwd* username-urile sunt unice, ceea ce impune artificii tehnice din partea administratorului de server
- fisierele *passwd/shadow* nu scaleaza bine pentru numar mare de conturi

Lucrul cu useri virtuali presupune evitarea bazelor de date de sistem si folosirea unor baze de date externe. Acestea pot merge de la simple fisiere text sau binare pana la baze de date SQL sau postgresql.

Oricare ar fi forma aleasa pentru baza de date externa, aceasta trebuie sa furnizeze serverului postfix urmatoarele informatii despre fiecare user:

- UID – user ID-ul utilizatorului
- GID – ID-ul grupului primar al utilizatorului
- email – adresa e-mail a utilizatorului, folosita de catre postfix drept cheie de cautare pentru a determina celelalte detalii ale userului

- calea catre casuta e-mail si formatul acesteia, astfel incat postfix sa poata determina, pe baza adresei e-mail, locul si formatul in care trebuie livrate mesajele
- (optional)parola, daca baza de date este folosita si de catre serverul POP/IMAP sau pentru autentificare SMTP SASL

UID si GID sunt necesare deoarece, la livrarea unui mesaj in casuta, procesul MDA ce efectueaza operatia ruleaza cu UID si GID ale userului destinatie. Permisiunile casutelor e-mail trebuie setate tinand cont de acest aspect.

Toate detaliile per-user de mai sus (UID, GID etc) sunt necesare si serverului POP/IMAP pentru a putea accesa mesajele livrate de postfix. Nu este practic sa duplicam aceasta informatie, de aceea in general baza de date cu utilizatori si detaliile acestora va fi partajata intre postfix si serverul POP/IMAP. Intr-un astfel de caz trebuie sa ne asiguram ca si serverul POP/IMAP suporta tipul de baza de date ales, si sa gandim structura de directoare si permisiunile ansamblului de casute astfel incat sa facilitam accesul ambelor softuri implicate.

9.4.2. Implementare in postfix folosind tabele externe de tip hash

A lucra cu useri virtuali in postfix presupune urmatorii pasi:

- definirea domeniilor pentru care serverul accepta mail ca domenii de tip virtual mailbox
- definirea bazei de date cu corespondente *email->locatie_casuta* pentru toti userii acelor domenii
- definirea bazelor de date cu corespondente *email->UID* si *email->GID*, pentru ca postfix sa stie cu ce UID/GID porneste procesele MDA care realizeaza livrarea mesajelor la casuta
- crearea structurii de fisiere/directoare ce va memora casutele email si stabilirea permisiunilor corecte in aceasta structura

Sa consideram exemplul unui server care gazduieste un singur domeniu virtual, *dom.ro*. Vom folosi tabele externe de tip hash pentru lista de corespondente *email->locatie_casuta* si *email->UID. V*om defini doi utilizatori, cu UID-urile 2001 si 2002; toti userii vor avea acelasi grup, 3000, caz in care nu mai este necesara definirea unei tabele externe. Vom plasa casutele in directorul */var/domenii/dom.ro* (in ideea in care vom dori, poate, pe viitor sa adaugam alte domenii virtuale, carora le vom crea alte subdirectoare sub */var/domenii*). Toate casutele vor fi create in */var/domenii/dom.ro*.

Serverul postfix va efectua livrarea la casuta pornind procese MDA care ruleaza cu UID-ul fiecarui user si cu GID-ul 3000; vom stabili 3000 ca grup proprietar pe directorul */var/domenii/dom.ro* si ii vom acorda permisiuni de scriere, astfel incat postfix sa poata crea automat casutele inauntrul sau la prima livrare.

```
# ******* fisierul main.cf *************
virtual_mailbox_domains = dom.ro                    # domeniile virtuale pt care acceptam mail
virtual_mailbox_base = /var/domenii                 # prefix pentru locatia casutelor de mail
virtual_mailbox_maps = hash:/etc/postfix/vmboxmaps  # corespondenta email->locatie casuta
virtual_uid_maps = hash:/etc/postfix/vuidmaps       # corespondenta email->UID
virtual_gid_maps = static:3000                      # corespondenta email->GID
```

```
# *********** fisierul vmboxmaps ***********

# /-ul final al caii indica format maildir; lipsa sa indica mailbox
user1@dom.ro        dom.ro/user1/
user2@dom.ro        dom.ro/user2/
```

Atentie! Directorul specificat ca valoare a lui virtual_mailbox_base este folosit ca prefix pentru toate caile de casute din virtual_mailbox_maps, chiar daca acestea incep cu / !

```
# ********* fisierul vuidmaps ***************
user1@dom.ro        2001
user2@dom.ro        2002
```

Testam corecta functionare prin injectarea locala a unui mesaj destinat lui user1@dom.ro si verificarea corectei sale livrari in directorul */var/domenii/dom.ro/user1/new*.

9.4.3. Alternativa MySQL

Daca dorim memorarea detaliilor userilor intr-o baza de date MySQL, principiile enuntate anterior se pastreaza dar trebuie operate cateva modificari:

- setup MySQL: crearea bazei de date, crearea si popularea tabelei cu useri si parole, crearea userului pentru acces MySQL si acordarea privilegiilor necesare

```
CREATE SCHEMA email; USE email;
CREATE TABLE useri( email VARCHAR(100), pass VARCHAR(100), uid INT, gid INT, casuta VARCHAR(200));
INSERT INTO useri VALUES('user1@dom.ro','pass',2001,3000,'dom.ro/user1/');
INSERT INTO useri VALUES('user2@dom.ro','pass',2002,3000,'dom.ro/user2/');
GRANT SELECT ON email.* TO 'mailuser'@'127.0.0.1' IDENTIFIED BY 'pass';
FLUSH PRIVILEGES;
```

- modificarea lui *main.cf* pentru a folosi baze de date de tip mysql:

```
virtual_mailbox_maps = mysql:/etc/postfix/vmboxmaps.sql.cf
virtual_uid_maps = mysql:/etc/postfix/vuidmaps.sql.cf
```

- editarea fisierelor de configurare pentru tabelele SQL (exemplificam doar fisierul pentru casute):

```
# ********** fisierul vmboxmaps.sql.cf *************
hosts = 127.0.0.1
user = mailuser
password = pass
dbname = email
query = SELECT casuta FROM useri WHERE email = '%s'
# ...iar pentru UID-uri am fi avut:
# query = SELECT uid FROM useri WHERE email = '%s'
```

Testarea se efectueaza prin interogare cu postmap si apoi prin injectarea unui mesaj local, urmarind logurile.

9.4.4. Configurare dovecot

Pasul urmator este configurarea serverului dovecot astfel incat sa raspunda urmatoarelor cerinte:

- serverul trebuie sa cunoasca locatia si formatul corect pentru casutele e-mail, asa cum le-a creat anterior postfix
- in plus fata de postfix, dovecot trebuie configurat sa autentifice utilizatorii pentru a le acorda acces la casuta proprie

Configurarea locatiei casutelor de mail se realizeaza folosind directiva *mail_location*, care primeste o valoare de forma *format:cale*. Formatul poate fi *mailbox* sau *maildir*, iar calea poate contine secvente de forma *%litera* in scopul automatizarii accesului la casutele de mail, dupa cum urmeaza:

- %u – username-ul folosit la autentificare (in cazul nostru intreaga adresa email, *user1@dom.ro*)
- %n – partea stanga a adresei email (*user1*)
- %d – partea dreapta a adresei e-mail (*dom.ro*)

Data fiind structura pe care am gandit-o pentru casutele de mail, directiva *mail_location* necesara va fi:

```
# ************** fisierul /dovecot.conf ****************
mail_location = maildir:/var/domenii/%d/%n
```

Acest mod de configurare lasa posibilitatile deschise pentru adaugarea unor viitoare domenii virtuale.

Pentru configurarea autentificarii este necesara activarea bazelor de date cu useri si parole de tip sql:

```
# ************** fisierul /etc/dovecot.conf ****************
passdb sql {
      args = /etc/dovecot-sql.conf
}
userdb sql{
      args = /etc/dovecot-sql.conf
}
```

Ca si in cazul postfix, fisierul *dovecot-sql.conf* contine datele necesare conectarii la serverul de baze de date si obtinerii informatiei:

```
# ************** fisierul /etc/dovecot-sql.conf ****************
driver = mysql                                          # alternative: pgsql, sqlite
connect = host=127.0.0.1 user=mail password=pass dbname=email
default_pass_scheme = PLAIN                             # alternative: PLAIN-MD5, MD5-CRYPT
password_query = SELECT email AS user, pass AS password FROM useri WHERE email='%u'
user_query = SELECT email AS user, uid, gid, casuta as home FROM useri WHERE email = '%u'
```

Interogarea pentru parola trebuie sa produca exact un rand si care trebuie sa contina campurile denumite *user* si *password* (avand exact aceste nume!). Interogarea pentru extragerea detaliilor userilor trebuie sa returneze un rand care contine cel putin campurile *uid,gid,home,mail* (cu exact aceste nume). In cadrul interogarilor pot fi folosite aceleasi secvente speciale ca in cazul lui *mail_location*: in exemplu, %u este username-ul furnizat la autentificare.

Nota: *pentru a afla rapid sintaxa necesara poate fi consultat fisierul /usr/share/doc/dovecot/dovecot-sql-example.conf.*

9.5. Masuri de securitate aplicabile unui server SMTP

Descriem pe scurt principalele operatii posibile in cadrul unui server SMTP, urmand a le detalia in paragrafele urmatoare:

- securizarea serverului folosind SSL/TLS. Asigura autentificarea uni- sau bilaterala si criptarea comunicatiilor acestuia cu clientii
- controlul accesului in functie de continutul mesajului. Administratorul poate defini filtre interne sau externe pe baza carora mesajele spam sau potential periculoase sa fie respinse sau plasate in carantina
- controlul relay-ului. Derivat din controlul accesului, acesta permite administratorului sa specifice clientii (sau mesajele!) pentru care serverul accepta mesajul dar nu il livreaza local – cu alte cuvinte, joaca rolul de intermediar. Principala modalitate de control al relay-ului presupune autentificare SMTP, realizata de obicei prin intermediul SASL (vezi capitolul dedicat)

9.6. Securizare cu SSL/TLS

9.6.1. Concepte

Protocolul SMTP, in specificatia sa originara, functiona fara criptare. Aceasta facilitate i-a fost adaugata sub forma extensiei ESMTP STARTTLS care introduce si comanda SMTP cu acelasi nume. Comanda permite clientului ca, din cadrul unui dialog necriptat cu un server SMTP, sa initieze stabilirea unui canal de comunicatie criptat SSL/TLS.

Istoric vorbind, serverele SMTP care isi cripteaza comunicatiile pot functiona in 3 moduri:

- fara criptare – serverul asculta pe portul SMTP clasic (25) si nu prezinta facilitatea STARTTLS ca raspuns la comanda EHLO
- in mod SSL – serverul functioneaza exclusiv criptat, pe un port separat (in general 465). Orice client care se conecteaza trebuie sa inceapa direct cu handshake-ul SSL, comenzile SMTP circuland abia dupa stabilirea canalului securizat. In aceste conditii nu este nevoie de extensia STARTTLS
- in mod TLS – serverul asculta pe portul 25, clientii se pot conecta la el fara criptare. Serverul prezinta capabilitatea STARTTLS ca raspuns la EHLO. Clientul poate initia o conexiune criptata folosind comanda STARTTLS, restul comunicatiei desfasurandu-se in continuare pe acelasi port 25 dar cu criptare

9.6.2. Configurarea serverului pentru a folosi TLS

Parametrii de configurare necesari pentru a implementa TLS in postfix sunt:

```
smtpd_tls_security_level = may    # sau in versiunile mai vechi de postfix, smtpd_use_tls = yes
                                  # Vezi mai jos efectul lui smtpd_tls_security_level = encrypt
smtpd_tls_cert_file = /etc/postfix/cert.pem
smtpd_tls_key_file = /etc/postfix/key.pem
smtpd_tls_loglevel = 2            # cresterea nivelului de logging TLS; valori posibile 0...4
```

Alti parametri TLS utili:

- **smtpd_tls_security_level = encrypt** – folosit pentru a interzice complet conexiunile necriptate (neindicat pentru un server public)
- **smtpd_tls_ask_ccert = yes** – solicita clientului certificat digital, fara a-l face insa obligatoriu
- **smtpd_tls_req_ccert = yes** – obliga clientul sa furnizeze certificat (stabilirea sesiunii TLS esueaza in lipsa lui)
- **smtpd_tls_auth_only = yes** – autentificarea SMTP nu va fi oferita/permisa decat atunci cand conexiunea este criptata TLS

Testarea corectei functionari poate fi efectuata si cu ajutorul lui *openssl*, a carui comanda *s_client* poate emula extensia SSL/TLS pentru cateva protocoale majore, printre care si cele e-mail:

```
root# openssl s_client -starttls stmp -CApath /cale/catre/fisierCAtrusted -connect localhost:25
# optiunea -starttls accepta ca argumente smtp, pop, imap sau ftp
```

9.7. Controlul accesului

9.7.1. Concepte

Controlul acesului la server presupune a dicta, pentru fiecare mesaj, ce actiune i se aplica – de preferinta cat mai devreme in curgerea mesajului prin server, astfel incat el sa nu consume resurse in mod inutil.

Administratorul postfix poate filtra mesaje folosind doua tipuri de mecanisme:

- **interne** (built-in) – sunt filtre rapide dar relativ simple, disponibile ca parte a postfix. Sunt nepotrivite pentru verificari de anvergura (antivirus, antispam) dar pot fi folosite pentru a respinge mesaje pe criterii simple (ex: portiuni din headere sau continut)
- **externe** – postfix trimite mesajul catre aplicatii externe si ele distrug mesajul, il introduc in carantina sau intorc un verdict inapoi catre server. Postfix merge pe principiul implementarii mecanismelor strict necesare, lasand facilitatile complexe in grija altor aplicatii specializate, care ofera servicii de calitate superioara

Dupa cum s-a discutat anterior, un mesaj prelucrat de postfix trece printr-o anumita succesiune de operatii. Filtrarea se poate efectua in cateva momente cheie:

- in timpul primirii mesajului de la client. Filtrarea de acest fel este implementata de modulul *smtpd* prin intermediul directivelor de configurare *smtpd_*_restrictions*. Putem folosi aici filtre built-in sau externe, cu mentiunea ca cele externe este posibil sa consume prea mult timp si clientul sa faca timeout. Avantajul folosirii filtrelor de acest fel este ca mesajul poate fi respins imediat in caz de nevoie, fara a mai genera un bounce
- dupa primirea mesajului dar inainte de introducerea lui in coada (filtre *before-queue*) – implementate de catre modulul de cleanup, prin intermediul directivelor **_checks*. Putem de asemenea folosi filtre built-in sau externe
- dupa introducerea in coada (filtre *after-queue*) – sunt implementate prin directiva *smtpd_proxy_filter*, care face modulul de client SMTP sa livreze mesajele catre o aplicatie externa, care apoi le reinjecteaza in postfix

Orice tip de filtru am folosi, verdictul dat de acesta in privinta mesajului poate tine cont de:

- caracteristici ale clientului (IP, nume DNS)
- informatii din envelope
- informatii din headerele SMTP ale mesajului sau ale mesajelor atasate la acesta

- continutul mesajului

9.7.2. Mecanisme interne

9.7.2.1. Posibilitati

Filtrele interne postfix pot actiona in doua locuri/momente:

- in timpul comunicatiei cu clientul. Fiind aplicate cat timp clientul este inca conectat la server, acesta din urma are posibilitatea aplicarii unei actiuni mesajului inainte ca el sa patrunda in coada si sa mobilizeze resurse mai bogate. Pe de alta parte, filtrele trebuie sa actioneze rapid, astfel incat clientul sa nu faca timeout. Aceasta categorie de filtre este aplicata de modulul *smtpd*
- dupa ce mesajul a fost acceptat, inainte de a fi introdus in coada. Sunt aplicate de modulul *cleanup*, dupa ce *smtpd* a acceptat mesajul dar inainte de introducerea sa in coada de mesaje

9.7.2.2. Filtrare in timpul dialogului SMTP

Cele mai timpurii filtre pot fi aplicate in timp ce clientul este conectat la serverul nostru si ne trimite, comanda cu comanda, mesajul e-mail. Putem alege filtrele dorite la nivel de fiecare comanda SMTP trimisa de client, bazandu-ne pe informatiile disponibile pana in acel moment. Folosim in acest scop directivele *smtpd_*_restrictions*, denumite in functie de stadiul dialogului SMTP in care apare informatia necesara lor din partea clientului. Iata-le in ordinea in care sunt aplicate:

- **smtpd_client_restrictions** – restrictionare clienti in functie de adresa IP sau nume DNS (disponibile imediat dupa ce clientul s-a conectat)
- **smtpd_helo_restrictions** – restrictii aplicabile odata ce clientul a dat comanda HELO/EHLO. Le cuprind pe cele deja disponibile de la primul pas si adauga restrictii legate de validitatea hostname-ului furnizat de client in comanda HELO/EHLO
- **smtpd_sender_restrictions** – restrictii aplicabile dupa ce clientul a furnizat expeditorul din envelope (comanda SMTP *MAIL FROM:*). Cuprind toate restrictiile posibile pana acum si adauga conditii legate de validitatea adresei e-mail a expeditorului (sa fie completa, sa existe inregistrari A sau MX corespunzatoare domeniului ei DNS etc)
- **smtpd_recipient_restrictions** – restrictii aplicabile dupa ce clientul a furnizat destinatarul din envelope (comanda SMTP *RCPT TO:*). Aceeasi functionare ca la pasul anterior dar cu aplicare pentru adresa destinatarului
- **smtpd_data_restrictions** – restrictii aplicabile dupa ce clientul a initiat comanda SMTP *DATA*

In general se foloseste destul de mult doar *smtpd_recipient_restrictions*, din doua motive:

- este suficient de devreme in cadrul prelucrarii mesajului pentru ca acesta sa fie respins. Mesajul nu apuca sa patrunda in coada de mesaje, unde ar mobiliza in mod inutil mult mai multe resurse din partea serverului
- este suficient de tarziu in cadrul dialogului SMTP cu clientul incat sa avem la dispozitie o intreaga serie de parametri colectati pana atunci pentru a lua decizia asupra mesajului

Printre restrictiile posibile se numara si cele de tip RBL (Real-Time Blackhole List). RBL-urile sunt colectii de adrese cunoscute ca surse de spam, publicate prin intermediul DNS. Exista multiple astfel de liste disponibile pe internet. Un server SMTP care doreste sa determine daca adresa unui client se regaseste intr-un RBL va efectua o simpla interogare DNS (intr-un format ce aminteste de rezolutia DNS inversa). Un server poate fi configurat sa se foloseasca de mai multe RBL-uri (vezi si exemplul de mai jos).

Iata un exemplu ce cuprinde cele mai des intalnite restrictii aplicate mesajelor in tipul dialogului cu clientul:

```
smtpd_recipient_restrictions =
      reject_invalid_helo_hostname,
      reject_non_fqdn_helo_hostname,    # a se folosi cu grija, unele servere configurate prost nu
                                        # trimit hostname complet
      reject_unauth_pipelining,
      reject_non_fqdn_sender,
      reject_unknown_sender_domain,     # domeniul expeditorului nu are A sau MX valid
```

```
        reject_non_fqdn_recipient,
        reject_unknown_recipient_domain,  # domeniul destinatarului nu are A sau MX valid
        permit_mynetworks,                # acces de la clientii trusted
        permit_sasl_authenticated,        # acces pentru clientii autentificati SMTP
        reject_unauth_destination,        # Interzis pentru domeniile in cazul carora postfix nu
                                          # este destinatia finala si nici nu permite relay-ul
        check_client_access hash:/etc/postfix/maps/access_client,
        check_helo_access hash:/etc/postfix/maps/access_helo,
        check_sender_access hash:/etc/postfix/maps/access_sender,
        check_recipient_access hash:/etc/postfix/maps/access_recipient,
        reject_rhsbl_client blackhole.securitysage.com,       # Verificarile blacklist trebuie
        reject_rhsbl_sender blackhole.securitysage.com,       # plasate la final, pentru ca
        reject_rbl_client relays.ordb.org,                    # presupun interogari DNS si deci
        reject_rbl_client blackholes.easynet.nl,              # sunt costisitoare; verificarea
        reject_rbl_client cbl.abuseat.org,                    # ajunge la ele numai in cazul in care
        reject_rbl_client proxies.blackholes.wirehub.net,     # mesajul a trecut de restul de
        reject_rbl_client bl.spamcop.net,                     # conditii
        reject_rbl_client sbl.spamhaus.org,
        reject_rbl_client opm.blitzed.org,
        reject_rbl_client dnsbl.njabl.org,
        reject_rbl_client list.dsbl.org,
        reject_rbl_client multihop.dsbl.org,
        permit
```

9.7.2.3. Filtrare inaintea introducerii in coada

Un mesaj acceptat de catre daemonul smtpd este pasat mai departe modulului cleanup, care implementeaza la randul sau o serie de filtre built-in. Acestea sunt configurabile prin intermediul directivelor **_checks* dupa cum urmeaza:

- **header_checks** – filtrare in functie de headerele mesajului
- **body_checks** – filtrare in functie de continutul mesajului (“body”). In cazul mesajelor MIME multipart, aceste filtre se aplica fiecarei portiuni aflate intre doua seturi de headere MIME!
- **mime_header_checks** – filtrare in functie de headerele MIME ale mesajului
- **nested_header_checks** – filtrare in functie de headere din mesaje email atasate la mesajul curent

Fiecare dintre aceste directive are ca valoare un set de verificari aplicate mesajului, sub forma unei tabele externe de tip pcre:

```
header_checks = pcre:/etc/postfix/header_checks
```

In fisierul referit, prima pozitie a fiecarei linii reprezinta un regex Perl-Compatible, iar pozitia a doua reprezinta actiunea aplicata mesajului impreuna cu un eventual mesaj de eroare trimis clientului la aplicare. Dintre actiunile posibile amintim:

- REJECT [text] – respinge explicit mesajul afisand eventualul text specificat
- DISCARD – semnalizeaza clientului acceptarea mesajului dar il distruge imediat dupa acceptare
- REDIRECT alta_adresa – trimite mesajul catre o alta adresa (util pentru a plasa in carantina mesaje suspecte)
- WARN [text] – consemneaza in loguri textul specificat si continua evaluarea mesajului
- FILTER – marcheaza mesajul pentru a fi trimis catre un filtru extern (definit in master.cf) dupa includerea in coada

```
# fisierul header_checks
/^Subject.*Rolex/   REJECT Mesajul este spam - si oricum am deja Rolex :-)
```

Nota: *header_checks si body_checks nu decodeaza attachment-uri si nu dezarhiveaza, ci doar analizeaza string-uri asa cum apar ele in mesaj. Ele sunt gandite numai pentru oprirea flood-urilor generate de virusi si alte asemenea actiuni simple, nu pentru verificari elaborate de continut (ex: detectie virusi in executabile sau arhive).*

9.7.3. Filtre externe

9.7.3.1. Abordari

Filtrele externe reprezinta programe separate, specializate, de care se foloseste postfix pentru a prelucra mesajele. Exemple de filtre sunt analizoarele de continut (care pot scana mesajul determinand, pe baza unor criterii complexe, daca este spam, daca contine virusi etc) sau programele care adauga diferite semnaturi digitale mesajelor in scopul autentificarii lor la primire de catre alte servere.

Exista urmatoarele momente din "curgerea" unui mesaj prin postfix cand il putem trimite catre un filtru extern:

- cat timp clientul este inca conectat ("before-queue content filtering"). Serverul paseaza mesajul aplicatiei externe, care il analizeaza si produce raspunsuri SMTP care sunt trimise inapoi catre serverul SMTP si de acolo catre client. Trebuie folosite cu prudenta, deoarece exista riscul ca analiza continutului sa dureze prea mult si clientul sa faca timeout
- dupa ce mesajul e introdus in coada ("after-queue content filtering"). Mesajul e initial acceptat, clientului i se semnlizeaza acest lucru prin codul SMTP corespunzator, mesajul e introdus in coada, dupa care e pasat unei aplicatii externe. Aceasta poate distruge mesajul, il poate livra "personal" sau il poate reinjecta in coada

In ambele cazuri, daca filtrul extern a incheiat prelucrarea si a decis ca mesajul va fi acceptat, este necesara re-introducerea mesajului in postfix, dar pe o alta "intrare" - in general una de incredere, care nu mai aplica la randul sau alte filtre (ca sa nu se creeze o bucla infinita). Aceasta "intrare" ia in general forma unui al doilea daemon smtpd, configurat din cadrul fisierului *master.cf*.

9.7.3.2. Filtrare externa before-queue

Filtrarea externa before-queue se realizeaza trimitand mesajul, cat timp clientul este conectat, catre o aplicatie externa. Aplicatia in cauza trebuie definita ca serviciu postfix in fisierul master.cf. Aplicatia trebuie la randul sau configurata sa re-injecteze mesajele acceptate inapoi in postfix, in care scop este necesara pornirea unui alt doilea daemon smtpd.

Pasii necesari implementarii sunt urmatorii:

- se editeaza fisierul master.cf si se adauga doua servicii:
  - unul corespunzator aplicatiei de filtrare externa, in care se specifica numele serviciului, modalitatea de conectare la aplicatia externa si diversii parametrii ai operatiunii
  - altul corespunzator daemonului smtpd care va primi inapoi de la aplicatia externa mesajele acceptate. Acest daemon ruleaza doar local (in general asculta aor pe interfata de loopback) si accepta mesajele fara a le mai aplica restrictii (ele fiind deja validate de catre filtrul extern)
- se specifica faptul ca mesajele trebuie redirectionate de catre smtpd catre filtrul extern definit anterior in main.cf, folosind directiva *smtpd_proxy_filter*
- se configureaza aplicatia externa:
  - sa asculte pe adresa si portul specificate in directiva *smtpd_proxy_filter* de mai devreme
  - sa reinjecteze mesajele inapoi in postfix folosindu-se de cel de-al doilea daemon smtpd

```
# **** fisierul main.cf ********
smtpd_proxy_filter=127.0.0.1:10025
smtpd_client_connection_count_limit=10
```

```
# **** fisierul master.cf ************
# daemonul smtpd secund care primeste mesajele reinjectate de catre filtrul extern
127.0.0.1:10026 inet n  -       n       -       -       smtpd
        -o smtpd_authorized_xforward_hosts=127.0.0.0/8
        -o smtpd_client_restrictions=
        -o smtpd_helo_restrictions=
        -o smtpd_sender_restrictions=
        -o smtpd_recipient_restrictions=permit_mynetworks,reject
        -o smtpd_data_restrictions=
        -o mynetworks=127.0.0.0/8
```

```
        -o receive_override_options=no_unknown_recipient_checks
```

9.7.3.3. Filtrare externa after-queue

Filtrarea externa after-queue se realizeaza setand explicit filtrul aplicat intregului continut ce traverseaza postfix. Mesajele patrund in coada de mesaje si abia dupa aceea sunt trimise catre filtrul extern. Acesta din urma este definit sub forma unui serviciu in master.cf si este referit din main.cf folosind directiva content_filter.

Pasii necesari implementarii sunt:

- se editeaza fisierul master.cf si se adauga urmatoarele servicii:
  - unul corespunzator filtrului extern
  - altul corespunzator daemonului smtpd care va receptiona mesajele acceptate returnate de filtrul extern
- se editeaza main.cf si se specifica filtrul extern aplicat mesajelor folosind directiva content_filter
- se configureaza aplicatia externa pentru a re-injecta corect mesajele in postfix folosindu-se de al doilea daemon smtpd

Pentru un exemplu de implementare vezi sectiunea urmatoare.

9.7.3.4. Aplicatie: interfatarea cu programe antivirus si anti-spam

9.7.3.4.1. Amavisd-new. Principii de functionare

Amavisd-new este o aplicatie scrisa in Perl care permite interfatarea postfix (si nu numai) cu felurite programe de tip antivirus sau antispam. Putem defini amavisd-new ca filtru extern (configurand si traseul de re-injectare a mesajelor) iar acesta va verifica mesajul folosindu-se de programele antivirus/antispam suportate si disponibile pe masina gazda.

Amavisd-new functioneaza in general ca filtru after-queue, deoarece verificarile efectuate sunt costisitoare ca timp (aceluiasi mesaj ii pot fi aplicati mai multi antivirusi si mai multe softuri antispam).

9.7.3.4.2. Instalare si configurare aplicatii necesare

Vom instala si configura amavisd-new impreuna cu un antivirus free (clamav) si un program antispam (spamassassin). Etapele necesare sunt:

- instalare amavisd-new si programe conexe. Numele pachetelor necesare depind de distributia folosita. In Mandriva, ele sunt *amavisd-new*, *spamassasin* si *clamd*
- configurare postfix
  - definirea serviciilor necesare in master.cf: cel pentru amavis si cel pentru daemonul smtpd secund
  - configurarea lui amavis ca content_filter
- actualizarea bazei de date cu semnaturi de virusi

```
root # urpmi amavisd-new spamassasin clamd
```

```
# ******* fisierul master.cf *******
# adaugam/decomentam (daca este cazul) cele doua servicii:

#daemonul smtpd secund
127.0.0.1:10026        inet    n       -       n       -       -       smtpd
  -o content_filter=
  -o smtpd_restriction_classes=
  -o smtpd_client_restrictions=permit_mynetworks,reject
  -o smtpd_helo_restrictions=
  -o smtpd_sender_restrictions=
  -o smtpd_end_of_data_restrictions=
  -o smtpd_etrn_restrictions=
  -o smtpd_data_restrictions=
```

```
  -o smtpd_delay_reject=no
  -o smtpd_recipient_restrictions=permit_mynetworks,reject
  -o mynetworks=127.0.0.0/8
  -o smtpd_authorized_xforward_hosts=127.0.0.0/8
  -o strict_rfc821_envelopes=yes
  -o smtpd_error_sleep_time=0
  -o smtpd_soft_error_limit=1001
  -o smtpd_hard_error_limit=1000
  -o receive_override_options=no_unknown_recipient_checks,no_header_body_checks

# filtrul extern amavisd-new
amavis-filter    unix    -       -        n       -       -       smtp
  -o smtp_data_done_timeout=1200
  -o smtp_send_xforward_command=yes
  -o max_use=20
```

Atentie! Testarea configuratiei nu se va face cu mesaje injectate local, deoarece acestea nu trec prin primul daemon SMTP – singurul configurat sa le trimita catre filtrul extern! Putem folosi in acest scop utilitarul mailx care se poate conecta folosind SMTP si, in plus, suporta adaugarea de attachment-uri:

```
echo "test email" | mailx -S smtp=127.0.0.1 user1@dom.ro
```

Pentru a verifica corecta respingere a unui mesaj care contine un virus, se va descarca fisierul http://www.eicar.org/download/eicar.com care contine o semnatura recunoscuta ca virus de catre majoritatea programelor antivirus. Tehnic, acest fisier nu este un virus, ci un executabil DOS inofensiv care prin conventie este raportat ca virus tocmai pentru diagnosticarea aplicatiilor antivirus. Daca atasam acest fisier la un mesaj email si il trimitem propriului server, trebuie sa gasim in loguri raportul lui amavis despre blocarea mesajului:

```
$ wget http://www.eicar.org/download/eicar.com
$ echo "test email" | mailx -S smtp=127.0.0.1 -a eicar.com user1@dom.ro
$ tail -20 /var/log/mail/info.log|grep Blocked
Aug 10 15:38:25 localhost amavis[28735]: (28735-02) Blocked INFECTED (Eicar-Test-Signature),
MYNETS LOCAL [127.0.0.1] [127.0.0.1] <root@localhost.localdomain> -> <user1@dom.ro>, quarantine:
virus-xPlTXqXHlKI1, Message-ID: <4c614841.pHPYQFVawa8PQb88%root@localhost>, mail_id: xPlTXqXHlKI1,
Hits: -, size: 1070, 142 ms
```

9.8. Controlul relay-ului

9.8.1. Mecanisme

Spunem ca un server joaca rolul de relay atunci cand el accepta mesaje pentru care nu este destinatia finala. Acest lucru se poate intampla fie cand serverul a fost configurat explicit sa actioneze ca relay pentru unul sau mai multe domenii destinatie, sau in cazul in care serverul permite unui set de clienti “trusted” sa trimita mesaje prin intermediul sau. Din acest punct de vedere, controlul relay-ului este de fapt o forma de control al accesului mesajelor in coada de mesaje, aplicata la nivel de daemon smtpd.

De-a lungul timpului au fost folosite diferite modalitati de control al accesului:

- in functie de adresa IP a clientului – era singura modalitate disponibila initial. Odata cu modbilitatea clientilor a parut necesitatea ca un acelasi utilizator sa poate livra mail prin intermediul aceluiasi server SMTP indiferent de reteaua in care se afla (si deci de adresa sa IP momentana)
- in functie de domeniul destinatie al mesajului – serverul SMTP poate fi configurat explicit sa joace rolul de relay pentru unul sau mai multe domenii (spre exemplu, atunci cand functioneaza ca backup MX)
- in functie de proprietatea clientului de a se fi autentificat SMTP inaintea transmiterii mesajului – acest lucru este posibil gratie extensiei SMTP AUTH. Este cea mai folosita modalitate de control al relay-ului in caest moment
- in functie de detalii din certificatul clientului, daca conexiunea este securizata TLS si clientul a furnizat certificat
- in functie de proprietatea clientului de a se fi autentificat POP sau IMAP inaintea trasmiterii mesajului. Solutiile de tipul POP-BEFORE-SMTP sau IMAP-BEFORE-SMTP reprezinta artificii tehnice care nu mai sunt necesare,

nu sunt usor de configurat (presupun o buna colaborare intre serverul SMTP si cel POP/IMAP) si nici nu ofera un grad de siguranta foarte ridicat

9.8.2. Control relay in functie de IP client

Controlul relay-ului in functie de adresa IP sursa sau subnet-ul din care face parte clientul se realizeaza in urmatorii pasi:

- definirea setului de adrese/subnet-uri "trusted", folosind directivele *mynetworks* sau *mynetworks_style*:
  - **mynetworks_style** permite definirea rapida a setului de adrese trusted, sub 3 forme:
    - host – este permis relay-ul numai pentru adresele locale ale statiei
    - subnet – toate subretelele din care face parte statia server sunt trusted
    - class – toate adresele din reteaua de clasa respectiva (A, B sau C) sunt trusted
  - **mynetworks** permite definirea exacta a setului de IP-uri/subnet-uri trusted, sub forma unei liste ale carei elemente sunt separate prin spatiu, sau ca tabela externa
- permiterea accesului pentru clientii trusted folosind optiunea *permit_mynetworks* ca parte a valorii lui *smtpd_recipient_restrictions*

```
# **** fisierul main.cf *********
mynetworks = 10.0.0.0/24 192.168.0.0/28
smtpd_recipient_restrictions = .......,permit_mynetworks,.............
```

9.8.3. Autentificare SMTP

9.8.3.1. Principii

Autentificarea SMTP reprezinta o extensie care permite clientului sa se autentifice inaintea trimiterii unui mesaj. In functie de calitatea sa de client autentificat corect serverul il poate trata diferit fata de alti clienti (ex: ii poate permite relay-ul chiar daca IP-ul sau nu se regaseste in lista de adrese trusted).

Un server care suporta SMTP AUTH va anunta acest lucru ca raspuns la comanda EHLO a clientului. Atentie! In scopul cresterii securitatii, serverul poate fi configurat sa prezinte aceasta capabilitate numai dupa ce clientul a stabilit un canal de comunicatie securizat SSL/TLS cu serverul! De asemenea, lista de mecanisme de autentificare suportate poate diferi dupa intrarea in TLS.

Nota: *putem impune ca autentificarea sa fie disponibila numai printr-o conexiune criptata cu directiva smtpd_tls_auth_only = yes.*

In scopul adaugarii capabilitatii de autentificare pentru protocoale care functionau la origini clear-text a fost gandit SASL (Simple Authentication and Security Layer) – o specificatie care permite unui astfel de protocol autentificarea decuplandu-l de aspectele low-level ale memorarii informatiilor de autentificare (useri, parole etc). Serverul SMTP interfateaza cu un soft SASL folosind un API prestabilit, iar softul in cauza ii ofera accesul la multiple mecanisme si back-end-uri de autentificare.

Exista cel putin doua implementari pe care postfix le poate folosi:

- **Cyrus SASL** – implementarea SASL cea mai cunoscuta, insa relativ greoaie
- **Dovecot SASL** – implementare recenta integrata in serverul dovecot, si care va fi folosita in continuarea materialului

9.8.3.2. Implementare

Configurarea postfix pentru autentificare SASL presupune urmatoarele etape:

- verificare ca postfix a fost compilat cu suport de dovecot SASL

```
$ postconf -a
cyrus
dovecot
```

- adaugare configurare SASL dovecot in *main.cf*:

```
# activare autentificare SASL; serverul va prezenta clientilor posibilitatea ca raspuns la EHLO
smtpd_sasl_auth_enable = yes

# solutia de autentificare SASL folosita: dovecot sau cyrus
smtpd_sasl_type = dovecot

# calea catre socket-ul deschis de daemon-ul SASL al lui dovecot pentru a comunica cu postfix
smtpd_sasl_path = private/auth

# permitem relay-ul pentru clientii autentificati SASL (vezi sectiunea de control acces)
smtpd_recipient_restrictions= ......., permit_sasl_authenticated, ..............
# ...sau pentru a permite accesul doar clientilor autentificati SASL:
smtpd_recipient_restrictions=permit_sasl_authenticated,reject
```

Configurarea dovecot presupune specificarea bazelor de date cu useri si parole si activarea socket-ului folosit pentru autentificare SASL de catre postfix:

```
auth default {
  mechanisms = plain digest-md5
  passdb sql {
     args = /etc/dovecot/dovecot-sql.conf
  }
  userdb sql {
     args = /etc/dovecot/dovecot-sql.conf
  }
  user = root
  socket listen {
    client {
      path = /var/spool/postfix/private/auth
      mode = 0660
      user = postfix       # important, pentru ca postfix sa poata citi/scrie in socket!
      group = postfix
    }
  }
}
```

9.8.4. Control relay/acces pe baza de certificat client

Certificartl prezentat de catre client este o forma de autentificare si de aceea decizia acceptarii accesului/relay-ului poate fi luata si in functie de validitatea certificatului sau de informatiile cuprinse in cadrul acestuia. Masurile necesare implementarii sunt descrise in continuare:

- solicitam clientului sa prezinte certificat – cu doua grade de obligativitate:
  - smtpd_tls_ask_ccert = yes – clientului i se solicita certificat insa are optiunea de a nu furniza unul
  - smtpd_tls_req_ccert = yes – clientul este obligat sa furnizeze certificat, in caz contrar handshake-ul TSL esueaza. *A se folosi cu atentie, deoarece clientii sunt deseori alte servere care incearca sa ne livreze mail!*
- folosim una dintre urmatoarele abordari, in functie de necesitati:
  - permitem accesul tuturor clientilor care au un certificat valid si trusted (conform listei de CA-uri trusted configurate pe server). Nu se impun restrictii suplimentare certificatului clientului. Aceasta abordare are sens atunci cand CA-urile trusted sunt administrarea noastra, astfel incat sa putem avea incredere in orice client pentru care garanteaza acestea.

```
smtpd_*_restrictions = ..............,permit_tls_all_clientcerts,...........
```

  - permitem accesul numai clientilor care au fingerprint-ul certificatelor intr-o baza de date creata de noi (si in plus prezinta un certificat valid!). Este o solutie mai granulara comparativ cu cea precedenta insa presupune manopera suplimentara: administratorul trebuie sa genereze niste fingerprint-uri (amprente) ale certificatelor clientilor trusted si sa le includa intr-o baza de date folosita pentru autorizare

```
smtpd_*_restrictions = permit_tls_clientcerts
relay_clientcerts = tip_tabela:/cale/catre/tabela/externa
```

Cea de-a doua directiva stabileste tabela cu fingerprint-uri de certificate; prima coloana a tabelei contine fingerprint-ul, iar cea de-a doua nu este luata in considerare si poate contine orice valoare.

- permitem accesul clientilor in functie de criterii complexe. Este cea mai granulara abordare, dar si cea mai complexa. Presupune definirea unei tabele externe in care specificam, la nivel de fiecare client si mesaj, conditiile pentru acceptarea acestuia:

```
smtpd_*_restrictions = check_ccert_access tip_tabela:/cale/catre/tabela/externa
```

Fingerprint-ul unui certificat se obtine prin trecerea acestuia printr-o functie de hashing. Functia implicita se defineste cu parametrul smtpd_tls_fingerprint_digest si are valoarea md5. Fingerprint-ul unui certificat se poate obtine folosind *openssl*:

```
[ student@server ]$ openssl x509 -noout -fingerprint -sha1 -in certfile.pem
SHA1 Fingerprint=68:3B:2F:3E:1D:14:00:89:0B:55:76:8B:63:22:E7:1C:A0:CB:87:32
```

9.9. Masuri de prevenire a falsificarii mesajelor e-mail

9.9.1. SPF

SPF (Sender Policy Framework) este un sistem gandit sa ingreuneze falsificarea adresei sursa a mesajelor e-mail. In mod normal, un client care livreaza un mesaj e-mail unui server poate specifica in envelope orice adresa de expeditor, serverul putand face doar verificari limitate in privinta legitimitatii acelei adrese.

SPF permite administratorului DNS sa publice in cadrul zonei lista de statii care au voie sa trimita mail in numele acelui domeniu. Lista ia forma unei inregistrari de tip TXT sau, mai nou, SPF. Un server e-mail care receptioneaza un mesaj de la un client poate verifica chiar in timpul dialogului SMTP daca clientul este autorizat sa trimita mesajul, obtinand inregistrarea SPF corespunzatoare domeniului sursa si confruntand adresa clientului cu lista de statii autorizate.

Inregistrarea SPF are forma urmatoare:

```
domeniu.ro. SPF "v=spf1 a mx -all"
```

Inregistrarea de mai sus declara urmatoarele statii autorizate, eliminandu-le pe toate celelalte:

- statia sau statiile corespunzatoare inregistrarii de tip A cu numele domeniu.ro
- statiile corespunzatoare inregistrarii MX a domeniului

SPF realizeaza doar validarea expeditorului din envelope, insa nu poate detecta un mesaj falsificat (ex: headerul From: poate contine in continuare o valoare falsificata). In acest scop au fost gandite sisteme precum Domain keys sau DKIM.

9.9.2. Domain keys si DKIM

Cele doua sisteme au fost lansate in aceeasi perioada, fiind foarte asemanatoare. Domain Keys reprezinta implementarea propusa de Yahoo, pe cand DKIM (DomainKeys Identified Mail) reprezinta numele versiunii finale standardizate de IETF (vezi RFC 4871). DomainKeys, specificat si el in cadrul unui RFC, are acum statut de *historical*, si de aceea vom prezenta in continuare doar DKIM.

DKIM foloseste algoritmi criptografici pentru a asocia continutul unui mesaj cu un anumit domeniu DNS. Acest lucru se realizeaza astfel:

- mesajul este semnat digital de catre serverul expeditor folosind o cheie privata a domeniului; semnatura este inclusa in cadrul mesajului SMTP sub forma headerului *DKIM-Signature*
- cheia publica corespunzatoare este inclusa in zona DNS a domeniului pentru care se semneaza mailul
- un server care receptioneaza un astfel de mesaj semnat va extrage printr-o interogare DNS cheia publica corespunzatoare domeniului din adresa expeditorului (cea din headerul From: !) si va valida semnatura mesajului, asigurandu-se astfel de faptul ca mesajul a fost trimis de catre un server legitim al domeniului din From: .

In acest fel, destinatarul se poate asigura ca *continutul* mesajului a fost generat de catre un expeditor legitim al domeniului sursa, deoarece altcineva nu s-ar fi putut afla in posesia cheii private a domeniului.

9.10. BIBLIOGRAFIE

- Documentatia oficiala postix: http://www.postfix.org/documentation.html
- Lista parametrilor de configurare postfix: http://www.postfix.org/postconf.5.html
- Filtrare externa before-queue: http://www.postfix.org/SMTPD_PROXY_README.html
- Implementare SASL in postfix:
  - Configurare SASL in postfix (documentatie postfix): http://www.postfix.org/SASL_README.html
  - Folosirea implementarii SASL Dovecot in postfix (documentatie dovecot): http://wiki.dovecot.org/HowTo/PostfixAndDovecotSASL
- Documente RFC relevante:
  - RFC 5321 – SMTP: http://tools.ietf.org/html/rfc5321
  - RFC 4954 – SMTP AUTH: http://tools.ietf.org/html/rfc4954
  - RFC 4422 – SASL: http://tools.ietf.org/html/rfc4422
- Lista extensiilor SMTP: http://fortytwo.ch/smtp
- Carti:
  - The Book of Postfix: State-of-the-Art Message Transport
  - Postfix – The Definitive Guide
- Verificare online a unui server SMTP TLS: http://www.checktls.com/TestReceiver?LEVEL=2