DJBDNS – Élet a Bind-en túl

Az Internet infrastruktúrájának kritikus eleme a DNS szolgáltatás, amelyet jellemzően az ISC BIND alkalmazásával valósítanak meg. Nem mindenki elégedett azonban ezzel.

A ***djbdns*** talán az első olyan alkalmazás, amely pénzgaranciát tartalmaz. Az alkalmazás írója, ***D.J. Bernstein***, 500 dolláros díjat ajánlott fel annak, aki kihasználható biztonsági hibát talál benne. Úgy tudom, a pénz még mindig nála van. A ***bind*** egy nagy, monolitikus alkalmazás (minden funkciót egy alkalmazás végez), ebből következően túl bonyolult (***Bernstein*** 300,000 kódsorról beszél; összehasonlításként a ***Solaris*** forráskódja hosszú ideje stabilan 7 millió sor), a konfigurációs fájl formátuma nehézkes, egyetlen pont hiánya nagy kavarodás okozója lehet, és a biztonsága sem győzött meg minden kételkedőt. Bernstein egy írásában (➲ ***http://cr.yp.to/djbdns/blurb/unbind.html***) éles kritikával illeti a ***bind***-et, a jelenleg domináns ***DNS*** implementációt, ill. annak hibáit. Tervezési hibákat említ, szerinte a ***BIND 9*** gyakrabban összeomlik, mint a ***BIND 8***, a ***BIND 9*** módosításnaplójában (***Changelog***) kereken 672 hiba javítását találta, ami szerinte nem fér össze egy robusztusnak mondott rendszerrel.
Ha az olvasó egy egyszerűen használható, moduláris, nagy teljesítményű és biztonságos ***DNS*** implementációt szeretne, ne keressen tovább: a ***djbdns*** mindent tud, ami ma egy ***DNS*** kiszolgálótól elvárható.
Az alábbiakban bemutatom, hogyan lehet egy fiktív vállalatnál 2 ***DNS*** kiszolgálót építeni, amely nem csak a cég tartományát kezeli, de névfeloldást is biztosít a belső felhasználóknak (*1. ábra*). Legyen a cég tartományának neve ***xxxx.hu***, az ***1.2.3.4*** és ***1.2.3.5*** ***IP***-címeken szolgálja ki az ún. autoritatív kéréseket, amelyek közül legyen az ***1.2.3.4*** az elsődleges kiszolgáló, a cég belső felhasználóinak pedig a ***10.1.1.4*** és ***10.1.1.5*** címeken biztosít névfeloldást. Üzembiztonsági okok miatt a másik kiszolgálót célszerű egy másik hálózaton (például ***co-location***) elhelyezni, így a másodlagos kiszolgáló ***IP***-címe más lenne (és kellene egy újabb gép a cég hálózatán belül, amelyik a belső felhasználóknak névfeloldást biztosít), de ezzel most a példában nem foglalkozom, a telepítés szempontjából mindegy.
Telepítsük először a ***daemontools*** (➲ ***http://cr.yp.to/daemontools/install.html***) ill. az ***ucspi-tcp*** (➲ ***http://cr.yp.to/ucspi-tcp/install.html***) csomagot (ezeken az oldalakon részletesen le van írva a telepítésük), majd töltsük le a ***djbdns*** legutolsó verzióját (➲ ***http://cr.yp.to/djbdns/install.html***), és hajtsuk végre az alábbi parancsokat:

```
tar zxvf djbdns-1.05.tar.gz
cd djbdns-1.05
echo gcc -O2 -include /usr/
➥include/errno.h > conf-cc
make
su -c 'make setup check'
```

Hozzunk létre három felhasználót a ***djbdns*** programjainak futtatásához:

```
groupadd dnslog
groupadd dnscache
groupadd tinydns
useradd -g dnslog -s /bin/false
➥dnslog
useradd -g dnscache -s /bin/
➥false dnscache
useradd -g tinydns -s /bin/
➥false tinydns
usermod -L dnslog
usermod -L dnscache
usermod -L tinydns
```

Konfiguráljuk először a névfeloldást:

```
dnscache-conf dnscache dnslog
➥/opt/dnscache 10.1.1.4
ln -s /opt/dnscache /service
touch /service/dnscache/root/
➥ip/10.1.1
```

És készen is vagyunk az egyik gépen! A

```
svstat /service/dnscache
```

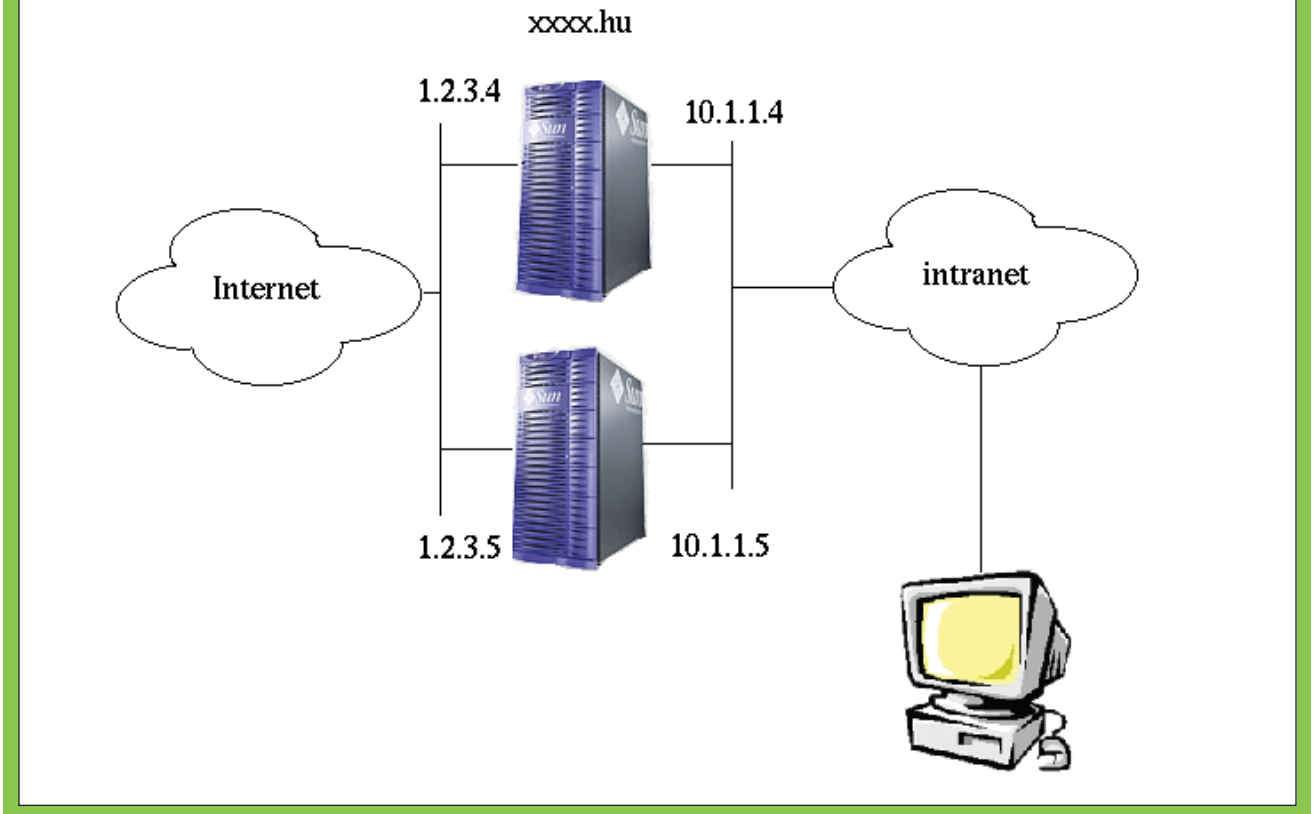
parancs kiadásával meggyőződhetünk, hogy a `dnscache` valóban fut, az én gépemen ezt a választ kaptam:

```
/service/dnscache: up (pid 237)
➥1556402 seconds
```

Az utolsó sor pedig azt jelenti, hogy a ***10.1.1.0/24*** hálózat gépei használhatják a gépen futó `dnscache`-t. Igény szerint több címtartományt illetve gépet is felsorolhatunk.
Opcionálisan hangolhatjuk a `dnscache`-t, például beállíthatjuk a gyorsítótár (cache) méretét 4 MB-ra:

```
echo 4096000 > /service/
➥dnscache/env/CACHESIZE
```

A `dnscache` az ügyfelektől induló lekérdezések kiszolgálásakor először mindig a gyökér kiszolgálókat kérdezi meg – amelyek a */service/dnscache/root/servers/@* fájlban vannak felsorolva – és a kapott úton megy végig



1. ábra

addig a *DNS* kiszolgálóig, amely tudja a választ a megadott kérésre. Így elkerülhető az úgynevezett *cache-poisoning* (gyorsítótár szennyezés), amikor valaki érvénytelen adatok tárolására bírja rá a kiszolgálónkat (➲ *http://en.wikipedia.org/wiki/DNS_cache_poisoning*). A részeredményeket gyorsítótárba teszi, így nem jár be egy utat többször feleslegesen. Csak rekurzív kéréseket fogad el. Következő lépésben indítsuk el az autoritatív névszerver szolgáltatást, hogy le lehessen kérdezni az *xxxx.hu* tartomány gépeinek címét!

```
tinydns-conf tinydns dnslog
➥/opt/tinydns 1.2.3.4
ln -s /opt/tinydns /service
```

Fontos! Nem telepíthetjük a `dnscache` és a `tinydns` kiszolgálót ugyanarra az *IP*-címre, mert mindkettő az *udp/53*-as portot használja. A nehezén már túl is vagyunk. Az előbbihez hasonlóan lekérdezhetjük az alkalmazásunk állapotát. A másik gépen is végezzük el ezeket a beállításokat a megfelelő *IP*-címekkel. Következő lépésben hozzuk létre a cégünk tartományát, és adjunk hozzá néhány *DNS* rekordot (`RR`). Ezt megtehetjük a *djbdns* segédprogramjaival vagy egy szövegszerkesztő segítségével (például *vi*).

```
cd /service/tinydns/root
vi data
make
```

Amint az látható, a *data* fájl egyszerű szöveges állomány. Nem nehéz felismerni a példában szereplő rekordok típusát. Az 1. sorban a zóna *SOA* rekordját láthatjuk, és egy `Z` jel vezeti be.
A pont (`.`) *NS* rekordokat jelöl, míg az = A rekordokat. Tehát a 2-5. sorokban regisztráltuk a tartományunk *DNS* kiszolgálóit. A 6-7. sorokban a levelezést tettük lehetővé: készítettünk egy *MX* rekordot 20-as preferenciával, illetve egy A rekordot a levelező kiszolgálónknak. Végül a cégünk honlapjait tettük elérhetővé, a 8. sorban egy A rekord, a 9-ben pedig egy álnév (alias) szerepel. Végül egy *TXT* rekordot helyeztem el a zónában. A *data* fájl formátumáról bővebben a ➲ *http://cr.yp.to/djbdns/tinydns-data.html* oldalon található információ.

A *tinydns* azonban nem szövegfájlból dolgozik, hanem *CDB* formátumú adatállományból. A szükséges konverziót a `make` paranccsal végezhetjük el. A *djbdns* sokkal hibatűrőbb, mint a *bind*. Ha a *bind* konfigurációs állományában csak egy hibát is vétünk, az jó időre elfoglaltságot biztosít, amíg azt ki nem javítjuk.
A *djbdns* azonban egyszerűen figyelmen kívül hagyja a hibás rekordot, és megy tovább az élet. A *bind*-del ellentétben nem szükséges `HUP` jelzést küldeni vagy más módon értesíteni a kiszolgálót, hogy megváltozott a zóna adata, automatikusan az új adatokkal dolgozik.
A zónánk adatait szinkronizálni szükséges az elsődleges és a másodlagos kiszolgálónk között.
A *bind* ezt jellemzően inkrementális (*IXFR*) vagy teljes (*AFXR*) zónatranszferrel oldja meg. A *djbdns* ezt a feladatot sokkal elegánsabban végzi el, nem találja fel újra a kereket, hanem egy külső eszközzel másolja át a platform független *data.cdb* fájlt a másik gépre: `rsync`-kel vagy egyszerűen `scp`-vel (aligha van adatátvitelre jobb megoldás ennél). Ehhez

1. Lista A data fájl tartalma

```
Zxxxx.hu:ns1.xxxx.hu:hostmast
➥er.xxxx.hu:2005012001:28800:
➥7200:604800:10800::
.xxxx.hu:ns1.xxxx.hu:ns1.xxxx
➥.hu:259200
.xxxx.hu:ns2.xxxx.hu:ns2.xxxx
➥.hu:259200
=ns1.xxxx.hu:1.2.3.4:86400
=ns2.xxxx.hu:1.2.3.5:86400
@xxxx.hu::mx1.xxxx.hu.:20:
➥86400
=mx1.xxxx.hu:1.2.3.6:86400
=www.xxxx.hu:1.2.3.7:86400
+www2.xxxx.hu:1.2.3.7:86400
'xxxx.hu:tinydns really
➥rocks:86400
```

készítsünk el egy jelszó nélküli *SSH* kulcsot, és módosítsuk a *Makefile*-t így:

```
data.cdb: data
  /usr/local/bin/tinydns-data
  scp -i /home/sj/.ssh/
  ➥ssh_kulcs data.cdb
1.2.3.5:/service/tinydns/root
```

Ezután amikor csak aktualizáljuk a `make` paranccsal az elsődleges kiszolgáló adatait, az automatikusan átkerül a másodlagosra. Feladat megoldva, cégünk felhasználóinak névfeloldást végzünk, az egész internet számára pedig kiszolgáljuk az *xxxx.hu* tartományba intézett kéréseket.
Tegyük fel, hogy a cégünknek van egy belső, nem regisztrált zónája is, például *sales.aa*, amelyeket szintén az *1.2.3.4* és *1.2.3.5* gépeken akarunk tartani. Hogyan tudathatjuk a névfeloldást végző gépekkel, hogy ezt a zónát hol keressék? Hozzuk létre a */service/dnscache/root/servers/aa* fájlt, amelybe beírjuk az adott tartomány kiszolgálóit (soronként egyet).
Ha valaki nem olyan szerencsés, hogy a nulláról kezdheti egy *DNS* kiszolgáló építését, annak sem reménytelen a helyzete: a *djbdns* hasznos eszközökkel segíti a migrációt. Vegyük azt az esetet, amikor van egy bind8/9 kiszolgálónk, amelynek az adatait *tinydns*-re akarjuk költöztetni. Tegyük fel, hogy van két tartományunk: *domain1.hu* és *domain2.hu*. Állítsuk be a *bind* kiszolgálón, hogy engedélyezze számunkra az *AXFR* zóna transzfert az *1.2.3.3:53/tcp*-n:

```
options {
    request-ixfr no;
    allow-transfer { 1.2.3.4; };
};
```

```
for i in `domain1.hu
➥domain2.hu`; do
/usr/local/bin/tcpclient
➥1.2.3.3 53 \
/usr/local/bin/axfr-get $i
➥file1 file1.tmp; cat file1 >>
➥data; rm -f file1; done
```

A parancs lefuttatása után megkapjuk a *data* fájlt, amit a `make` paranccsal „fordíthatunk le”, és készen vagyunk.
A naplózással kapcsolatos dolgok a */service/tinydns/log* könyvtárban vannak. A *run* fájl egy héjprogram, ami a naplózó alkalmazást futtatja. Módosítsuk ezt az alábbi módon:

```
#!/bin/sh
exec setuidgid dnslog multilog t
➥2048000 ./main
```

Ennek hatására a `multilog` program 2 MB-onként lerotálja a naplófájlt, így az nem fog a végtelenségig hízni. A modularitás előnye, hogy a naplózást minden *djbdns* programnál (sőt *Bernstein* összes programjánál) a `multilog` végzi, így ugyanezt beállíthatjuk a *dnscache*-nél vagy akár a *qmail*-nél is.
A konkrét naplófájl a */service/tinydns/log/main/current* fájlban található.

```
@4000000043cbb73f05ee267c
➥0a010102:802a:3bce + 000f
➥acts.hu
```

Ez a bejegyzés az *acts.hu MX* rekordjának lekérdezése során keletkezett. Az időbélyeg úgynevezett *TAI64N* formátumban (➲ *http://cr.yp.to/daemontools/tai64n.html*) van az elején. Emlékszik még valaki a 2000. év körüli dátum-mizériára? A *UNIX* 4 byte-os időbélyeg formátuma 2038-ban fog túlcsordulni (de addigra már biztosan 8, 16 esetleg 32 byte-on tárolja a dátumokat a jövő 256-bites processzora). Ezzel ellenben a *TAI64N* formátum már a jelenlegi architektúrákon is több milliárd évet képes ábrázolni. Ezután a kliens *IP*-címe (10.1.1.2) és *UDP* kapuja következik, a *DNS* kérés azonosítója, majd az eredmény (+: ok), végül pedig a lekérdezés típusa (`000f: MX`) és a keresett tartomány név (a *DNS* csomag úgynevezett *DNAME* része). A *djbdns* naplóformátumáról bővebb információ a ➲ *http://dqd.com/~mayoff/notes/djbdns/tinydns-log.html* oldalon található.
Nem olyan régen történt, hogy a *Verisign* egy helyettesítő (*wildcard*) A rekordot helyezett el a *.com* és *.net* tartományokban, amely a *64.94.110.11 IP*-címre mutatott (➲ *http://tinydns.org/*). Ennek hatására minden olyan kérés, amely nem létező tartományra irányult, a *Verisign* gépének címét kapta vissza. Ezért ezt semlegesítendő *Russ Nelson* készített a *djbdns*-hez egy úgynevezett *Verisign*-foltot (➲ *http://tinydns.org/djbdns-1.05-ignoreip.patch*). Aztán kiderült, hogy több más regisztrátor is folytat(ott?) ilyen megkérdőjelezhető gyakorlatot, ezért *Nelson* írt még egy foltot, amivel több címet is figyelmen kívül lehet hagyni (➲ *http://tinydns.org/djbdns-1.05-ignoreip.patch2*). Egy másik lehetőség lehet a védekezésre az, ha a problémás regisztrátorok gyökér névszervereit töröljük a *dnscache* listájából (*/service/dnscache/root/servers/@*), csak aztán maradjon néhány (➲ *http://homepages.tesco.net./~J.deBoynePollard/FGA/verisign-internet-coup.html*).
Tegyük fel, hogy az olvasó még mindig *bind*-et futtat, és ugyanazon az *IP*-címen szolgálja ki a hozzá tartozó tartományokat és az ügyfelek névfeloldás kéréseit. Legyen adott egy támadó, aki több adattal árasztja el a gyorsítótárat, mint amit a *bind* kezelni képes. Ekkor szétszakad nem csak a kimenő, de a bejövő web forgalom is a *bind* zavara miatt (➲ *http://cr.yp.to/djbdns/separation.html*). Nagyban megkönnyíti hát az életet, ha szétválasztjuk a névfeloldást és az autoritatív névszerver szolgáltatást, ahogyan azt az *ISC* is javasolja. De amíg ez korántsem magától értetődő a *bind*-nél, addig a *djbdns* kikényszeríti ezt.
Nagyobb mennyiségű adatnál szükség lehet arra, hogy az egyes zónák adatait külön fájlokban, esetleg adatbázisban tároljuk. A *http://tinydns.org/* oldalon találunk olyan programokat,

1. táblázat *A djbdns csomag segédprogramjai*

`dnsip`	Egy FQDN tartománynevet old fel
`dnsname`	Egy IP-címhez tartozó tartománynevet ad vissza
`dnsname`	Egy IP-címhez tartozó tartománynevet ad vissza
`dnsmx`	Egy tartomány MX kiszolgálóit és azok preferenciáit mutatja meg
`dnstxt`	Megkeresi a tartománynév TXT rekordját
`dnstrace`	Végigkövethetjük vele, mely DNS kiszolgálók ismerhetik az adott rekordot
`dnsq`	Nem rekurzív lekérdezést küld a megadott kiszolgálónak (tinydns tesztelésére)
`dnsqr`	Rekurzív lekérdezést küld a /etc/resolv.conf-ban megadott kiszolgálók felé (dnscache tesztelésére)

amelyekkel kényelmesen, egy böngészőből kezelhetjük a ***tinydns*** adatait, némelyiket kifejezetten többfelhasználós, elosztott környezetbe szánt az írójuk, így tartalmaznak például felhasználó kezelést és különböző jogosultság szinteket.
Bizonyos esetekben szükség lehet arra is, hogy egyes gépek számára zónatranszfert biztosítsunk. Az ***axfrdns*** szolgál erre a célra, és a ***tinydns data.cdb*** állományából dolgozik. Természetesen meg lehet és kell adni azt, hogy mely gépek ***AXFR*** kéréseit szolgálja ki.
A feketelistákat gyakran alkalmazzák a levelezés kapcsán. Az ***rbldns*** kifejezetten erre a célra készült. Abban különbözik a ***tinydns***-től, hogy kizárólag olyan ***DNS*** lekérdezésekre válaszol, amelyek vagy A vagy ***TXT*** rekordra irányulnak, illetve az adatállománya is egyszerűbb formátumú.
A ***tinydns-rrd*** (➲ ***http://develooper.com/code/tinydns-rrd/***) alkalmazás az ***rrdtool*** segítségével képes grafikonokat készíteni a ***djbdns*** terheléséről, amelyek jól jöhetnek egy esetleges hibaelhárítás során is.
A ***djbdns*** csomagban különböző segédprogramok is vannak, amelyek a `host`, `nslookup` és `dig` parancsokat helyettesítik (*1. Táblázat*).
A ➲ ***http://www.fefe.de/dns/*** oldalon található folt segítségével felokosítható a ***djbdns***, hogy kezelni tudja az ***IPv6*** címeket is.
Egy ***DNS*** kiszolgálóval szemben követelmény, hogy könnyen használható, nagy teljesítményű, hibatűrő és biztonságos legyen. Szinte észrevétlenül kell működnie, hogy észre se vegyük a jelenlétét. A ***djbdns*** egy ilyen ***DNS*** implementáció, nekem bevált, ajánlom mindenkinek.

Sütő János
(jsuto@freemail.hu)

1997 óta használ Slackware Linux-ot. Szabadidejében a postfix clapf nevű vírus- és spamszűrőjét polírozza.