Razor

Spammerek borotvaélen

Bár az előző cikkemben (Linuxvilág 2005. november) bemutatott postgrey nevű szürkelista minden bizonnyal drasztikusan lecsökkentette az olvasók postafiókjába kerülő spamet, mégsem nyújt tökéletes védelmet.

A spammerek egy része ugyanis az *RFC* szabványok szerint működő levelező alkalmazással küldi el leveleit, így a spam nem az elküldés után azonnal, hanem 5 vagy 10 perc múlva, de megérkezik a postaládánkba. Ha az olvasó is „zéró toleranciával" viszonyul a spamhez, hadd mutassak egy újabb, a szürkelistát jól kiegészítő megoldást.
Ebben az írásban egy olyan kollaboratív spamszűrőt mutatok be, amely egy közösség erejével, kollektív tudásával harcol a spammerek ellen. A két legismertebb ilyen alkalmazás a *DCC* és a *Razor*. Cikkemben ez utóbbit ismertetem. Azért esett a választásom a *razor*-ra, mert viszonylag egyszerűen használható, és ha egyszer már bekonfiguráltuk, nem igényel további felhasználói karbantartást.

Telepítés

Töltsük le a *razor-agents* legutolsó változatát – ez a cikk írásakor 2.77 – az alkalmazás honlapjáról (➲ *http://razor.sourceforge.net/*).
A *razor Perl* nyelven írt alkalmazás, amely különféle *Perl* modulokat használ. Ezeket szerencsére nem kell egyenként letöltenünk, a *razor* készítői a *razor-agents-sdk* csomagban mindet összeszedték, így elég csak azt letölteni, szintén a *razor* honlapjáról. Azonban a modern *Linux* disztribúciók az *sdk*-ban található modulok többségét eleve tartalmazzák, nekem *Slackware Linux* alatt elég volt a *Digest-SHA1* és *URI* modulokat telepíteni a *CPAN*-ról (➲ *ftp://ftp.cpan.org/*), a többit tartalmazza a kedvenc disztribúcióm.

Csomagoljuk ki a *razor*-t:

```
tar jxvf razor-agents-
➥2.77.tar.bz2; cd
➥razor-agents-2.77
```

Készítsük el a *Makefile*-t:

```
perl Makefile.PL
```

Fordítsuk le, majd telepítsük:

```
make; su -c 'make install'
```

Beállítás

Első lépésként létre kell hoznunk a *razor* konfigurációs állományait, adjuk ki az alábbi parancsot:

```
razor-admin -create
```

Ez a home könyvtárunkban létrehozza a *.razor* könyvtárat, amelyben alapértelmezett paramétereket helyez el. Ezután regisztráljunk magunknak egy fiókot, vagy ha már létező nevet adtunk meg – ebben az esetben hibaüzenetet kapunk róla – akkor válasszunk másikat:

```
razor-admin -register -user
➥bela3 -pass anagyonnehezjelszo
```

Ha minden rendben ment (ebben az esetben a `razor-admin` 0 visszatérési értéket adott vissza, amit a `$?` héjváltozó tartalmaz), akkor létrejött a

```
~/.razor/identity-bela3
```

fájl és az erre mutató szimbolikus link:

```
~/.razor/identity
```

Próbáljuk ki!

Nézzük meg, mit tud a *razor*. Ehhez vegyünk egy levelet (teljes fejléccel), majd futtassuk az alábbi parancsot,

amely egy számot ad vissza. Ha az 0, akkor a levél spam, ha 1, akkor nem spam.

```
cat level | razor-check; echo $?
```

Mi történik a háttérben? A `razor-check` elemzi a levelet, és különféle ellenőrző összegeket (úgynevezett ***fuzzy checksums***) képez belőle. Ezután – amint a következő kódrészletből látható – gépünk felveszi a kapcsolatot a ***66.151.150.22*** IP című géppel (ez igazából a ***shock.cloudmark.com*** vagyis az egyik ***razor*** szerver), amely egy bannerrel válaszol a mi kapcsolatunkra, amire a mi kliensünk a program nevével és verziójával válaszol, ezután pedig elküldi a már említett összegeket. Ha a levél spam, azaz már szerepel a szerver adatbázisában, akkor a válaszban a `p=1` sztring szerepel, ha nem, akkor a `p=0`. (*1. lista*) Szeretném megnyugtatni a leveleik titkát féltő olvasókat, hogy a `razor-check` a levél tartalmát nem, csak az ellenőrzőösszegeket küldi el a ***razor*** szerverekre.

1. Lista

```
#ngrep -X port 2703
interface: eth0 (10.1.1.0/255.255.255.0)
filter: ip and ( port 2703 )
####
T 66.151.150.22:2703 -> 10.1.1.2:56089 [AP]
  sn=C&srl=5084&a=l&a=cg&ep4=7542-10..
##
T 10.1.1.2:56089 -> 66.151.150.22:2703 [AP]
  cn=razor-agents&cv=2.77..
##
T 10.1.1.2:56089 -> 66.151.150.22:2703 [AP]
  -a=c&e=4&ep4=7542-
  ➥10&s=o3dTCeHTppRgGT9B4M5wsdOOUoIA..a=c&e=4&ep4=
  7542-10&s=C6s6JzaIBPpnZQfOn4lTZqbpnUYA
  ➥..a=c&e=8&s=ySNqMCzAfDOA...
  ..
##
T 66.151.150.22:2703 -> 10.1.1.2:56089 [AP]
  -p=0..p=1&cf=100..p=1&cf=100.....
#
T 10.1.1.2:56089 -> 66.151.150.22:2703 [AP]
  a=q..
####
```

Élesítsük a borotvát

A ***razor*** leginkább a felhasználó oldali használatra felel meg, legegyszerűbben ***procmail***-lel, ***maildrop***-pal együtt lehet használni. Nézzünk egy példát ez utóbbira. Készítsük el a ***maildrop*** konfigurációs állományát (alapesetben ~/***.mailfilter***) az alábbi módon:

```
'/usr/bin/razor-check'

if($RETURNCODE == 0)
{
        to "mail/junk"
}

to "Mailbox"
```

Ennek hatására a ***maildrop*** minden egyes beérkező levelet átad a `razor-check` programnak, és annak visszatérési értékétől függően az adott levél vagy a kukába (***junk folder***) kerül, vagy a többi levél közé. Itt szeretném megemlíteni, hogy a `razor-check` és `razor-report` programok nem csak önálló leveleket, hanem ***mailbox*** formátumú folder-ben tárolt leveleket is fel tudnak dolgozni.

A ***razor*** előnye, hogy a tévesen spamnek ítélt levelek száma elhanyagolható, azaz, amire a ***razor*** azt mondja, hogy az spam, az – szinte – bizonyosan az. De a ***razor*** sem tévedhetetlen, ha a levél nem szerepel a ***razor*** szerveren, akkor azt nem ismeri fel spamként, még akkor sem, ha a levél valóban spam. Ez fordítva is igaz, ha a mi levelünk ellenőrző összege ütközik egy a már szerveren lévő másik levél összegével (***collision***), akkor a ***razor*** a mi levelünket ebben az esetben akkor is spamként értékeli, ha az valójában nem is az.

Ha a `razor-check` egy levelet tévesen ham-ként azonosított (***false negative***), akkor azt a `razor-report` programmal el tudjuk küldeni a ***razor*** szerverekre. Ebben az esetben azonban a kliensprogram – a hálózaton átmenő forgalom elemzése alapján – elküldi a teljes levelet. Úgy tapasztaltam, hogy néhány órának el kell telnie, mire a (fel)jelentésünk az éles adatbázisba kerül.

Teszt

Ideje megnézni, hogy milyen hatékonysággal, illetve pontossággal dolgozik a `razor-check`, amit 3 ***mailbox*** formátumú fájlon teszteltem. A vizsgálatban egy angol ill. egy magyar nyelvű levelező lista, továbbá a junk folderem vett részt. A két listán elvileg csak ham, míg a junk folderemben csak spam szerepelt – jellemzően

1. táblázat *Egy mérés eredménye*

	Levelek száma	Mailbox mérete [MB]	Feldolgozás ideje [sec]	Pontosság [%]
junk folder	319	1,9	24	73,04
angol nyelvű lista	309	2,5	18	98,7
magyar nyelvű lista	1121	15,7	75	98,82

angol és valami számomra értelmezhetetlen keleti nyelven. A mérés eredménye az *1. táblázatban* látható.
Futási idő viszonylatában – az arányokat tekintve – nincs különbség a két csoport, illetve 3 fájl között, átlag 15 levelet tudott kiértékelni a `razor-check` másodpercenként. Erről még annyit szeretnék megjegyezni, hogy ***qemu***-ban futtatott ***Slackware Linux*** alatt végeztem az említett teszteket. A számunkra érdekes különbség a pontosságban van. Míg a tiszta spamet tartalmazó junk folder levelei közül 86-t nem azonosított spamként, és így 73%-os pontosságot ért el, addig a levelező listákon meglepetésemre 2-4 levelet (körülbelül 1,3%) spamnek tekintett, tévesen. A magyar nyelvű listán elmarasztalt két levélben az a közös, hogy a levél törzse igen rövid (1 sor, ill. 1 betű), elképzelhető, hogy az erre kapott ellenőrző összegek hasonlítanak valaki valamelyik spam levelének lenyomatára. Hangsúlyozni szeretném, hogy ezek a számok az én leveleimre vonatkoznak, az Olvasó lehet, hogy ennél jobb vagy esetleg gyengébb eredményt ér el a ***razor*** használatával.

További gondolatok

Noha a ***razor*** leginkább kliensoldalon használható, de könnyen megoldható, hogy egy szerver oldali spamszűrő is használhassa. Ebben az esetben a `razor-admin` készítette konfigurációs fájlokat át kell helyezni a szűrőt futtató felhasználó home könyvtárába, majd azt kell elérni, hogy a spamszűrő alkalmazás például egy `exec()` rendszerhívással lefuttassa a `razor-check` programot, és annak visszatérési értékétől függően vagy automatikusan a spam folderbe mozgatja a levelet, vagy megjelöli valahogyan, ami alapján a kliensoldalon dönteni lehet róla, esetleg pozitív esetben extra büntetőpontot kap a levél.
Kezdetben az ezen az elven működő szűrők egyszerűen képezték a levél például ***MD5*** ellenőrző összegét. Azonban a spammerek elkezdték „testre szabni" a leveleket, azaz tettek bele például egy web bug-ot, ami egyedi azonosítót tartalmazott, így pusztán ettől már más lett ugyanannak a levélnek az ***MD5*** összege. Vagy tettek a levél végére néhány hottentotta „kifejezést", amellyel szintén az előbbi hatást érték el. Ezért ma már a levél ellenőrző összegét felhasználó szűrők akár több összeget is képeznek (mint például a most tárgyalt ***razor***), vagy igyekeznek kihagyni a számolásból a fenti módon perszonalizált részeket.
Szintén a szűrőket segíti néhány kevésbé ismert eljárás. A Rabin algoritmus hasonlóan működik, mint a jól ismert ***MD4***, ***MD5*** vagy ***SHA1***, ez is ujjlenyomatot képez a bementi adatból. Ebben eddig semmi új, de amíg például az ***MD5*** úgy van kitalálva, hogy az akár 1 biten is eltérő adatokhoz egymástól gyökeresen eltérő összegek tartozzanak, addig a Rabin algoritmus ezzel ellentétben két egymástól kevéssé eltérő adathoz hasonló összeget képez. Ha a spammerek minden levelet testre szabnak – manapság ezt automatikusan tudják a spamware alkalmazások – azaz a levelek gyakorlatilag viszonylag kis részletben térnek el egymástól, semmi gond, ezután is képezzük a szokásos ellenőrző összeget, és a szerver azt nézi meg, mekkora a „távolság" az ehhez leginkább hasonlító ujjlenyomattól. Ha ez egy határ alatt van, azaz a két szignatúra nagyon hasonlít egymáshoz, akkor nyilvánvalóan ugyanannak a spamnek két változatáról van szó. Ez az eljárás még másra is használható. Bár az írás témája a spamszűrés, de az elv a vírusok, elektronikus levélben terjedő férgek ellen is hatásos. Tegyük fel, hogy egy feketekalapos illető elkészíti a XXI. sz. szuperférgét, amely majd e-mailben fog terjedni. A vezető vírusírtó cégek nyilván csak egy bizonyos reakcióidő elteltével – ez a Commtouch szerint ➲ ***http://www.commtouch.com/site/OEM/zero_hour.asp*** akár 20-30 óra is lehet – tudnak hozzá vírusszignatúrát – vírusadatbázis frissítést – kiadni. Azonban az előbb tárgyalt módszer segítségével már akkor is védelmet élvezhetünk a „szuperféreg" ellen, amikor még nincs is rá „védőoltás", már a féreg elszabadulását követő pár perctől kezdve. De ehhez az is szükséges, hogy minél többen használják ezt a technológiát. A ***Commtouch*** 35 millió postafiókot említ a honlapján, amelyek szolgáltatói az ő technológiájukat licencelték.

Záró gondolatok

A ***razor*** hatékonyan használja ki az internet felhasználóinak kollektív erejét. Az Olvasónak is javaslom, hogy éljen ezzel a lehetőséggel, ha a levelezőszerverén használhatja a `procmail` vagy a `maildrop` programot. A ***razor*** által megjelölt leveleket bátran tehetjük a junk folder-be a `maildrop` segítségével. Érdemes továbbá a ***razor*** által fel nem ismert spamet egy külön spool fájlba gyűjteni, és azt rendszeres időközönként a ***razor-report*** programmal betölteni a ***razor*** adatbázisába.

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