# A clapf spamszűrő

Ebben a számban egy saját fejlesztésű, zlib/png licenc (http://www.opensource.org/licenses/zlib-license.php) alatt közreadott, nyílt forrású spamszűrőt mutatok be, amely gyors, könnyen és egyszerűen használható.



A *clapf* (http://freshmeat/projects/clapf/) kezdetben egyszerű antivírus modulként indult, *Postfixhez* kerestem olyan programot, amellyel ki lehet szűrni a vírusos leveleket, és úgy döntöttem, kreativitásomat egy új program elkészítésével fogom kiélni. Később jött az ötlet, hogy legyen képes felismerni és megjelölni a spam-et is. A *clapf* egy olyan démon, amely az nyílt forrású *clamav* (➲ *http://www.clamav.net/*) antivírus könyvtárat használja, és *SMTP* protokoll segítségével kommunikál a *Postfix*-szel (vagy bármely *MTA*-val, amely képes ilyen módon együttműködni vele).

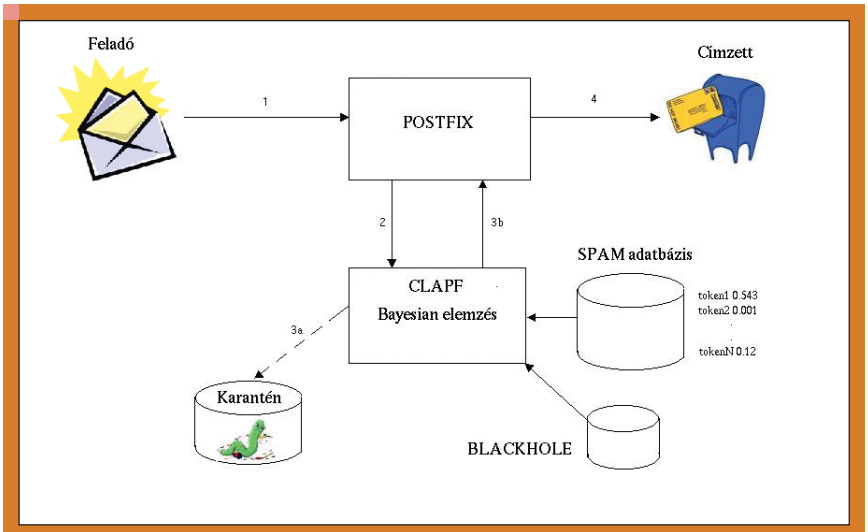
## A telepítés

Az alábbi példában a *clapf* a helyi gép (localhost) *10025/tcp* portján várja a leveleket, és a *10026/tcp* portra adja vissza a *Postfix*-nek. Mielőtt a *clapf* telepítéséhez látnánk, installáljuk fel a *clamav*, *tinycdb* (➲ *http://www.corpit.ru/mjt/tinycdb.html*) csomagokat és a *CDB_File* nevű *Perl* modult (➲ *http://www.cpan.org/*).
Első lépésben telepítsük a *clapf*-ot! Hozzunk létre egy dedikált felhasználót, aki a spamszűrőt futtatni fogja:

```
groupadd av
useradd -g av -s /bin/sh -d
/opt/av av
usermod -L av
```

Töltsük le a program legfrissebb változatát, majd csomagoljuk ki:

```
tar zxvf clapf-0.3.20.tar.gz;
cd clapf-0.3.20
```




*1. ábra* A nagy kép

Igény esetén módosítsuk a *config.h*-ban szereplő változókat, majd fordítsuk le a *clapf*-ot:

```
./configure
make
su -c 'make install'
```

Ha csak spamszűrésre van szükségünk, akkor használjuk a

```
./configure --disable-clamav
```

parancsot.

Hozzuk létre a működéséhez szükséges könyvtárakat:

```
mkdir -p /opt/av/quarantine
chown -R av:av /opt/av
chmod 700 /opt/av
```

A 0.3.19 verziótól kezdve a *clapf* változóinak nagy részét konfigurációs fájlban tárolja. Kiindulási alapként használható a */usr/local/etc/clapf.conf* fájl. Módosítsuk ezt igény szerint, amit megjegyzések bősége segít.
Készítsük el a *Bayesian* elemzéshez szükséges adatbázist. Ehhez szükségünk lesz a *HAM* és *SPAM* nevű *mbox* formátumú (ahol a *Maildir* formátummal ellentétben egyetlen fájlban vannak a levelek) fájlokra, amelyekben csak ham (értékes levél, egyszóval nem spam), ill. csak spam (kéretlen levél) van. Ha ez megvan, akkor az alábbi parancsokkal készítsük el a spam adatbázist (ez valójában egy *CDB* fájl), majd másoljuk azt be a */opt/av* könyvtárba:

```
sh util/kcdb.sh HAM SPAM
mv tokens*.cdb /opt/av
chmod 644 /opt/av/tokens*.cdb
```

Végre, minden készen áll, hogy a spam- és vírusszűrőnk elinduljon (Figyelem! A *clapf*-ot ne futtassuk rendszergazda jogosultságokkal):

```
su - av -c "/usr/local/bin/
➥clapf -c /usr/local/etc/
➥clapf.conf &"
```

A naplófájlban az alábbihoz hasonló üzeneteket kell találnunk:

```
Oct 12 13:26:43 thorium
➥clapf[15851]: reloaded
config: /usr/local/etc/
➥clapf.conf
Oct 12 13:26:43 thorium
➥clapf[15851]: using /opt/av/
➥tokens.cdb as spamicity file
Oct 12 13:26:43 thorium
➥clapf[15851]: clapf 0.3.xx
➥starting
Oct 12 13:26:45 thorium
➥clapf[15851]: reloaded with
➥40544 viruses
```

Ha a *clapf* rendben elindult, módosítsuk a *Postfix* konfigurációs állományait, hogy a leveleket beérkezés után adja át a *clapf*-nak. Ehhez először a */etc/postfix/main.cf* fájlhoz adjuk hozzá az alábbi sort:

```
content_filter =
➥smtp:[127.0.0.1]:10025
```

Majd a */etc/postfix/master.cf* fájlhoz pedig a következőt:

```
127.0.0.1:10026 inet n - n - 10
➥smtpd -o content_filter=
-o receive_override_
➥options=no_address_mappings
```

A *clapf* alapesetben maximum 16 címzettet kezel le (`MAX_RCPT_TO`), amit célszerű a *main.cf*-ben is beállítani (`smtpd_recipient_limit = 16`). Ajánlott limitálni a levelek hosszát is (`message_size_limit`).

Végül indítsuk újra az *MTA*-t az alábbi paranccsal:

```
postfix reload
```

Néhány dolgot nem említettem, hogy például a vírusszignatúrákat rendszeresen (egy konferencián azt javasolták, hogy óránként) frissíteni szükséges. Valahogyan gondoskodni érdemes arról, hogy a *clapf* esetleges leállása esetén újra elinduljon. A *util/* könyvtárban néhány héjprogram (shellszkript) segít ezekben.

## Az üzenet útja a felhasználóig

Nézzük meg, hogy egy bejövő üzenet milyen utat jár végig, mire a felhasználó postaládájába kerül. A levél első lépésben a *Postfix*-hez kerül, amely úgy van konfigurálva, hogy azt először a szűrőhöz irányítja. A *clapf* először vírusellenőrzést végez. Ha egy levélben vírust talál, akkor azt `"550 Access denied"` üzenettel utasítja el, amit a *Postfix* közöl a feladó *SMTP* szerverével (az üzenetben a vírus neve is szerepel). Ebben az esetben véget ért a levél útja. Opcionálisan – ha a konfigurációs fájlban a `use_quarantine=1` szerepel – a vírusos levelet karanténba teszi, ahol a rendszergazda később megnézheti, és dönthet felőle. Ha a levélben nincs ismert vírus, akkor jöhet a spam vizsgálat.

A *clapf* először a *Bayesian* formulát alkalmazza a levél tárgyára (`Subject`) és a levél törzsére. *Paul Graham* szerint (➲ *http://www.paulgraham.com/better.html*) jó, ha a teljes fejléc is szerepel a *Bayesian* elemzésben, mert fontos információk lehetnek annak egyéb részében is. Én azonban úgy gondolkodtam, hogy ott leginkább nyílt proxy-kat, fertőzött zombikat találunk, továbbá az időbélyegeknek, queue azonosítóknak sem látom igazán hasznát ebből a szempontból. Egyelőre ezek nélkül is megfelelő eredménnyel dolgozik a *clapf Bayesian* elemzése. Miután a *clapf* megvizsgálta a levelet, egy extra mezőt – `X-Clapf-spamicity` – tesz a fejlécbe, amelyből kiderül, hogy az mekkora valószínűséggel spam. Ha az érték a nullához van közelebb, akkor ham, ha inkább az egyhez, akkor spam, a 0,5 körüli eredmény pedig bizonytalanságot jelent. Akkor működik jól a szűrő, ha a spam valószínűség értékek (spamicity) a 0,001 ill. 0,999 körül képződnek. Az a levél, amelynek fejlécében például az

```
X-Clapf-spamicity: 0.0123
```

sor található, az nagy valószínűséggel ham. Ha a *clapf* úgy találta, hogy spam-mel került szembe, azaz a valószínűség értéke 0,94-nél (`spam_overall_limit`) nagyobb, akkor beilleszt még egy fejlécet a levélbe a kliens oldali felismerés (például *procmail*, *maildrop*) megkönnyítésére:

```
X-Clapf-spamicity: Yes
```

A *clapf* naplózza minden feldolgozott levél azonosítóját (ez valójában egy *MD5* hexa hash formátumú fájlnév, például `4b91c518b39ddaa9c8cf609d1db467`), spam valószínűség értékét, a levél méretét bájtban és a feldolgozáshoz szükséges időt mikroszekundumban.

A *clapf* nem rendelkezik sor (queue) kezeléssel. Az elején terveztem ezt a funkciót, de a probléma meggyőzött arról, hogy fölösleges újra feltalálni a kereket: a *Postfix* mind a *clapf* előtt, mind pedig utána megoldja ezt a problémát.

Szintén egy konferencián hallottam olyan antispam termékről, amely – ha nagyon biztos a dolgában – eldobja a levelet. Ennek megvan az az előnye, hogy a felhasználó postaládájába be sem jut a *spam* – ez különösen

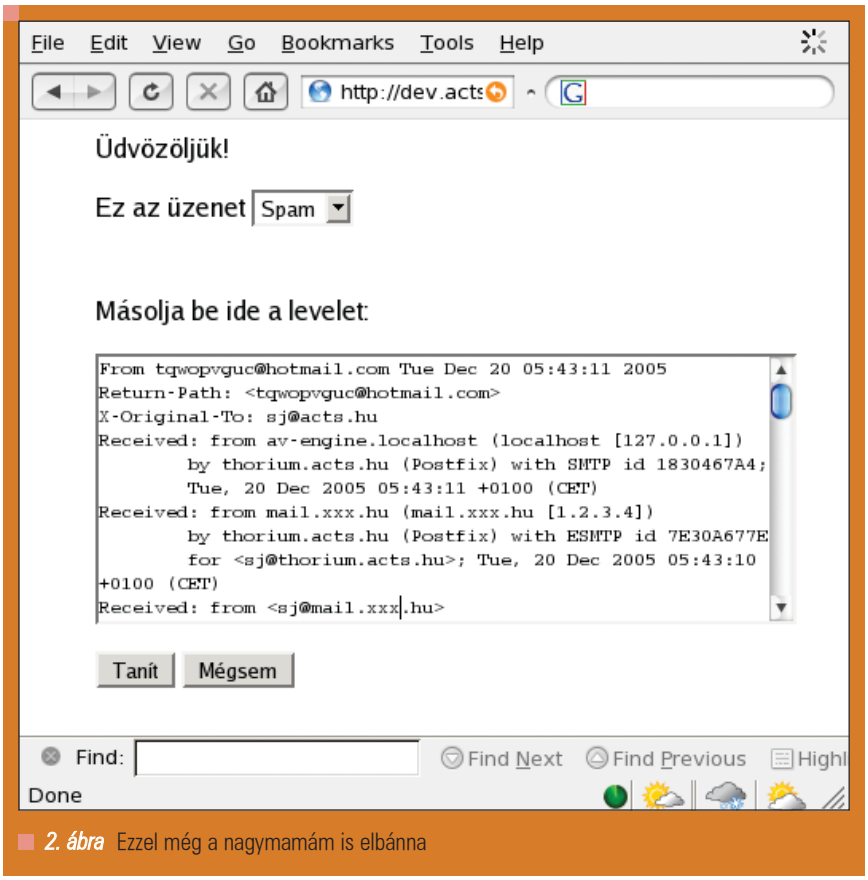
akkor érdekes, ha a felhasználó a postafiók mérete után fizet a szolgáltatójának. De mégis rossz ötletnek tartom ezt a megoldást, mert ha vesszük például a ***CRM114***-et, amelyről 99,87% pontosságot említenek (ez durván 1 téves azonosítás 1000 levelenként), felmerülhet a kérdés, hogy mi van akkor, ha pont abban a levélben volt egy nagy összegű megrendelés? Higgye el az olvasó, ***Murphy*** törvénye szerint előbb-utóbb lesz ilyen. Hacsak az előbb említett termék nem a gyártó folyamatosan karbantartott adatbázisából dolgozik. Ebben az esetben azonban lényegesen kisebb a rugalmassága, hiszen egy uniformizált spam adatbázis sohasem veheti fel a versenyt a mi leveleinkből képzett adatbázis pontosságával. A ***clapf*** ezért (vírusfertőzés esetét kivéve) soha nem utasít el levelet, csak megjelöli, és a felhasználóra bízza, mit tesz vele. Az átlagos spam mérete jellemzően néhány kB – bár a csatolt képet tartalmazó spam-ek 40-100 kB körüli méretűek is lehetnek. Ezért a `max_message_size_to_filter` paraméter segítségével megadható egy méret, amely fölötti leveleket spamszűrés nélkül átengedi.

## Feladat megoldva – vagy mégsem?

A figyelmes olvasónak bizonyára feltűnt az a diplomatikus megfogalmazás, hogy a ***clapf*** csak „nagy valószínűséggel" tudja eldönteni, hogy az adott levél spam vagy sem. Miért nem tudja biztosan? Bármely ***Bayesian*** szűrő minőségét két dolog határozza meg:

a) Milyen ügyesen bontja tokenekre az üzenetet,
b) a spam adatbázis minősége.

A spammer-ek sokszor ártatlannak látszó szavakat tesznek az üzenetbe (***poisoning***), amelyek valószínűleg nem szerepelnek a spam adatbázisban – sőt, ha egy mód van rá, akkor olyan szavakat, amelyek előfordul(hat)nak a felhasználó ham adatbázisában, mert ekkor sokkal kisebb lesz a levél összesített spam valószínűsége. A cikk írásakor kaptam egy olyan spam-et, amely az üzenet mellett tartalmazott egy részletet a ***„Tarzan New Yorkban"*** című műből. Ezért ma már a ***Bayesian*** spamszűrők csak a semleges középértéktől (0,5) leginkább eltérő 15-20 tokent vizsgálják meg, és ebből számítják ki az egész levél spam valószínűségét.

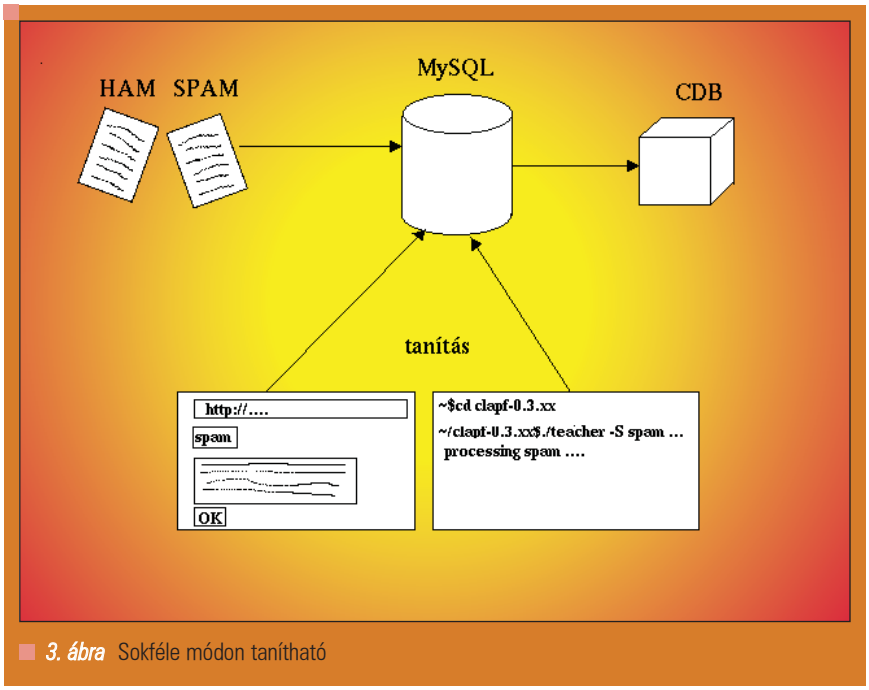



*2. ábra* Ezzel még a nagymamám is elbánna

Szóval ott jártam, hogy elkészült a spam adatbázis, minden rendben működött, de időnként mégiscsak becsúszott néhány spam. Az ilyen leveleket elemezve kiderült, hogy a vizsgált 15 token között bár voltak spam-re jellemző tokenek, de olyanok is, amelyek „ártatlannak" tűntek, és csökkentették az összesített valószínűséget, így a clapf által számított eredmény körülbelül 0,56-0,74 között volt, ami még nem spam, inkább bizonytalanságot jelez. A levelet tovább vizsgálva kiderült, hogy a top15 tokenen kívül volt még bőven spam-re jellemző token a levélben. Ezért módosítottam a ***clapf***-ot, hogy a semleges középértéktől nagyon eltérő összes tokent vegye be a számításba (`use_all_the_most_interesting_tokens=1`). A küszöbértéket nyilván nagyon magasra kellett állítanom, hogy ne lehessen ezzel megzavarni a számítást. Azonban időnként kaptam olyan spam-et is, amelyben az így kibővített top15 tokenek között óriási volt a spam tokenek aránya. Ezért egy új funkciót készítettem: ha itt a spam / összes token aránya nagyobb 0,9-nél (`spam_ratio_in_top10`), akkor a levelet spam-ként jelöli meg.

A ***DSPAM*** nyomán a ***clapf*** már nem csak önálló szavakat, de azok kombinációit (2 ill. 3 egymást követő tokenből képezi) is figyeli, mert ez növeli a ***Bayesian*** döntés pontosságát. A precízebb döntés érdekében a ***clapf*** már súlyozza a tokeneket és azok kombinációit aszerint, hogy hány elemből állnak. Minél specifikusabb, annál nagyobb súlyt kap, azaz egy 3 elemből álló kombináció többet nyom a latban, mint egy sima token. ***W. Yerazunis*** szerint a ***Bayesian*** elvvel van egy nagy baj: csak akkor érvényes, ha egymástól statisztikailag független eseményekre alkalmazzuk, azonban ez nem igaz levelek esetében. De még ezzel az alapvető logikai hibával együtt is rendkívül jól működik. A ***Bayesian*** szűrők pontosságának 99,9% az elméleti határa (➲ ***http://crm114.sourceforge.net/Plateau_Paper.html***), ami 1 tévesztést jelent 1000 levelenként, és ez azért nem rossz. De szerinte ez az arány tovább javítható a Markov diszkrimináció alkalmazásával. A legutolsó fejlesztői verzióban tesztelem ezt a változatot, ez ugyanis precízebb súlyozást tesz lehetővé.

Külön érdekesség a keleti spam, amely az én terminálomon sok esetben értelmezhetetlen, és sajnos nem tudok ezen a nyelven, hogy elmondhassam nekik, hogy ha ilyen karakterkészletet használnak, akkor még csak esélyük sincs, hogy tőlük vegyek ***Viagrát*** (vagy mit akarnak rám tukmálni). Ezt úgy küszöböltem ki, hogy készítettem egy `invalid_junk_characters[]` nevű tömböt, ahol több kínai, koreai jelet soroltam fel, amelyek tipikusan előfordulnak az ilyen és csak az ilyen levelekben. A ***clapf*** ezeket a karaktereket számolja, és egy határ után nem büntetőpontot ad, hanem az egész levelet spam-nek tekinti. Erre a problémára egyébként az az igazán frappáns megoldás, ha figyeljük az adott levél karakterkészletét, és ha az garantáltan értelmezhetetlen (például ***GB2312***, ***iso-2022-jp***, ***big5***, ***Windows-1251***), akkor azt eleve egy külön mappába gyűjtjük, amit én egy egyszerű ***maildrop*** szabállyal oldottam meg. Eddig csak spammer-ektől láttam, hogy – a levél elemzésével működő szűrőket megtévesztendő – akkor is ***base64***-kódolva küldik el az üzenetet a levél törzsében, ha annak típusa (text/plain) egyébként ezt nem tenné szükségessé. A ***clapf*** ezt képes dekódolni, ahogyan az ***UTF-8*** és ***Quoted-printable*** kódolt részeket is. Valószínűnek tartom, hogy a base64 kódolt, sima szöveges leveleket a jövőben automatikusan spam-nek fogom tekinteni. Az egyik felhasználó részéről felmerült egy olyan igény, hogy érdemes lenne vírusos levél vagy spam esetén értesíteni a feladót vagy a helyi postamestert. Azonban ezt nem tartom jó ötletnek, mert jelenleg a spam-ek tekintélyes részének hamisított a fejléce, másrészt a spam-ek tényleges küldői több esetben megfertőzött, esetleg feltört gépek, akiknek hiába is menne vissza egy panaszkodó levél, arról nem is beszélve, hogy kevés értelmét látom például 200 ***Zafi-B*** vírus esetén 200 vírusról értesítő levélnek ugyanazzal a szöveggel. Kérését mindenesetre teljesítettem, és a konfigurációs fájlban ez a funkció is aktiválható (alapértelmezésben kikapcsolt).




*3. ábra* Sokféle módon tanítható

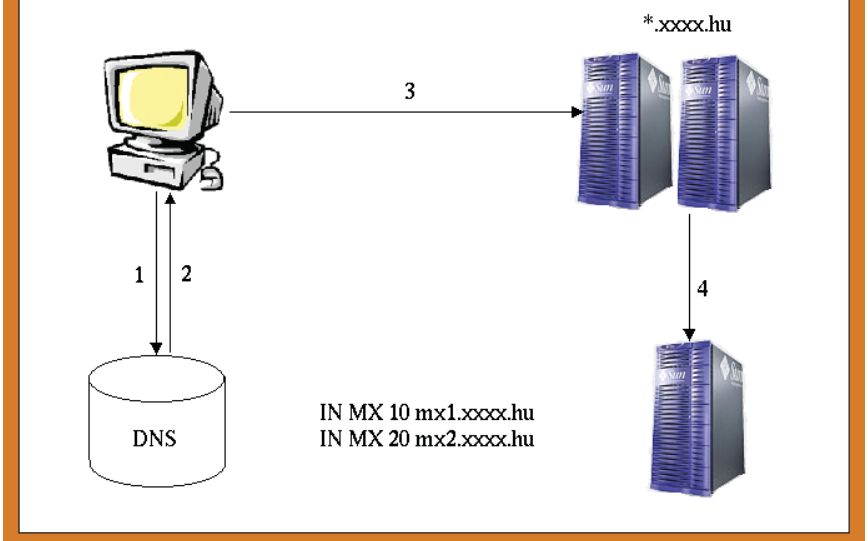
## Hogyan tovább?

Egy időben néhány heurisztikus tesztet is implementáltam (például `Cc:` mezőben szereplő címek száma, van-e benne IP-címmel megadott url, html tag-ek aránya, stb.). Sok spam-et megfogtam velük, azonban ezek könnyelmű beállítása megnövelte a téves pozitív riasztások esélyét. Ezért a 0.3.21-rc1-es verziótól kezdve a clapf már nem tartalmaz heurisztikát.

Aztán szembetalálkoztam a „jövő szemetével" (***Graham*** nevezte így), amely csak egy – viszonylag hosszú - URL-t tartalmazott (a ***http://uk.geocities.com/***... alatt valahol), ahol a spam tényleges üzenete olvasható. Mindezt pár véletlenszerű szóval fűszerezték meg, hátha sikerül még jobban összezavarni a ***Bayesian*** döntést. Egyelőre azonban nem akarom azzal komplikálni a ***clapf***-ot, hogy letöltöm az adott weboldalt, és annak is elemzem a tartalmát. Egy másik komplikált eset az, amikor a levél csak egyetlen képet tartalmaz, amit megnyitva olvasható a konkrét spam. Ez ellen úgy lehet védekezni, hogy egy képfeldolgozó alkalmazás segítségével kinyerjük a szöveget, és azt a megszokott módon elemezzük. Azonban a cikk írásakor nem találtam erre a célra megfelelő grafikus könyvtárat.

A ***clapf CDB*** formátumú token adatbázisból dolgozik. Azért esett erre a választásom, mert ezt találtam a legegyszerűbb megoldásnak, továbbá nem igényel külső relációs adatbázist – például ***Pgsql***, ***MySQL***) – amit külön felügyelni kell, csak olvasás módban nyitja meg a token adatbázist, így az nem hibásodik meg, nem kell zárolással foglalkozni, illetve rendkívül gyors, csak 2 diszk művelet, és megvan a keresett kulcs értéke. A jelenlegi körülbelül 7000 levélből (ham+spam) képezett token adatbázis mérete □29 MB, és körülbelül 602k rekord szerepel benne. A legutolsó fejlesztői változatban már 3 részre bontottam a token adatbázist, hogy minél gyorsabban le tudjam kérdezni a valószínűség értékeket. Hogy még kezelhetőbb legyen, a clapf ún. redukált adatbázist használ, amely nem tartalmazza azokat a tokeneket, amelyek csak egyszer szerepelnek vagy a ham vagy a spam tokenek között.

Egy hátránya azonban van a ***CDB*** formátumnak: nem frissíthető, márpedig téves azonosítás esetén tanítani kell. Ha módosítani akarjuk az adatbázist, akkor újra le kell generálni, ami akár pár percig is eltarthat. A ***doc/TRAINING*** fájlban azt fejtegetem, hogy az (lenne) a legegyszerűbb megoldás, ha a felhasználók elküldhetik a tévesen azonosított leveleket 2 belső email címre – az egyikre a tévesen spam-ként azonosított leveleket, a másikra a szűrőn átcsúszott spam-et, amelyből aztán rendszeres időközönként frissíteni lehet a központi adatbázist. Azonban ahány levelezőprogramot megnéztem eddig, annyiféle módon továbbítják a leveleket: az egyik csatolt fájlként, a másik beágyazott (inline) szövegként. Jelenleg azt a verziót preferálom, hogy




*4. ábra* clapf elosztott környezetben

a felhasználók egy spártai web oldalon egy űrlap segítségével taníthatják a *MySQL* adatbázisban szereplő tokeneket. Ebből pedig egy időzített feladat segítségével rövid idő alatt elkészíthető a *CDB* adatbázis.
Tervezem, hogy készíteni fogok egy programot, amelyet megfelelően felparaméterezve az adminisztrátor parancssorból is taníthatja a *MySQL* adatbázisban levő tokeneket. Ez megoldja a *Maildir* formátumú levelek importálását is. Lehet, hogy a jövőben közvetlenül *MySQL* adatbázisból fog dolgozni a *clapf*.
Az előbb vázolt megoldásban minden felhasználó egy közös adatbázison osztozik. Jobb megoldás azonban az, ha minden postafiókhoz tartozik egy – a felhasználó által karbantartható – token adatbázis, és a *clapf* mindig a megfelelőt használja. Ez azonban intenzívebb felhasználói közreműködést igényel, amire nem mindenki képes vagy hajlandó, ill. ehhez megfelelő diszk kapacitás is szükséges. Az egyes email címekhez (pontosabban felhasználói fiókokhoz) tartozó adatbázisok összerendelését tarthatjuk adatbázisban vagy akár *LDAP* kiszolgálón is.
Egy későbbi verzióban valamelyik megoldást implementálni fogom.
Aki pedig nem szerepel a címtárban, az használhatja az alapértelmezett központi adatbázist.
Egy web oldalon az adminisztrátor különféle statisztikákat fog majd tudni lekérdezni, például a spam/ham arányát, a feldolgozás átlagos idejét, különféle színes grafikonokat, stb. De ez a funkció a cikk írásakor még nincs kidolgozva.
Bár eddig folyton csak rosszakat mondtam a feketelistákról, bizonyos környezetben mégis jól használhatóak. Készítsünk egy csapda email címet, amit aztán mindenhol „reklámozunk" – csak ne a legitim partnereinknek. Ha erre a címre levél jön, az csak spam lehet, és a fejlécét elemezve, ki lehet nyerni belőle a feladó IP-címét, és azt egy helyi feketelistára tenni néhány órára. A *clapf* képes a bejövő levelek fejlécében szereplő `Received: from` sorok elemzésére, és ha az azokban talált IP-címek valamelyikét megtalálja az adatbázisban, akkor a levelet spam-ként értékeli (a *clapf* nem utasít el egyetlen levelet sem). Ezt a technikát nevezik oltásnak (*inoculation*).
Az is megoldható, hogy ne az *SMTP* szerverünkkel közvetlen kapcsolatba került feladót büntessük így, hanem keressük meg azt az IP-címet, ahonnan elindult a levél. Azonban a fejlécben szereplő útválasztási információ hamisítható, így fennáll a veszélye, hogy a feketelistánkra olyan címek kerülnek, amelyeket pedig nem akarnánk oda tenni. Ezért ezt változatot egyelőre nem támogatom, legalábbis amíg nem készítek egy olyan kódot, ami a fejléc információk hamisítását képes detektálni. Hamisított fejléc esetén viszont egészen biztosak lehetünk benne, hogy a feladó szándékai nem tiszták, így bátran megjelölhetjük az ilyen leveleket.

Egy *cron* héjprogram is szükséges az oltás teljesen automatizált működéséhez, amely rendszeres időközönként törli a lejárt bejegyzéseket. Az oltás az `--enable-blackhole` opcióval aktiválható. Ebben az esetben ajánlott a *clapf* konfigurációs fájl jogosultságát 640-re állítani, nehogy illetéktelenek hozzáférjenek egy adatbázis fiókhoz.
Előző írásaimban több konkrét spamszűrőn mutattam be a leggyakoribb védekezési elveket. Előfordulhat azonban, hogy egy vállalatnál nincs erőforrás erre a feladatra. Ma már azonban vannak olyan szolgáltatók, akik vállalják, hogy egy cég teljes bejövő levelezését átvizsgálják vírus és spam tekintetében. Ezt általában úgy teszik, hogy az adott tartomány *MX* rekordjait az ő gépeikre állítják be. Így a megbízó cég összes levele a szolgáltató szerverfarmjára megy. Itt átvizsgálják a leveleket, tartalmaz-e vírust vagy spam-et, majd ennek eredményét beillesztik a levélbe, és továbbítják a cég levelező szerverei felé. A *clapf* egy ilyen környezetben is képes helyt állni.

## A számok

Egy 12 napos időszak alatt 445 levelet kaptam, megfogott 119 spam-et, de egy azonban átcsúszott a szűrőn, így a pontossága 99.77% volt. A kategorizálás az én átlagos (<10kB) méretű leveleimnél 20-30 ms körüli időt vesz igénybe. A pontosságot illetően el kell mondanom, hogy valójában több levelet kaptam, például különféle listákról, de azokkal nem tanítottam az adatbázist, így azokat nem is vettem figyelembe a pontosság számításakor.
Angol levelek esetén időnként előfordult téves pozitív azonosítás, ami a korábban említett heurisztikus változók finomabb hangolásával (talán) elkerülhető lett volna. A heurisztikus elemzés eltávolítása óta azonban nem találkoztam ilyen hibával. Mire az olvasó idáig ért, valószínűleg már további finomításokat, funkciókat építettem a szűrőbe.

*Sütő János*
(jsuto@freemail.hu)
1997 óta használ Slackware Linux-ot. Szabadidejében a postfix clapf nevű vírus- és spamszűrőjét polírozza.