6. SERVERUL DNS – CONFIGURARE AVANSATA SI SECURIZARE

6.1. Securizare generala

6.1.1. Stabilire de limite

Pentru a evita scenariile de tip Denial-of-Service, in care incarcarea excesiva a serverului sau chiar epuizarea unei resurse (memorie, procesor etc) impieteaza asupra calitatii sau continuitatii serviciului oferit clientilor, administratorul de server trebuie sa stabileasca limite pentru diversele aspecte ale functionarii serverului. Limitele trebuie impuse si pentru a asigura un serviciu de calitate, in lipsa unor atacuri DoS – spre exemplu, putem limita TTL-ul inregistrarilor primite, astfel incat acestea sa nu stationeze in cache pe perioade exagerate.

In cazul serverului BIND enumeram urmatoarele configurari utile:

- numarul maxim de clienti serviti simultan care adreseaza serverului interogari recursive: directiva **recursive-clients** ce primeste ca argument numarul de clienti (default: 1000)
- limite pentru transferurile de zona
  - numarul maxim de transferuri de zona simultane solicitate de catre serverul nostru (pentru rol de slave): directiva **transfers-in**. Valoarea default este 10
  - numarul maxim de transferuri de zona simultane acordate de catre serverul nostru (pentru rol de master): directiva **transfers-out**. Valoarea default este 10
  - numarul maxim de transferuri de zona simultane per master (serverul nostru nu va putea deschide cu o aceeasi statie master un numar mai mare de transferuri de zona, pentru a nu supraincarca master-ul): directiva **transfers-per-ns**. Valoarea default este 2
  - numarul maxim de transferuri de zona pentru o anumita statie master. Daca dorim tratament special pentru anumite statii, putem folosi in cadrul directivei server optiunea transfers (vezi exemple)
  - durata maxima a transferului de zona
    - pentru rol de slave: directiva **max-transfer-time-in**
    - pentru rol de master: directiva **max-transfer-time-out**
  - stabilirea de limite pentru timpii ce regleaza transferul de zona (aplicabil atunci cand serverul nostru joaca rol de slave). Serverul slave foloseste in mod normal timpii de refresh si retry specificati in RR-ul de tip SOA al zonei; pentru a impune limite pentru acesti timpi putem folosi directivele **max-refresh-time**, **min-refresh-time**, **max-retry-time** si **min-retry-time**
- limite pentru TTL-ul intregistrarilor memorate in cache: directivele **max-cache-ttl** si **max-ncache-ttl**
- intervalul de eliminare a inregistrarilor expirate din cache. Serverul nu monitorizeaza inregistrarile permanent, astfel incat sa le stearga din cache de indata ce expira, ci analizeaza cache-ul periodic, eliminandu-le pe cele expirate. Intervalul de curatare a cache-ului se seteaza cu directiva **cleaning-interval**

Exemplu:

```
options{
      transfers-in 15;
      transfers-out 20;
      transfers-per-ns 5;
      max-transfer-time-in 300;
      max-transfer-time-out 300;
      max-refresh-time 604800;          # 7 zile
      max-retry-time      86400;        # 1 zi
};
# particularizare setari pentru un anume server
server 10.0.0.100{
      transfers 3;
};
```

6.1.2. Rulare ca user neprivilegiat si problemele introduse

Orice server trebuie sa ruleze cu privilegiile minime necesare pentru a-si putea oferi serviciile. In cazul lui BIND acest lucru se realizeaza folosind optiunea **-u** a executabilului *named*, urmata de username-ul dorit:

```
named -u bind
```

Trebuie avut insa grija ca serverul sa aiba acces la diversele fisiere folosite, plus permisiunile necesare:

- drept de citire pentru fisierul de configurare
- drept de citire pe fisierele zona
- drept de citire/scriere pe fisierele log, daca serverul produce fisiere log proprii (vezi sectiunea din material dedicata logging-ului)
- drept de creare pentru fisierul jurnal in cazul in care se foloseste Dynamic DNS pentru una sau mai multe zone (vezi sectiunea din material dedicata DDNS)

6.1.3. Chrooting

6.1.3.1. Principiul de functionare

Chroot (change root) este un mecanism Unix/Linux prin care unui proces al sistemului de operare i se creeaza iluzia ca directorul radacina (/) se afla intr-un loc specificat de catre programator/administrator, altul decat cel original. Efectul este ca procesul va "trai" cu impresia ca sistemul de fisiere incepe din punctul impus prin operatia de chroot si nu va mai putea accesa fisiere aflate in afara acelui director. Directorul in care este consemnat procesul poarta denumirea de "chroot jail".

Exemplu: daca aplicam operatia de chroot serverului BIND, indicand ca director jail /var/named, atunci toate caile absolute folosite de catre named vor fi de fapt relative la acest director real. Fisierul de configurare, pe care serverul il acceseaza cu calea /etc/named.conf, se va afla de fapt in /var/named/etc/named.conf, deoarece ceea ce serverul percepe ca / este de fapt directorul real /var/named.

O astfel de restrictie aplicata unui proces server are ca rol minimizarea pagubelor in cazul unui atac reusit, care preia controlul asupra serverului. Atacatorul nu va mai putea parasi directorul jail, posibilitatile sale de actiune fiind astfel drastic limitate.

Trebuie avute insa in vedere urmatoarele aspecte:

- un server, pentru a functiona corect, are nevoie de un numar de resurse prezente in sistemul de fisiere original. Cand serverul ruleaza sub chroot, aceste fisiere trebuie create inauntrul jail-ului astfel incat ele sa re-devina disponibile si functionarea serverului sa nu fie afectata.
- pentru ca serverul sa nu poata iesi din jail este imperativ ca el sa nu fie rulat ca root, ci folosind un user neprivilegiat creat in acest scop

Nota: *lista de fisiere deschise de catre un proces activ poate fi vizualizata cu ajutorul comenzii lsof.*

6.1.3.2. Configurare si operatii necesare

Pentru a consemna serverul BIND intr-un director ales de catre administrator sunt necesare urmatoarele operatii:

1. **Crearea unui cont neprivilegiat pentru rularea serverului**

```
groupadd named
useradd -g named -s /bin/false named
```

2. **Crearea directorului jail si a subdirectoarelor necesare**. Directorul jail trebuie sa contina fisierul de configurare al serverului, eventualele fisiere zona si fisierul special /dev/null. Vom crea urmatoarele subdirectoare ale directorului jail:
   - **etc** – pentru a memora fisierul de configurare
   - **var** – folosit pentru fisierele zona si eventuale loguri produse de server
     - subdirectorul **run** – folosit pentru fisierul .pid al serverului (locatia fisierului pid se poate modifica din options{} folosind directiva pid-file urmata de calea dorita)

- **dev** – va contine fisierul special /dev/null

```
mkdir /var/chroot/BIND
cd $_
mkdir -p dev etc var/run
mknod dev/null c 2 2
```

3. **Stabilirea permisiunilor necesare pe fisiere si directoare**. In functie de configurarea sistemului de operare (si mai exact umask-ul default), este foarte posibil ca fisierul de configurare sa aiba permisiune de citire pentru toti utilizatorii. In schimb, serverul – care va rula cu userul named! - va avea nevoie de permisiune de creare de fisiere in var/run si posibil si in alte subdirectoare ale lui var (in cazul in care creeaza loguri proprii sau foloseste Dynamic DNS)

4. **Pornirea serverului cu optiunile corespunzatoare**:

```
named -u named -t /var/chroot/BIND
```

6.2. Configurarea logging-ului si diagnosticare avansata a serverului

6.2.1. Concepte generale ce tin de logging in BIND9

Serverul BIND poate genera informatii foarte bogate despre operatiile pe care le efectueaza. Fiecare mesaj de logging generat are doua caracteristici principale:

- **categorie** – specifica operatia de care apartine mesajul de logging. In acest fel putem inregistra separat logurile pentru anumite actiuni ale serverului: transfer de zona, interogari primite de la clienti, erori legate de autentificare etc. Lista completa a categoriilor de logging se gaseste in Anexa 1.
- **nivel de importanta** – serveste de asemenea la tratarea diferentiata a informatiilor de logging. In etapa de configurare/testare a serverului putem alege sa vizualizam toate informatiile, indiferent de gradul de importanta (ceea ce va produce o cantitate mare de informatie), urmand ca, dupa ce ne-am asigurat ca serverul este stabil si functioneaza corect, sa nu pastram in loguri decat mesajele de la un anumit nivel de importanta in sus

In functie de aceste doua caracteristici, mesajele fiecarei categorii pot fi trimise in mod independent catre unul sau mai multe canale. Un canal reprezinta o destinatie posibila pentru mesaje de logging. BIND suporta patru astfel de destinatii:

- **fisier** – pentru cazul in care dorim ca serverul sa isi creeze fisiere log proprii
- **syslog** – informatia este inmanata daemonului syslogd (system logger) care va decide unde si daca sa o consemneze pe baza fisierului sau de configurare
- **stderr** – informatia este afisata in terminalul la care este conectat serverul (util de folosit daca serverul este pastrat in foreground folosind optiunea -f)
- **null** – informatia este distrusa, neinregistrandu-se niciunde

Aceste doua elemente – categorii si canale – pot fi “incrucisate” de catre administrator dupa cum doreste: putem trimite informatiile mai multor categorii in acelasi fisier, si de asemenea putem trimite mesajele unei singure categorii in mai multe fisiere.

Configurarile legate de canale si asignarea lor la diversele categorii se realizeaza in cadrul directivei bloc *logging{}* din named.conf.

6.2.2. Definirea unui canal

Canalele de logging se definesc folosind directiva *channel{}* in interiorul lui *logging{}*, ca in exemplul urmator:

```
logging{
      channel log-general{
            file "/var/log/named/general.log" versions 4 size 1M;
            severity info;
            print-time yes;
      };

      channel log-syslog{
            syslog daemon;
            severity info;
      };

      channel terminal{
            stderr;
            severity info;
      }

      channel gunoi{
            null;
      };
};
```

Orice canal are urmatoarele caracteristici:

- destinatia datelor trimise catre acel canal – este una dintre cele 4 destinatii prezentate anterior. In functie de cea aleasa, directivele de configurare difera:
  - fisier: se foloseste directiva **file**, care specifica si calea catre fisierul dorit, ca in exemplul de mai sus. Exista posibilitatea limitarii dimensiunii maxime a fisierului si a numarului de fisiere pastrate (odata ce un fisier log atinge dimensiunea maxima, el este redenumit si in locul sau se deschide altul nou)
  - syslog: folosim directiva **syslog** urmata de facility (categoria syslog in care se incadreaza informatia). Categoria poate fi *kern, user, mail, daemon, auth, syslog, lpr, news, uucp, cron, authpriv, ftp, local0, local1, local2, local3, local4, local5, local6* sau *local7*
  - consola/terminalul la care este conectat serverul – folosim directiva **stderr**. Aceasta destinatie este utila numai in masura in care serverul ruleaza in foreground (optiunea -f utilizata la pornire)
  - ignorare – pentru a "arunca la gunoi" o informatie, folosim ca destinatie a canalului directiva **null**
- nivelul minim de importanta pe care mesajul trebuie sa il aiba ca sa fie consemnat de catre acest canal. Aceasta valoare actioneaza ca un filtru, pastrandu-se numai mesajele avand importanta superioara nivelului specificat. Folosim in acest scop directiva **severity**, urmata de un nivel care se poate incadra intr-una din doua categorii:
  - prioritati syslog, cu valorile posibile *critical, error, warning, notice, info*
  - prioritati BIND, care sunt inferioare ca nivel celor syslog si sunt folosite pentru debugging. Valori posibile: *debug [nivel]* sau *dynamic.* In primul caz vor fi logate doar mesajele cu nivel de debug superior celui specificat, pe cand in al doilea caz vor fi logate doar cele al caror nivel de debug se afla deasupra celui curent al serverului. Nivelurile de debug pornesc de la 1 si continua pana la 99. **Atentie! Mesajele de debug nu pot fi trimise catre syslog!**

Nota: *nivelul de debug al serverului de seteaza fie la pornire, folosind optiunea -d, fie in timpul rularii, folosind comanda* ***rndc trace****.*

In plus fata de cele doua caracteristici principale, un canal dispune de cateva optiuni utile:

- consemnarea momentului de timp cand a fost generat mesajul de logging, folosind directiva **print-time**. Valori posibile: yes sau no
- consemnarea categoriei in care se incadreaza mesajul, folosind directiva **print-category**. Valori posibile: yes sau no
- consemnarea nivelului de importanta al informatiei logate, folosind directiva **print-severity**. Valori posibile: yes sau no

Exemplu:

```
logging{
      channel fisier{
            file "/var/log/named.log";
            severity dynamic;    # numai mesajele care au acelasi nivel de debug cu serverul
            print-time yes;      # in syslog se adauga automat timestamp; in fisier adaugam noi
      };
};
```

BIND dispune de patru canale predefinite, a caror definitie este prezentata mai jos, asa cum apare ea in BIND ARM:

```
channel default_syslog {
    syslog daemon;                      // send to syslog's daemon facility
    severity info;                      // only send priority info and higher
};

channel default_debug {
    file "named.run";                   // write to named.run in the working directory
                                        // Note: stderr is used instead of "named.run" if the
                                        //     server is started with the '-f' option.
    severity dynamic;                   // log at the server's current debug level
};

channel default_stderr {
    stderr;                             // writes to stderr
    severity info;                      // only send priority info and higher
};

channel null {
   null;                                // toss anything sent to this channel
};
```

Atentie! *Un canal, odata definit, nu poate fi redefinit! Acest lucru este valabil si pentru canalele default.*

6.2.3. Categorii si asignarea lor la canale

Pentru a folosi canalele create este nevoie sa specificam ca categorii de mesaje vor fi trimise catre fiecare canal. Realizam acest lucru cu ajutorul directivei *category*, care are structura de mai jos:

```
category nume_categorie{   nume_canal_1; nume_canal_2; ... };
```

Mesajele unei categorii pot fi trimise catre mai multe canale si viceversa (un canal poate primi mesaje ale mai multor categorii).

Lista extinsa a categoriilor poate fi gasita in Anexa 1. Iata cateva categorii mai importante:

- **xfer-in** – cuprinde mesajele de logging legate de transferuri de zona receptionate (rol de slave)
- **xfer-out** – mesaje legate de transferuri de zona in care serverul joaca rol de master
- **queries** – mesaje legate de interogarile primite de la clienti. Logarea interogarilor este dezactivata din oficiu, deoarece produce cantitati mari de informatie, insa poate fi activata pentru debugging-ul configuratiei serverului in cel putin doua moduri (vezi mai jos informatiile legate de query logging)
- **update** – cuprinde mesajele Dynamic DNS, care produc modificari asupra zonelor
- **default** – "prinde" toate mesajele apartinatoare de categorii care nu au fost specificate explicit in directiva logging

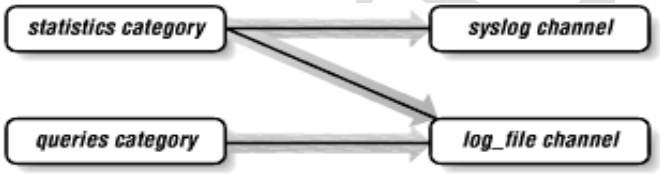
Exemplu: configurarea corespunzatoare figurii alaturate:

```
logging{
      channel syslog{
            syslog daemon;
            severity info;
      };

      channel log_file{
            file "/var/log/named/statistics.log" versions 2 size 10M;
            severity dynamic;
            print-time yes;
      };

      category statistics{
            syslog;
            log_file;
      };

      category queries{
            log_file;
      };
};
```



Daca administratorul serverului nu ia niciun fel de masura privind logging-ul, configuratia implicita BIND este urmatoarea:

```
category default { default_syslog; default_debug; };
category unmatched { null; };
```

Atentie! *La pornirea serverului, toate mesajele referitoare la erori de sintaxa in fisierul de configurare vor fi trimise catre canalele default (sau catre terminal daca serverul a fost pornit cu optiunea -g). Aceasta deoarece, in caz de eroare de sintaxa, serverul nu ajunge sa interpreteze directiva logging si sa preia setarile dorite de administrator.*

6.2.4. Rularea serverului in mod debug si informatiile generate

Pentru a diagnostica un server DNS avem la dispozitie diferite modalitati:

- rularea serverului cu optiunea -g. Aceasta va rula serverul in foreground, fortand tot output-ul serverului sa fie afisat pe ecranul terminalului din care acesta a fost pornit. Serverul nu va mai scrie informatii in fisiere, directiva logging fiind in marea sa majoritate ignorata

  Nota: *exceptie face specificarea categoriei queries, care are ca efect activarea query logging-ului si in cazul rularii cu -g.*

  Rularea cu -g trimite toate informatiile de logging catre canalul default_stderr, care are severity dynamic si deci putem ajusta gradul de detaliu al informatiei de debug afisate (vezi mai jos setarea nivelului de debug).

- rularea serverului cu optiunea -f (foreground). Diferenta fata de -g este ca logging-ul se realizeaza conform directivei logging{}; acum putem alege ce informatii sa fie afisate pe ecran prin definirea unui canal cu logare in stderr si trimiterea catre el doar a informatiilor dorite
- rularea serverului in mod debug. Este utila doar in masura in care exista cel putin un canal de logging care "prinde" informatiile de nivel debug (declarat cu severity debug *nivel* sau severity dynamic). Exista doua posibilitati de intrare in mod debug:
  - pornire de la bun inceput in mod debug, folosind optiunea -d urmata de nivelul de debug dorit
  - modificare runtime a nivelului de debug, folosind comanda **rndc trace *nivel_dorit***. Anularea modului debug se realizeaza cu **rndc notrace**

  In general, nivelul 3 de debug ofera un grad de detaliu suficient pentru majoritatea operatiilor serverului, nivelurile superioare fiind mai degraba utile dezvoltatorilor:

```
received control channel command 'trace 3'
debug level is now 3
client 193.230.161.3#64303: UDP request
client 193.230.161.3#64303: request is not signed
client 193.230.161.3#64303: recursion not available
client 193.230.161.3#64303: query
client 193.230.161.3#64303: query: www.firma.ro IN A -
client 193.230.161.3#64303: query 'www.firma.ro/A/IN' approved
client 193.230.161.3#64303: send
client 193.230.161.3#64303: sendto
client 193.230.161.3#64303: senddone
client 193.230.161.3#64303: next
client 193.230.161.3#64303: endrequest
client @0xb538c008: udprecv
```

6.2.5. Query logging

Consemnarea in loguri a interogarilor primite de la clienti este dezactivata din oficiu, deoarece poate genera cantitati mari de informatie. Activarea se poate realiza in urmatoarele moduri:

- folosind optiunea querylog in sectiunea options{} din named.conf. Valori posibil: yes sau no
- in timpul rularii serverului, folosind comanda **rndc querylog.** Aceasta comuta starea query logging-ului la fiecare apelare, avand avantajul ca putem astfel sa adaugam informatii de logging fara a fi necesara oprirea serverului
- specificand categoria queries in cadrul directivei logging{} si trimitand-o catre un canal care nu are ca destinatie /dev/null

Atentie! *Simpla folosire a categoriei queries in directiva logging activeaza query logging-ul in cazul rularii serverului cu named -g, chiar daca aceasta categorie este trimisa in null! Aceasta deoarece -g are ca efect fortarea trimiterii logging-ului catre stderr, ignorand destinatiile specificate in fisierul de configurare.*

6.3. Perspective diferentiate (views)

6.3.1. Principii

BIND poate fi configurat sa prezinte informatii diferite in functie de client. Spre exemplu, sa consideram cazul unei mici retele locale care se foloseste de un server DNS propriu; cateva statii ale retelei au atat adresa privata cat si publica, una dintre ele fiind serverul web. Cerinta este ca, atunci cand este interogat din interiorul retelei in legatura cu statia www, serverul DNS sa raspunda cu adresa privata a serverului web, iar cand este interogat din afara sa raspunda cu cea publica. Aceasta presupune ca serverul DNS sa poata trata diferit cererile clientilor in functie de adresele IP ale acestora.

Pentru a decide ce informatii sa ofere clientului, serverul DNS poate tine cont de:

- adresa IP a clientului – raspunsul va diferi in functie de *cine* intreaba
- adresa IP a serverului pe care vine interogarea – raspunsul difera in functie de *unde* vine interogarea
- tipul interogarii – iterativa sau recursiva

6.3.2. Configurare BIND

Administratorul obtine acest efect definind asa-numite view-uri (perspective diferentiate in functie de parametrii cererii primite de la client). Fiecare view reprezinta o configurare a serverului care se aplica numai pentru anumite cereri – un fel de server virtual. Un view poate contine optiuni proprii, zone proprii etc, semanand din acest punct de vedere cu ideea de virtual host intalnita la serverul web.

Un view se creeaza folosind directiva bloc cu acelasi nume. Cererile carora li se aplica setarile din acel view pot fi selectate in diverse moduri:

- in functie de adresa IP a clientului, folosind directiva bloc **match-clients{}**. Aceasta primeste ca argument o insiruire de adrese, subnet-uri, acl-uri sau chei. **Atentie! In interiorul unui view nu se pot defini acl-uri!**

```
acl "retea1" { 172.16.0.0/16; };
key "cheie1" { .....definitie cheie...};
view "retea-locala"{
      match-clients { 192.168.0.0/24; retea1; key cheie1; };
      recursion yes;
};

view "extern" {
      match-clients{ any; };
      recursion no;
      zone "test.ro" { .......};
};
```

- in functie de adresa destinatie a cererii (adresa IP a serverului pe care acesta a receptionat cererea), folosind directiva bloc **match-destinations{}**. Aceasta primeste ca parametri o lista de adrese IP ale serverului
- in functie de tipul de cerere – recursiva sau iterativa. Putem folosi directiva simpla **match-recursive-only**, cu argumentele posibile yes sau no

Nota: *cele 3 directive sunt cumulabile – putem impune ca un view sa se aplice doar interogarilor recursive venite de la anumiti clienti si care au o anumita adresa destinatie.*

Daca administratorul nu defineste niciun view, serverul creeaza automat unul default care se aplica tuturor clientilor, iar directivele *zone{}* sunt considerate a face parte din acest view. Daca a fost definit cel putin un view, atunci orice declaratie de *zone{}* trebuie sa faca parte dintr-un view, nemaiputand fi plasata in "radacina" fisierului de configurare.

Un view poate contine si optiuni. Optiunile, atunci cand apar, nu mai sunt specificate in cadrul unei directive bloc options{} ci direct in cadrul view-ului (vezi directiva recursion din exemplele de mai sus). Ele vor avea prioritate fata de optiunile cu acelasi nume din directiva options globala.

Exemplu: configurare pentru un server dual care actioneaza intr-o retea locala. Serverul permite recursivitatea numai statiilor din reteaua locala si prezinta zona diferite pentru clientii din reteaua locala si cei externi:

```
acl "lan"{ 192.168.0.0/24; };

view "intern"{
      match-clients { lan; };
      recursion yes;
      forwarders { 193.231.236.30; 193.230.161.3; };
      zone "firma.ro" { type master; file "/var/named/firma.ro-intern"; };
};

view "extern"{
      match-clients { ! lan; };
      recursion no;
      zone "firma.ro" { type master; file "/var/named/firma.ro-extern"; };
};
```

Cele doua fisiere zona sunt complet distincte, administratorul populandu-le in functie de necesitati.

6.4. Securizarea comunicatiilor serverului folosind TSIG

6.4.1. Probleme de securitate si solutii

Protocolul DNS se bazeaza pe UDP, ceea ce il face vulnerabil la o intreaga serie de atacuri. Principala problema este ca nu exista o etapa de stabilire a conexiunii, ca in cazul TCP; clientul poate trimite direct o interogare serverului iar acesta va genera un raspuns. Un atacator ar putea exploata acest lucru in multiple moduri:

- atacatorul poate falsifica adresa sursa a cererii adresate unui server

  - posibilitatea 1: pachetul de raspuns va fi trimis catre o alta statie. In acest fel pot fi create atacuri cu amplificare de banda, in care atacatorul interogheaza multe servere DNS folosind ca adresa sursa adresa victimei, iar toate pachetele de raspuns vor fi trimise catre victima, cauzand Denial of Service
  - posibilitatea 2: cererea este una de actualizare a zonei, adresata serverului primar. Atacatorul poate falsifica adresa sursa astfel incat cererea sa para ca vine din partea unuia dintre serverele secundare. Daca primarul permite serverelor secundare update-ul de zona bazandu-se numai pe adresa IP a acestora, atunci atacatorul are posibilitatea de a modifica continutul zonei dupa plac
- atacatorul poate falsifica adresa sursa a unui raspuns trimis catre un client DNS. In acest fel el poate incerca "pacalirea" clientului, pasandu-i informatie care directioneaza clientul catre masini aflate sub controlul atacatorului. Intr-un caz mai elaborat, atacatorul poate falsifica un raspuns trimis de catre un server altuia; daca acesta din urma este nu caching name server si atacatorul reuseste sa "strecoare" in cache-ul acestuia o informatie falsa, vor fi afectati toti clientii acelui server care a suferit atacul ("DNS cache poisoning" este numele acestui tip de atac)

Protectia impotriva acestui tip de atacuri presupune ca mesajele schimbate intre servere si clienti (fie acestia din urma resolver-e sau alte servere) sa fie autentificate, astfel incat un atacator sa nu poata genera acele mesaje. In acest scop a fost gandit mecanismul **TSIG (transaction signatures)**, prin care fiecarui mesaj i se adauga o semnatura digitala obtinuta prin criptarea hash-ului mesajului DNS folosind o cheie secreta cunoscuta de catre ambele parti.

Desi foarte util, TSIG are totusi neajunsurile sale: presupunand totusi ca un server a fost compromis, de unde pot sti clientii care comunica cu el ca informatia pe care o primesc nu este valida? (aceasta in conditiile in care semnaturile digitale generate de acel server sunt valide!) Este necesara autentificarea inregistrarilor care compun fisierul zona. De aceea a fost gandit **DNSSEC (DNS Security Extensions)**, care ataseaza fiecarui RRSET o semnatura digitala si creeaza o ierarhie de trust in arborele DNS asemanatoare cu cea a CA-urilor (detalii in sectiunea DNSSEC).

Aplicabilitate:
- TSIG autentifica servere sau clienti, nu informatii cuprinse in zone
- DNSSEC autentifica informatiile din zone

Sectiunea de fata trateaza TSIG. DNSSEC este un subiect mai complex si va fi prezentat intr-o sectiune separata, dedicata lui.

6.4.2. Descrierea mecanismului TSIG

Mecanismul TSIG presupune adaugarea unei semnaturi digitale la fiecare mesaj schimbat intre doua statii. Semnatura este calculata criptand hash-ul mesajului cu o cheie simetrica, cunoscuta de ambele parti. Semnatura este inclusa in pachet sub forma unei inregistrari de tip TSIG; aceasta este o asa-numita *meta-inregistrare,* in sensul in care ea nu apare in fisierul zona, ci doar in cadrul mesajelor DNS.

Nota: *TSIG permite doar verificarea integritatii si autentificare, insa nu cripteaza in niciun fel datele!*

In BIND, algoritmul folosit pentru obtinerea semnaturii este HMAC-MD5 – un algoritm de hashing parametrizat cu o cheie. Algoritmul foloseste ca date de intrare informatiile cuprinse in mesajul DNS si inca cateva suplimentare – de exemplu, momentul de timp in care a fost generata semnatura, pentru a proteja impotriva atacurilor de tip replay[1].

Corecta functionare a mecanismului TSIG presupune existenta unei chei secrete cunoscuta de ambele statii. Dupa cum se cunoaste, aceasta abordare ridica problema distributiei cheii. Solutiile sunt urmatoarele:
- pre-shared secret: cheia secreta este distribuita prin mijloace externe pe ambele statii care vor fi implicate in dialog inaintea debutului acestuia
- shared secret generat in mod dinamic – cele doua statii pot folosi un algoritm (ex: Diffie-Hellman) pentru a construi impreuna aceeasi cheie secreta, pe care o vor folosi ulterior pentru a genera semnaturile TSIG. In acest scop este folosita o meta-inregistrare suplimentara, de tip TKEY

1 Replay attack – un tip atac in care un atacator inregistreaza succesiunea de pachete schimbata intre doua statii (chiar daca nu le intelege continutul si nu le poate modifica) si apoi le retransmite in aceeasi succesiune ("replay") catre una dintre parti

Prezentam in continuare implementarea primei variante in BIND.

6.4.3. Configurare BIND

Pentru a autentifica comunicatia a doua servere sunt necesare urmatoarele etape:

- se alege sau se genereaza cheia simetrica
- se configureaza cheia pe ambele statii
- se foloseste cheia ca conditie in directive de configurare care reglementeaza operatii ale serverului (transfer de zona, dynamic update etc)

Cheia poate fi aleasa de catre administrator sau poate fi generata folosind utilitarul dnssec-keygen cuprins in distributia BIND. Utilitarul este folosit atat pentru generarea de chei simetrice cat si asimetrice. Iata un exemplu de comanda de generare de cheie simetrica:

```
dnssec-keygen -a hmac-md5 -n host -b 128 s1-s2
```

In comanda de mai sus, optiunile au urmatoarele semnificatii:

- **-a** specifica algoritmul folosit (la ora actuala singura varianta fiind HMAC-MD5)
- **-n** specifica tipul de cheie, deoarece utilitarul dnssec-keygen este folosit si pentru a genera alte tipuri de chei, folosite pentru semnarea informatiilor din cadrul unei zone
- **-b** specifica numarul de biti ai cheii generate

Argumentul comenzii, *s1-s2*, reprezinta numele cheii. Ca urmare a comenzii de mai sus vor fi generate doua fisiere in directorul curent: unul de forma *Ks1-s2.+157+47504.private* si altul de forma *Ks1-s2.+157+47504.key*. Cele doua contin informatie similara, dar formatata diferit; dnssec-keygen creeaza intotdeauna doua fisiere (.private si .key) chiar si atunci cand genereaza o cheie simetrica. Fisierul *Ks1-s2.+157+47504.key* va contine urmatoarele:

```
s1-s2. IN KEY 512 3 157 CkAB6zn3AkZDdzJ/NxZtJw==
```

De interes este numai ultimul camp, care reprezinta forma codata BASE64 a cheii.

Nota: *daca cheia este aleasa de catre administrator, ea va trebui la randul sau codata BASE64, iar stringul rezultat va fi cel folosit in fisierul de configurare BIND. Codarea BASE64 poate fi realizata folosind openssl astfel:* ***echo cheie_aleasa|openssl base64***

Cheia aleasa/generata trebuie acum configurata pe fiecare dintre cele doua servere BIND, folosind in named.conf directiva bloc **key{}**, dupa cum urmeaza:

```
key s1-s2 {
    algorithm hmac-md5;
    secret "CkAB6zn3AkZDdzJ/NxZtJw==";
};
```

Atentie! *Numele cheii se transmite pe post de nume in meta-inregistrarea TSIG – trebuie ca el sa fie identic pe ambele statii!!*

Odata configurata pe cele doua servere, cheia se poate folosi:

- pentru a impune restrictii in unele directive de configurare, cum ar fi:
  - **allow-query** – putem permite interogarea numai de la clientii care folosesc o anumita cheie:

```
allow-query { key s1-s2; };
```

  - **allow-update** – permitem modificarea dinamica a zonei numai de catre clientii care se autentifica cu o anumita cheie:

```
allow-update { key server1-server2; };
```

- **allow-transfer** – permitem transferul de zona numai catre secundarele care se autentifica folosind o anumita cheie:

```
zone "exemplu.ro"{
      allow-transfer { key s1-s2; };
      .....
};
```

- ca parametru in alte directive de configurare
  - in cazul unui slave trebuie specificata cheia folosita pentru autentificarea comunicatiei cu master-ul:

```
masters { 10.0.0.2 key cheie; };
```

  - putem specifica cheia folosita de serverul nostru pentru generarea semnaturii mesajelor trimise unui anume alt server, incluzand declaratia de cheie intr-o directiva bloc server{} ca in exemplul de mai jos. Nota: aceasta directiva nu obliga celalalt server sa isi semneze mesajele cu aceeasi cheie!

```
# cheia de mai jos va fi folosita pentru a semna mesajele trimise lui server2
server 10.0.0.100 {
      keys { server2; };
};
```

Atentie! Semnatura depinde de momentul de timp in care a fost generata, asa cum este el perceput de catre statia care o creeaza. Pentru ca semnatura sa fie considerata valida trebuie ca deplasamentul dintre ceasurile celor doua statii sa fie de maxim 5 minute. Se recomanda folosirea NTP pentru sincronizarea ceasurilor celor doua statii.

Exemplu: securizare transfer de zona:

```
# configurare server master
options{
      allow-transfer { none; };
};

key cheie_transfer{
      algorithm hmac-md5;
      secret "CkAB6zn3AkZDdzJ/NxZtJw==";
};

zone "exemplu.ro"{
      type master;
      allow-transfer{ key cheie_transfer; };
      ......
};

# configurare slave
key cheie_transfer{
      algorithm hmac-md5;
      secret "CkAB6zn3AkZDdzJ/NxZtJw==";
};

zone "exemplu.ro"{
      masters { 10.0.0.5 key cheie-transfer; };
};
```

6.5. Dynamic DNS

6.5.1. Scop, utilitate

DNS-ul a fost gandit initial ca un sistem care creeaza o baza de date statica – inregistrarile care o compun sunt editate de catre administratorii DNS, avand in vedere ca numarul de interogari primite de baza de date este mult inferior numarului de actualizari necesare.

Odata cu raspandirea DHCP a aparut insa nevoia de a mentine sincronizarea nume-adresa chiar si in conditiile in care adresa unei statii se schimba relativ des. Acest lucru ar fi presupus un efort important pentru administratorul DNS (presupunand ca ar fi avut sens ca sincronizarea sa se efectueze manual), solutia fiind un mecanism prin care actualizarile bazei de date DNS sa se efectueze in mod automatizat. Extensia DNS Dynamic Updates a fost specificata in RFC2136 si deschide calea modificarii informatiei unei zone prin mesaje DNS de tip UPDATE. In acest fel, in momentul in care un client DHCP primeste o noua configurare de la serverul sau, este posibil ca fie clientul, fie serverul DHCP insusi sa instiinteze serverul DNS al zonei de noua adresa a statiei client, actualizand inregistrarea corespunzatoare ei din cadrul zonei.

6.5.2. Prezentare mecanism

Actualizarile bazei de date DNS pot fi solicitate din diverse directii:

- clienti DHCP care au primit o noua configurare si doresc ca perechea nume-adresa ce le corespunde in zona din care fac parte sa fie adusa la zi
- servere DHCP care au acordat o noua configurare unui client si care solicita in numele clientului actualizarea inregistrarii DNS corespunzatoare. Aceasta solutie are avantajul de a fi mai sigura – serverul DNS va fi configurat sa permita actualizari de zona din partea unei singure statii, nu a tuturor clientilor
- utilitare de interogare manuala. BIND pune la dispozitia administratorului unelte cu care sa poata trimite cereri de update unui server in scop de diagnosticare

Pentru a efectua o modificare a unei zone DNS, clientul care doreste acest lucru va obtine mai intai lista de inregistrari NS ale zonei ce face subiectul update-ului. Update-ul va fi directionat catre unul dintre aceste servere, dupa cum urmeaza:

- daca serverul este cel primar al zonei, acesta verifica daca conditiile pentru efectuarea update-ului sunt indeplinite (vezi mai jos) si daca clientul este autorizat sa produca acele modificari si, in caz afirmativ, efectueaza actualizarile cerute
- daca serverul este unul secundar, el va pasa cererea de update mai departe catre master-ul sau; daca acesta nu este serverul primar, va pasa la randul sau cererea mai departe pana cand ea poposeste pe serverul primar al zonei, unde este prelucrata conform celor de la punctul anterior. Serverul primar al unei zone ramane singurul pe care se pot aduce modificari zonei, fie ele manuale sau dinamice

Operatiile pe care le poate solicita un update sunt cele de adaugare sau stergere. Ele pot fi aplicate pentru:

- inregistrari individuale (RR)
- RRSET-uri - grupuri de inregistrari care au acelasi nume DNS si acelasi tip (ex: mai multe adrese pt acelasi nume DNS)

O operatie de update poate avea atasat un set de “prerequisites” (impuneri) - conditii obligatorii ce trebuie indeplinite pentru ca update-ul sa poata fi efectuat. Conditiile se pot specifica fie in termeni de nume si tip de RR (ex: sa nu existe o inregistrare *www* de tip *A*) fie specificand si valoarea corespondenta (ex: sa nu existe o inregistrare *www* de tip *A* cu adresa *1.2.3.4*). Tipurile de conditii posibile sunt:

- inregistrari care sa existe deja (yxrrset)
- inregistrari care sa lipseasca (nxrrset)
- domenii care sa existe deja (yxdomain)
- domenii care sa lipseasca (nxdomain)

6.5.3. Cum trateaza BIND update-urile de zona

Cand primeste o interogare care solicita modificarea unei zone, BIND nu va opera imediat asupra fisierului zona, ci va efectua modificarea in copia zonei pe care o are incarcata in memorie si va inregistra operatia dorita intr-un fisier log aditional – asa-numitul "jurnal" al zonei.

Jurnalul are doua utilizari:

- serverul efectueaza periodic (cu perioada configurabila) actualizarea fisierului zona folosindu-se de fisierul jurnal. In acest fel, serverul este degrevat de overhead-ul actualizarii imediate a fisierului zona la fiecare update primit. Dupa fiecare actualizare dinamica a unei zone, serverul BIND9 va incrementa automat versiunea acesteia
- jurnalul este folosit in cazul transferurilor de zona incrementale (cele in care se transmit slave-ului numai modificarile efectuate de la ultima sincronizare)

Fisierul jurnal are format binar si este creat automat de catre server la primul update adresat zonei. Numele sau este identic cu al zonei, avand insa extensia .jnl.

Atentie! *Fisierul jurnal este creat de catre server – nu poate fi creat sau editat de catre administrator. In acest scop, serverul (care ruleaza cu un UID neprivilegiat) trebuie sa aiba drept de scriere pe directorul respectiv, pentru a putea crea fisierul!*

6.5.4. Probleme de securitate ridicate si rezolvarea lor

Mecanismul de dynamic update, desi evident util, poate deveni periculos in lipsa unui control strict al listei de clienti care au permisiunea de a efectua update-uri. BIND9 permite administratorului sa autorizeze update-ul numai pentru:

- clienti cu anumite adrese IP (spre exemplu, cazul unui server DHCP, care are in general adresa IP stabila)
- clienti care cripteaza comunicatia cu serverul DNS folosind o anumita cheie simetrica

6.5.5. Configurare dynamic update in BIND

In configurarea implicita, BIND nu permite actualizarea dinamica a zonelor sale. Administratorul trebuie sa autorizeze update-ul la nivel de fiecare zona, in mod controlat, folosind una dintre urmatoarele directive:

- **allow-update{}** - directiva bloc utilizabila in cadrul declaratiei *zone{}* si care poate restrictiona update-ul numai pentru anumiti clienti, in functie de caracteristicile clientului. Odata clientul autorizat, acesta poate efectua orice operatii in cadrul zonei. Conditiile de autorizare se pot pune:
  - specificand IP-urile/subnet-urile clientilor - mecanismul trebuie aplicat cu grija pentru ca nu este nicidecum unul sigur: pachetele de update sunt UDP si deci pot fi usor falsificate sa contina orice adresa sursa. In acest fel, orice soft care poate falsifica adresa sursa a unui pachet UDP poate produce modificari in fisierul zona
  - specificand numele cheilor TSIG ale clientilor carora li se permite update-ul

```
allow-update{ 10.0.0.5; };
allow-update{ key updkey1; };
```

- **update-policy{}** (introdusa in BIND9) – permite un control al accesului mult mai granular in cazul in care comunicatia este autentificata folosind TSIG, prin specificarea unei liste de filtre (reguli) ce se aplica clientilor si operatiilor cerute de catre acestia, asemanator cu cazul unui firewall. In acest fel clientul, odata autorizat, nu capata neaparat privilegii complete asupra zonei. Sintaxa generala a unuia dintre filtrele componente este:

```
( grant | deny ) numeCheie tipNume numeDNS [ tipuri ]
```

Parametrii au urmatoarele semnificatii:

- *numeCheie* este numele cheii asa cum apare el in directiva *key{}* si este folosit pentru a identifica clientul caruia i se aplica regula curenta

- *tipNume* este folosit in conjunctie cu *numeDNS* pentru a restrange setul de inregistrari pe care are voie sa le modifice clientul. Serverul compara numele DNS al inregistrarii pe care clientul doreste sa o modifice cu valoarea campului *numeDNS*. Comparatia poate fi facuta in mai multe feluri, in functie de valoarea lui *tipNume*:
  - **name** – verifica daca numele DNS al inregistrarii pe care clientul doreste sa o modifice se potriveste exact cu *numeDNS*
  - **subdomain** – indica o potrivire partiala; numele DNS solicitat de client trebuie sa constituie un subdomeniu al lui *numeDNS* (sau poate fi identic cu acesta)
  - **wildcard** – adauga posibilitatea folosirii de metacaractere in numeDNS. Spre exemplu, putem permite unui client update-ul numai pentru inregistrarile de forma *.test.ro
  - **self** – permite clientului sa modifice numai inregistrarile care au acelasi nume cu al CHEII (nu cu numeDNS!). Solutia este utila atunci cand avem cate o cheie per inregistrare si distribuim cate o cheie fiecarui client pentru a-si actualiza propria inregistrare

Atentie! *Directivele update-policy{} si allow-update{} se exclud reciproc, neputand fi folosite simultan in cadrul aceleiasi zone BIND.*

```
update-policy{
      # serverul DHCP care foloseste cheia cu numele dhcp.test.ro. poate actualiza orice
      # inregistrari de tip A din test.ro
      grant dhcp.test.ro. wildcard *.test.ro A
}
```

- **allow-update-forwarding –** aplicabila numai pentru serverele slave. Atunci cand un server slave primeste o cerere de dynamic update, el o va trimite mai departe catre serverul sau master (“update forwarding”) numai in masura in care aceasta directiva de configurare i-o permite. Masura este luata deoarece, presupunand ca serverul master permite update-ul pentru slave si slave-ul forward-eaza toate cererile primite, atunci orice client al slave-ului ar putea produce modificari pe master prin intermediul slave-ului, chiar daca master-ul nu-i permite accesul direct.

```
allow-update-forwarding{ 10.0.0.0/24; 127.0.0.1; };
```

Nota: *pentru sistarea temporara a update-urilor adresate unei zone se poate folosi comanda* ***rndc freeze numezona****. Reluarea actualizarilor se efectueaza cu* ***rndc thaw numezona****.*

6.5.6. Diagnosticare dynamic updates

BIND pune la dispozitia administratorului utilitarul **nsupdate**, prin intermediul caruia se pot efectua la cerere operatii de dynamic update, cu sau fara autentificare TSIG. Cuplat cu faptul ca putem loga intr-un fisier separat mesajele categoriei update, aceasta ne permite diagnosticarea facila a problemelor de dynamic update.

Utilitarul nsupdate citeste operatiile de efectuat fie dintr-un fisier (specificat la apelarea sa) fie de la tastatura, in mod interactiv. Iata cateva optiuni utile:

- **-k** – atunci cand operatia de update foloseste TSIG si necesita o cheie, optiunea -k specifica locatia fisierului ce contine cheia. Acesta trebuie sa aiba un nume de forma *K{nume_cheie}.+157.+{secventa_random}.private* si sa fie insotit de fisierul .key cu acelasi nume (acesta este formatul in care dnssec-keygen genereaza cheile)
- **-y** – cheia poate fi specificata si direct in linia de comanda (cu dezavantajul ca astfel va fi salvata in history). Parametrul lui -y cuprinde 3 portiuni, separate prin ':' - algoritmul, numele cheii si valoarea cheii (vezi exemplu)
- **-d** – ruleaza nsupdate in mod debug, ceea ce va afisa si mesajele schimbate intre nsupdate si server
- **-D** – rulare cu nivel de debugging maxim

Iata principalele comenzi disponibile la rularea nsupdate in mod interactiv:

- **server** *nume_sau_adresa* – stabileste serverul catre care va fi trimisa operatia de update
- **key** *nume valoare* – stabileste cheia folosita pentru semnarea mesajelor schimbate cu serverul
- **debug** – activeaza modul debug

- **update add** *numeDNS ttl tipRR date* – solicita adaugarea unei inregistrari in fisierul zona
- **update delete** *numeDNS tipRR* – solicita stergerea RRSET-ului solicitat din fisierul zona
- **send –** trimite cererea curenta catre server
- **answer** – afiseaza raspunsul serverului

```
nsupdate -y hmac-md5:cheie1.domeniu.ro.:CkAB6zn3AkZDdzJ/NxZtJw==
> server localhost
> update add www.domeniu.ro. 180 A 10.0.0.5
> send
> update delete www.domeniu.ro. A
> send
```

6.6. Securizarea informatiilor DNS folosind DNSSEC

6.6.1. Principii

Mecanismul TSIG prezentat anterior are doua neajunsuri majore:

- foloseste o cheie secreta comuna, ceea ce ridica problema distributiei cheii secrete intre doua statii care se afla la primul lor dialog
- poate fi folosit numai pentru a autentifica comunicatia intre servere DNS sau intre servere si clienti, insa nu poate garanta autenticitatea informatiilor cuprinse in mesaj. Daca un client interogheaza un server DNS care a fost compromis, sau daca doar cheia secreta comuna a fost compromisa, semnatura digitala a mesajelor de raspuns primite va fi corecta, insa inregistrarile receptionate pot fi falsificate fara ca clientul sa realizeze

Pentru a putea autentifica informatia cuprinsa in interiorul zonelor a fost creat DNSSEC (DNS Security Extensions), care se bazeaza pe criptografie in cheie publica si care permite unui administrator sa semneze digital inregistrarile zonelor sale. Pentru semnare se foloseste o cheie privata a zonei; cheia publica corespunzatoare este expusa ca parte a zonei si este autentificata la randul sau folosind inregistrari plasate in domeniul parinte. Se creeaza astfel o ierarhie de incredere asemanatoare cu cea a CA-urilor, care culmineaza cu zona radacina (.) . In acest fel, chiar daca un atacator reuseste sa redirectioneze interogarile unei zone catre un server aflat sub controlul sau, in scopul de a furniza clientilor informatii false, el nu poate semna digital inregistrarile falsificate deoarece nu se afla in posesia cheii private a zonei; chiar daca atacatorul genereaza propria pereche de chei si semneaza digital zona falsificata, inregistrarile din domeniul parinte nu-i vor valida semnaturile.

Pentru a implementa acest mecanism au fost necesare multiple adaugiri in sistemul DNS:

- odata cu adaugarea semnaturilor digitale, dimensiunea fisierului zona creste foarte mult, limita traditionala DNS de 512 octeti per mesaj devenind o problema. Rezolvarea vine sub forma extensiei EDNS0 (vezi mai jos)
- pentru a implementa semnaturile digitale si lantul de incredere amintit mai sus au fost necesare noi tipuri de inregistrari DNS (descrise mai jos intr-o sectiune dedicata)
- pentru a gestiona tranzitia de la sistemul actual la unul autentificat cu semnaturi digitale, in timpul careia vor exista multe dialoguri mixte (in care fie serverul, fie clientul nu suporta DNSSEC) sunt necesare noi flag-uri DNS

6.6.2. Extensii DNS necesare: EDNS0 si noi flag-uri DNSSEC

EDNS0 (RFC2671) reprezinta o extensie a protocolului DNS care asigura scalabilitatea/extensibilitatea acestuia (fara a avea legatura initial cu DNSSEC). Modificarea introdusa consta in principal in adaugarea unei noi pseudo-inregistrari de tip OPT (option) in sectiunea “additional” a mesajului DNS.

Meta-inregistrarea OPT contine o portiune de informatie fixa si una variabila. Cea de-a doua este formata dintr-un numar variabil de optiuni sub forma de perechi nume-valoare, ceea ce asigura extensibilitatea; cea fixa contine informatii precum:

- dimensiunea maxima a datagramei UDP suportata de catre expeditorul mesajului
- o zona de 16 flag-uri noi ce urmeaza a fi definite pe masura ce apare necesitatea lor

Un client care suporta EDNS0 va include in mesajele sale catre un server inregistrarea OPT. Daca serverul suporta EDNS0 el va folosi EDNS0 in raspunsul sau; in caz contrar, va ignora pur si simplu inregistrarea OPT.

DNSSEC introduce 3 noi flag-uri DNS:

- doua in headerul DNS original, folosite de catre un server ce raspunde la interogari recursive in comunicatiile cu clientii sai:
  - **AD (Authentic Data)** – atunci cand clientul nu are capabilitati DNSSEC, serverul poate efectua validarea inregistrarilor obtinute in numele clientului, transmitandu-i acestuia rezultatul verificarii impreuna cu raspunsul, prin intermediul flag-ului AD. Daca cel putin o inregistrare din raspuns nu este validata, atunci intregul raspuns va fi considerat ca neautentificat corect.
  - **CD (Checking Disabled)** – prin intermediul acestui flag, clientul poate indica serverului ca suporta DNSSEC, fiind capabil sa faca propria validare a inregistrarilor receptionate. In acest fel, serverul va efectua validari in numele clientilor numai atunci cand este nevoie, scutind resurse
- in zona de flag-uri suplimentare din OPT
  - **DO (DNSSEC OK)** – indica unui server faptul ca clientul suporta DNSSEC si doreste informatii de acest fel incluse in raspuns. Astfel serverele vor oferi informatii DNSSEC numai la cerere, scutind resurse si banda

Nota: *flagul DO se seteaza in dig folosind optiunea +dnssec.*

6.6.3. Noi inregistrari DNS folosite in DNSSEC

6.6.3.1. Inregistrarea de tip DNSKEY

In DNSSEC, fiecare zona are atasata o pereche de chei – una publica si una privata. Cea privata este memorata intr-un fisier cat mai bine protejat, iar cea publica este expusa clientilor sub forma unei inregistrari de tip DNSKEY. Formatul acesteia este asemanator cu cel din exemplul de mai jos:

```
domeniu.ro.  DNSKEY  257  3  5  AwEAAbIfdkH90ON+KHl8dVIAnWK52A8oA8xk7yzsl4nz99T67JTBE+lA
nlDD/MHIXPXrydeky9JmIXp7ch+zjdW5CfE=
```

Campurile au urmatoarele semnificatii:

- **domeniu.ro** – numele zonei careia ii corespunde cheia
- **DNSKEY** – tipul de inregistrare
- **257** – flag-urile asociate inregistrarii. Reprezinta o valoare pe 16 biti, din care la ora actuala doar doi biti au primit semnificatii:
  - **Zone Key (ZK)** - bitul 7 (corespunzator valorii 256). Indica daca cheia este una folosita pentru a verifica semnaturi de zona (inregistrari RRSIG) sau un alt tip de cheie publica (ex: pentru securizare update/transfer)
  - **Secure Entry Point (SEP)** – bitul 15 (corespunzator valorii 1). Indica scopul cheii – pentru semnare de inregistrari sau semnarea cheii zonei (vezi mai jos sectiunea despre KSK/ZSK)
- **3** – specifica protocolul folosit. Reprezinta o reminiscenta din fazele incipiente ale elaborarii specificatiei DNSSEC. La ora actuala campul trebuie sa aiba valoarea 3
- **5** – indica algoritmul criptografic folosit pentru generarea cheii. In acest caz, 5 reprezinta RSA/SHA-1 (care este obligatoriu), dar sunt posibile si variante aditionale precum RSA/MD5, DSA/SHA1 etc
- ultimul camp este reprezentarea codata base64 a cheii

In practica se folosesc pentru fiecare zona doua chei private (plus perechile lor publice):

- **ZSK** (zone signing key) – este cheia cu care se genereaza semnaturile inregistrarilor zonei. Cheia publica corespunzatoare este publicata sub forma unei inregistrari DNSKEY care are bitul ZK setat si bitul SEP pus pe 0
- **KSK** (key signing key) – este cheia folosita pentru a semna ZSK-ul. Cheia publica corespunzatoare este plasata in cadrul zonei intr-o inregistrare DNSKEY care are bitii ZK si SEP setati. Scopul folosirii acestei chei suplimentare este de a usura schimbarea cheii de semnare a zonei (care trebuie efectuata periodic, din motive de securitate). Detalii suplimentare la sectiunea dedicata inregistrarii DS.

6.6.3.2. Inregistrarea de tip RRSIG

Inregistrarile de tip RRSIG memoreaza semnaturile digitale asociate informatiilor din fisierul zona. Semnarea nu se face pentru fiecare inregistrare in parte, ci per RRSET[2] – in fond, un client care interogheaza serverul ii furnizeaza acestuia numele DNS si tipul inregistrarilor dorite, obtinand astfel intregul RRSET asociat celor doua informatii (nu exista posibilitatea de a solicita o anumita inregistrare din cadrul RRSET-ului).

Inregistrarea RRSIG are formatul din exemplul de mai jos. Consideram cazul unei zone care contine doua inregistrari de tip A cu numele *www* (o implementare simpla de load sharing pentru web):

```
www.domeniu.ro.                 A       1.2.3.4
www.domeniu.ro.                 A       5.6.7.8
www.domeniu.ro.     86400       RRSIG     A 5 3 86400 20100802051501 (
                                          20100703051501 61785 domeniu.ro.
                                          Oq/LHIEX5GEbO9qygDCynCxONXbaeU7wW5Om
                                          uqdst3FGVhy5ptshPs6eC1ixV7KAnKF0smmJ
                                          YaymirPFVlYIbw== )
```

Iata semnificatiile campurilor, in ordinea in care apar ele in inregistrare:

- **www.domeniu.ro.** - numele DNS, identic cu al RRSET-ului pentru care a fost calculata semnatura
- **86400 RRSIG** – TTL-ul inregistrarii RRSIG si tipul de RR
- **A** – indica tipul RR-uri ce compun RRSET-ul pentru care s-a calculat semnatura
- **5** – indica algoritmul folosit pentru generarea semnaturii (in acest caz RSA/SHA-1)
- **3** – indica numarul de etichete text ce compun numele DNS al RRSET-ului. In cazul nostru avem 3 astfel de etichete (www, domeniu si ro). Numarul de etichete din RRSIG poate diferi de cel din nume in cazul inregistrarilor DNS de tip wildcard (cele care folosesc metacaracterul *)
- **86400** – indica TTL-ul original al RRSET-ului. Informatia este necesara clientilor care valideaza semnatura
- **20100802051501** – data de expirare a semnaturii. Este prezentata in formatul AAAALLZZOOMMSS. Odata ce o semnatura a expirat, ea nu mai este considerata valida si clientii vor inceta sa mai considere valide inregistrarile acoperite de acea semnatura. Avand in vedere ca semnaturile expira, zonele trebuie re-semnate periodic
- **20100703051501** – data generarii semnaturii
- **61785** – asa-numitul *key tag*. Reprezinta o amprenta a cheii publice corespunzatoare celei private cu care s-a semnat zona
- **domeniu.ro.** - asa-numitul *domain signer*. Reprezinta numele DNS al cheii cu care trebuie facuta verificarea semnaturii. Informatia este necesara clientului care valideaza semnatura pentru a putea obtine cheia publica corespunzatoare celei private cu care s-a generat semnatura; daca numele DNS in cauza are mai multe inregistrari DNSKEY corespondente, clientul va folosi *key tag* pentru a determina cheia corecta
- ultimul camp reprezinta semnatura digitala, codata base64

6.6.3.3. Inregistrarea de tip DS

Inregistrarile RRSIG nu ar creste cu nimic securitatea DNS daca nu ar exista un mecanism prin care clientul sa se asigure ca cheia publica a zonei (inregistrarea DNSKEY pe care o obtine prin interogare) este cea legitima si nu a fost inlocuita de catre un atacator. Aceasta presupune ca cineva sa garanteze pentru cheile publice ale unei zone – si cine ar fi mai potrivit decat zona parinte, cea care face delegarea?

In acest scop au fost introduse inregistrarile de tip DS (Domain Signer), care fac parte din zona parinte si garanteaza pentru cheile domeniilor catre care se face delegarea. Inregistrarea DS contine un hash al cheii pentru care garanteaza. Iata cum ar putea arata o astfel de inregistrare in zona *domeniu.ro* daca exista o delegare facuta catre *sub.domeniu.ro*:

2 Un RRSET reprezinta setul de inregistrari ale unei zone care au acelasi nume, tip de RR si clasa

```
sub.domeniu.ro.            DS 11476 5 1 79258DF4CCF082C3DF4B10280AADCD74B072A332
sub.domeniu.ro.          RRSIG   DS 5 2 86400 20060219234934 (
                                        20060120234934 23912 edu.
                                        Nw4xLOhtFoP0cE6ECIC8GgpJKtGWstzk0uH6
                                        nd2cz28/24j4kz1Ahznr/+g5oU3AADyv86EK
                                        CnWZtyOeqnfhMZ3UW0yyPcF3wy73tYLQ/KjN
                                        gPm1VPQA/Sl3smauJsFW7/YPaoQuxcnREPWf
                                        YWInWvWx12IiPKfkVU3F0EbosBA= )
```

Semnificatiile campurilor:

- **11476** – key tag - amprenta a cheii autorizate sa semneze inregistrarile din subdomeniu. Subdomeniul poate avea mai multe chei de natura diferita (ex: RSA/SHA1, RSA/MD5 etc)
- **5** – algoritmul folosit pentru generarea cheii, la fel ca in RRSIG
- **1** – algoritmul de hashing folosit pentru a obtine ultimul camp al inregistrarii DS
- ultimul camp reprezinta hash-ul cheii pentru care se garanteaza, codat base64

Dupa cum se observa, inregistrarea DS din zona parinte este insotita de propria inregistrare RRSIG, garantandu-se astfel pentru autenticitatea sa.

Prin acest mecanism se creeaza un lant al increderii care incepe din zona radacina. Zona root contine inregistrarile DS care garanteaza pentru cheile TLD-urilor, TLD-urile la randul lor garanteaza pentru nivelul imediat inferior etc. Un client care doreste sa valideze semnatura unui set de inregistrari primite va urma acest lant pana la intalnirea unui server de root in care are incredere. Fiecare administrator care introduce DNSSEC pentru o zona trebuie sa se adreseze administratorilor zonei parinte pentru a introduce inregistrarile DS corespunzatoare cheii sale.

Din motive de securitate, cheia de semnare a unei zone trebuie schimbata periodic (cu cat zona este mai bogata in informatie, cu atat mai mult material de analiza criptografica este disponibil atacatorilor). Daca o zona ar folosi o singura pereche de chei, atunci la fiecare schimbare de cheie trebuie modificata si inregistrarea DS din zona parinte. Pentru a evita acest lucru a fost introdus tandemul KSK-ZSK: inregistrarea DS garanteaza pentru KSK, iar KSK-ul este folosit pentru a semna ZSK-ul. Toate inregistrarile sunt semnate folosind ZSK, asadar KSK este mult mai putin vulnerabil si deci trebuie schimbat mult mai rar; pe de alta parte, ZSK poate fi acum schimbat de catre administratorul zonei fara a mai implica serverele zonei parinte, prin simpla generare a unei noi chei si semnarea ei folosind KSK-ul.

6.6.3.4. Inregistrarea de tip NSEC

Inregistrarile de pana acum lasa neacoperit un aspect: autentificarea raspunsurilor negative, care indica absenta unui nume DNS. Avand in vedere ca astfel de raspunsuri nu au un element distinctiv, care sa le diferentieze unele de altele (cum se intampla in cazul celor pozitive) si care sa poata fi semnat digital, a fost introdusa o noua inregistrare care sa permita semnarea digitala a unui raspuns negativ.

Inregistrarea NSEC (Next SECure) este folosita pentru a indica “golurile” dintre doua nume DNS consecutive ale unei zone. Fiecare inregistrare a unei zone dispune de un RR pereche de tip NSEC, care indica numele urmatoarei inregistrari existente. Atunci cand un client solicita un nume DNS inexistent, va fi returnata inregistrarea NSEC corespunzatoare golului in care se incadreaza numele cerut. Spre exemplu, daca intr-o zona exista numele *a.domeniu.ro* si *c.domeniu.ro*, intre ele va exista o inregistrare NSEC care atesta faptul ca intre a si c nu mai exista alte nume; un client care cere *b.domeniu.ro* va primi aceasta inregistrare NSEC, semnata digital.

Implementarea acestui mecanism presupune o ordine clara a inregistrarilor zonei. Numele zonei sunt ordonate canonic luand pe rand fiecare eticheta componenta, de la dreapta la stanga, si ordonand alfabetic dupa ea. Spre exemplu, daca in zona domeniu.ro ar exista domeniu.ro, a.domeniu.ro, a.x.domeniu.ro si m.domeniu.ro, ordinea lor canonica ar fi:

```
domeniu.ro.
a.domeniu.ro.
m.domeniu.ro.
a.x.domeniu.ro.
```

Pentru domeniu.ro se va genera o inregistrare NSEC care atesta ca urmatoarea inregistrare este a.domeniu.ro; pentru a.domeniu.ro, un alt NSEC care refera m.domeniu.ro etc. :

```
domeniu.ro.          30      NSEC    a.domeniu.ro. NS SOA RRSIG NSEC DNSKEY
domeniu.ro.          30      RRSIG   NSEC 5 2 30 20100804093445 (
                                         20100705093445 22980 domeniu.ro.
                                         GyH1ZTZTzIhO35UkXyVwEBXc2/+I7NB5ClDr
                                         WsHJRTA/9lrH1mNWeZySDRhjVHATDJwaRjsd
                                         eFy/k857Lp8ZNg== )
a.domeniu.ro.        30      NSEC    m.domeniu.ro. A RRSIG NSEC
                     30      RRSIG   NSEC 5 3 30 20100804093445 (
                                         20100705093445 22980 domeniu.ro.
                                         MZqmbFLkTcLhcG2J7lvnOJMxqMoiuIZc0PEV
                                         DQw3ylcf7I1LDrVZN/RlFXdoXChVlvP+FZHF
                                         BoJrhBid4Uy8VA== )
```

6.6.4. Implementare in BIND

6.6.4.1. Etape

Un administrator care doreste sa semneze digital o zona DNS trebuie sa parcurga urmatoarele operatii:

- generarea cheilor zonei:
  - KSK – are flag-ul SEP setat
  - ZSK – are flag-ul SEP resetat
- includerea cheilor publice corespunzatoare in cadrul zonei sub forma de inregistrari DNSKEY
- semnarea zonei folosind ZSK. Se realizeaza in BIND in mod automatizat, folosind utilitarul *dnssec-signzone*, care efectueaza:
  - generarea inregistrarilor NSEC
  - generarea semnaturilor pentru toate RRSET-urile zonei
  - (optional) generarea inregistrarilor DS care trebuie plasate in zona parinte
- transmiterea KSK-ului catre zona parinte pentru includerea inregistrarilor DS

6.6.4.2. Generare chei si includere in zona

Generarea cheilor se realizeaza in BIND cu utilitarul *dnssec-keygen* dupa cum urmeaza:

```
# ZSK-ul
dnssec-keygen -a rsasha1 -n zone -b 1024 domeniu.ro.

# KSK-ul
dnssec-keygen -a rsasha1 -n zone -b 1024 -f ksk domeniu.ro.
```

Atentie! Cheile DNSSEC trebuie sa aiba nume identic cu al zonei pentru care sunt generate! (spre deosebire de cele TSIG al caror nume poate fi ales de catre administrator)

Nota: *pe unele distributii este necesara folosirea optiunii* -r /dev/urandom *pentru a specifica sursa de informatii aleatoare necesare (de utilizat daca comanda de generare consuma foarte mult timp)*

Comanda creeaza doua fisiere in directorul curent – structura numelui lor a fost prezentata cu ocazia TSIG. Fisierul cu extensia .key contine cheia publica sub forma inregistrarii DNSKEY corespunzatoare, iar cel cu extensia .private cheia privata.

Nota: *cheia privata ii este necesara serverului numai daca folosim dynamic update impreuna cu dnssec; in aceste conditii serverul trebuie sa poata semna inregistrarile nou-introduse si inregistrarile NSEC care se modifica ca urmare a modificarilor aduse zonei.*

Includerea cheilor in zona se poate realiza prin simpla copiere a informatiei din fiserele .key in fisierul zona:

```
cat Kdomeniu.ro.+005+*.key >> domeniu.ro.zone
```

6.6.4.3. Generarea inregistrarilor NSEC si semnarea zonei

Ambele operatii sunt efectuate de catre utilitarul **dnssec-signzone**. Desi exista diferite modalitati de a-l apela, pentru simplitate preferam urmatorul set de conditii:

- in fisierul zona trebuie sa existe inregistrarile DNSKEY corespunzatoare cheilor KSK si ZSK
- fisierele .key si .private corespunzatoare celor doua chei vor fi plasate in directorul curent

Odata conditiile indeplinite, semnarea zonei poate fi efectuata cu comanda:

```
dnssec-signzone -o domeniu.ro domeniu.ro.zone
```

unde domeniu.ro.zone este numele fisierului zona, iar -o specifica origin-ul (numele zonei) atunci cand numele fisierului zona difera de al zonei. Utilitarul determina automat care chei sunt KSK/ZSK analizand flag-ul SEP.

In urma semnarii se genereaza urmatoarele fisiere:

- un fisier care poarta acelasi nume ca fisierul zona dar are extensia .signed (*domeniu.ro.zone.signed* in cazul exemplului de mai sus). Acesta este noul continut al zonei si il va inlocui pe cel vechi
- un fisier keyset-*numezona* (in cazul nostru, *keyset-domeniu.ro*) care contine cheile KSK ale zonei
- un fisier dsset-*numezona* (in cazul nostru, *dsset-domeniu.ro*) care contine inregistrarile DS care ar trebui introduse in zona parinte

Anexa 2 a materialului contine un exemplu complet de zona semnata.

Odata semnata zona, serverul trebuie determinat sa foloseasca noul continut: fie se redenumeste fisierul .signed, fie se editeaza named.conf pentru a pointa catre noul fisier zona. In ambele cazuri este necesara recitirea fisierului de configurare (ex: rndc reload).

6.6.4.4. Configurare BIND

Toate elementele de pana acum au afectat in principal fisierele zona. Pentru ca BIND sa se foloseasca de noile informatii si sa includa elemente DNSSEC in raspunsurile catre clienti este necesara activarea acestei facilitati:

```
options{
      dnssec-enable yes;
};
```

In plus, in cazul in care se foloseste DNSSEC impreuna cu facilitatea de dynamic update, serverul trebuie sa aiba acces la cheia privata a zonei pentru a putea semna inregistrarile care se modifica. Directiva *key-directory* ii indica serverului calea absoluta catre locul in care se gasesc fisierele pentru cheia privata si publica; in plus, perioada de validitate a semnaturii generate poate fi specificata cu optiunea *sig-validity-interval*:

```
options{
      sig-validity-interval 14;        # masurat in zile
      key-directory "/cale/absoluta";
};
```

In sfarsit, pentru a declara trust anchor (setul de chei de unde incepe lantul increderii – asemanator cu conceptul de root CA intalnit cu o alta ocazie) folosim directiva trusted-keys, in care fiecare cheie este listata intr-un format foarte asemanator celui din inregistrarea DNSKEY (lipsesc doar campurile ce reprezinta clasa si tipul RR-ului):

```
trusted-keys{
```

```
        nume_cheie    flaguri       protocol      algoritm      "continut_cheie_base64"
};
```

Nota: *directiva trusted-keys{} poate fi folosita pentru a declara ca trusted orice chei, nu neaparat cele ale serverelor de root.*

Iata un exemplu:

```
trusted-keys{
        ro.    257    3      5      "AwEAAazLVnPM7B5uPBZbm25vqhHTY1dAjL9L6VOiUMrYv3hN4m1oChFD";
};
```

6.7. BIBLIOGRAFIE

- BIND ARM: http://ftp.isc.org/www/bind/arm95/Bv9ARM.html
- Secure BIND template: http://www.cymru.com/Documents/secure-bind-template.html
- Atacuri DNS: http://www.mavensecurity.com/documents/J9-DNS-Security-handouts-2008-02-17.pdf
- Site-ul DNSSEC: www.dnssec.net
- RFC 2845 (TSIG) - http://tools.ietf.org/html/rfc2845
- RFC 2930 (TKEY) - http://tools.ietf.org/html/rfc2930
- RFC 2671 (EDNS0) - http://tools.ietf.org/html/rfc2671
- RFC 4034 (RR-uri DNSSEC) - http://tools.ietf.org/html/rfc4034
- RFC 4035 (Modificari ale protocolului DNS pentru DNSSEC) - http://tools.ietf.org/html/rfc4035
- Explicare NSEC si NSEC3: http://blog.dest-unreach.be/2010/01/20/dnssec-the-nsec-and-nsec3-record
- Urmarirea stadiului semnarii zonei root: http://www.root-dnssec.org/
- Exemple de implementare DNSSEC:
  - http://www.dyndns.com/support/kb/implementing_dnssec.html
  - http://garnser.blogspot.com/2008/02/how-to-enable-bind-with-dnssec-and.html

6.8. ANEXA 1. Categorii de logging in BIND 9

(conform BIND 9 ARM)

default	The default category defines the logging options for those categories where no specific configuration has been defined.
general	The catch-all. Many things still aren't classified into categories, and they all end up here.
database	Messages relating to the databases used internally by the name server to store zone and cache data.
security	Approval and denial of requests.
config	Configuration file parsing and processing.
resolver	DNS resolution, such as the recursive lookups performed on behalf of clients by a caching name server.
xfer-in	Zone transfers the server is receiving.
xfer-out	Zone transfers the server is sending.
notify	The NOTIFY protocol.
client	Processing of client requests.
unmatched	Messages that named was unable to determine the class of or for which there was no matching view. A one line summary is also logged to the client category. This category is best sent to a file or stderr, by default it is sent to the null channel.
network	Network operations.
update	Dynamic updates.
update-security	Approval and denial of update requests.
queries	Specify where queries should be logged to. At startup, specifying the category queries will also enable query logging unless querylog option has been specified. The query log entry reports the client's IP address and port number, and the query name, class and type. It also reports whether the Recursion Desired flag was set (+ if set, - if not set), if the query was signed (S), EDNS was in use (E), if DO (DNSSEC Ok) was set (D), or if CD (Checking Disabled) was set (C). client 127.0.0.1#62536: query: www.example.com IN AAAA +SE client ::1#62537: query: www.example.net IN AAAA -SE
dispatch	Dispatching of incoming packets to the server modules where they are to be processed.
dnssec	DNSSEC and TSIG protocol processing.
lame-servers	Lame servers. These are misconfigurations in remote servers, discovered by BIND 9 when trying to query those servers during resolution.
delegation-only	Delegation only. Logs queries that have have been forced to NXDOMAIN as the result of a delegation-only zone or a delegation-only in a hint or stub zone declaration.
edns-disabled	Log queries that have been forced to use plain DNS due to timeouts. This is often due to the remote servers not being RFC 1034 compliant (not always returning FORMERR or similar to EDNS queries and other extensions to the DNS when they are not understood). In other words, this is targeted at servers that fail to respond to DNS queries that they don't understand. Note: the log message can also be due to packet loss. Before reporting servers for non-RFC 1034 compliance they should be re-tested to determine the nature of the non-compliance. This testing should prevent or reduce the number of false-positive reports. Note: eventually named will have to stop treating such timeouts as due to RFC 1034 non compliance and start treating it as plain packet loss. Falsely classifying packet loss as due to RFC 1034 non compliance impacts on DNSSEC validation which requires EDNS for the DNSSEC records to be returned.

6.9. ANEXA 2. Exemplu de fisier zona semnat

```
domeniu.ro.             30      IN SOA  ns1.domeniu.ro. admin.domeniu.ro. (
                                        2010070401 ; serial
                                        300        ; refresh (5 minutes)
                                        30         ; retry (30 seconds)
                                        30000      ; expire (8 hours 20 minutes)
                                        30         ; minimum (30 seconds)
                                        )
                        30      RRSIG   SOA 5 2 30 20100804084448 (
                                        20100705084448 22980 domeniu.ro.
                                        iYzCPFj9sPh1sNtaRL0KAJ8TriEHuJrPO2qs
                                        TK65D8R3oh8FvUtNJxw0Sj+57vicbrd14J+n
                                        cCgEfs5NU0jZ+Q== )
                        30      NS      ns1.domeniu.ro.
                        30      RRSIG   NS 5 2 30 20100804084448 (
                                        20100705084448 22980 domeniu.ro.
                                        SVptAsJwdY8B9MUGZNeYiKe54XpPVZoo5fqp
                                        YDyG7ldcetWhShkVHsVLJNEcYQx6HJlsHmbs
                                        msDAOcjKdma9VQ== )
                        30      NSEC    ns1.domeniu.ro. NS SOA RRSIG NSEC DNSKEY
                        30      RRSIG   NSEC 5 2 30 20100804084448 (
                                        20100705084448 22980 domeniu.ro.
                                        HgH94jrsumFwOpNmFPAH1BGTVJyxRwgudH0g
                                        XCIXYAGP5BozMiArHFP0J8HBxfBqw7/Qutjs
                                        y3YudMOMxuptmA== )
                        30      DNSKEY  256 3 5 (
                                        AwEAAazLVnPM7B5uPBZbm25vqhHTY1dAjL9L
                                        6VOiUMrYv3hN4m1oChFDNMa+y7AlfDsIprGC
                                        VP9gDVprbQsHL0rqtrU=
                                        ) ; key id = 22980
                        30      DNSKEY  257 3 5 (
                                        AwEAAbX7QyoxgaXPG5X6jKdRoLdON2ix0UAm
                                        V28hLgXAH2lowMJ+6ZoH9iHhTV+IV/awfo61
                                        oqV8zE5ke6UB3rJv81c=
                                        ) ; key id = 58321
                        30      RRSIG   DNSKEY 5 2 30 20100804084448 (
                                        20100705084448 22980 domeniu.ro.
                                        SjIuV6oBdoLrFdPx5snFeQiCAXqvljyvtujt
                                        ArmsQN3NoDiF2OD5MOGvn36ctzsYTnkGmDpr
                                        AvQZ4loN/ToBiQ== )
                        30      RRSIG   DNSKEY 5 2 30 20100804084448 (
                                        20100705084448 58321 domeniu.ro.
                                        cMGqkrJRp94CJA3KEYrpeLRb4L/GFBZBu+3Z
                                        8+76Ot9h3YnXICjWzmSy7qXeewDa3Fjl3GQS
                                        nNsEZxM8FccQeg== )
ns1.domeniu.ro.         30      IN A    10.0.0.3
                        30      RRSIG   A 5 3 30 20100804084448 (
                                        20100705084448 22980 domeniu.ro.
                                        ekkK1I7Rkz6wWhZkvGf3GeoJ2BAzkuvmJyFF
                                        teEQ+V52GDQfJO4KxkInoNzFS023feuzug5r
                                        c1DtlYYC+RoNeg== )
                        30      NSEC    domeniu.ro. A RRSIG NSEC
                        30      RRSIG   NSEC 5 3 30 20100804084448 (
                                        20100705084448 22980 domeniu.ro.
                                        qrLxnpMx3BmNKiOA84LNtYHW5Pt3xIxWsIm+
                                        wv4R3t7ZjdVLM66ZUuvNIvEuT8agV5iy8N6C
                                        fHrWhwXJWKDOqQ== )
```