Proxy megoldások

Szabad szoftver keretrendszer

Készítette a Közigazgatási és Igazságügyi minisztérium E-közigazgatási
Szabad Szoftver Kompetencia Központja
Budapest, 2013

Kódszám: EKOP–1.2.15

A dokumentum legfrissebb változata letölthető a honlapunkról:
http://szabadszoftver.kormany.hu/szabad-szoftver-keretrendszer/

Tartalomjegyzék

Proxy megoldások

Proxynak[1] nevezzük azokat a szoftvereket, amelyek a kliensek kéréseit köztes elemként más szerverekhez továbbítják. A kliens alkalmazás csatlakozik a proxy szerverhez, akár transzparenciával (anélkül, hogy tudná hogy egy proxy szolgáltatásait veszi igénybe - átlátszó), akár a kliens szoftverében rögzített módon, majd onnan egy erőforrást, vagy szolgáltatást igényel, amely egy vagy több másik szerveren található. Lényegében a proxy egy erőforrást vagy szolgáltatást oszt el. A proxy természetesen a rajta keresztüláramló adatokat és kapcsolatokat szabadon, a beállításai szerint módosíthatja. Azaz elemzés után a lokális tárolójából szolgálhatja ki a klienst anélkül, hogy a szerverhez fordult volna, vagy a kérésben és/vagy a válaszban megváltoztatott volna bármit. Sokszor a proxy célja a gyorsabb és erőforrás-kímélő felhasználás, de lehet korlátozás, megfigyelés alapeszköze is. A proxy legszélesebb körben a HTTP kiszolgálásra alkalmas eszközként terjedt el az 1990-es és a 2000-es évek elején, főként a modemes internetes időkben. Manapság azonban a legtöbb magasabb szintű tűzfal alapvető része több fajta proxy (elemző megoldás), illetve használjuk tunnelezésre, vagy erőforrások elosztására is. *Másik elterjedt felhasználása, hogy az illegális szervereket rejtik el proxy szerverek mögé, főként a torrent világban. Így biztosítva azt, hogy ha az egyik vagy másik proxy kiszolgálót a hatóság le is kapcsolja, akkor nem a központi (core) modulhoz (adatbázis stb.) jut el.* Ugyanilyen felhasználási módja a TOR[2] project megoldása, ahol szintén proxy szervereken keresztül juthatunk el az internetre anonim módon (igaz, a működési elvéből adódóan a mi internet hozzáférésünkön keresztül mások is forgalmazhatnak tartalmat szintén anonim módon). A proxyk tehát széles körben használatosak a szerveroldali felhasználás számtalan területén, és sok esetben egy tűzfal mögött, vagy egyéni alkalmazásokat használva nincs is közvetlen tudomásunk róluk.

A HTTP proxy

Többgépes hálózat esetén szinte kikerülhetetlen tényező volt az 1990-2000-es évek derekán, amikor is a legtöbb cég vagy betárcsázós analóg vagy ISDN vonalon osztozott, de később a bérelt vonalak feltűnésével is maradt a legtöbb kisvállalkozás életében a web proxy. Tipikusan jellemző viselkedés minta, hogy az egy területen dolgozó kollégák szabadidejükben ugyanazon oldalakat látogatják. A megfelelően beállított proxy csak a változásokat figyelve, mindig csak aktualizálva a saját lokális gyorsítótárában tárolt tartalmat szolgálja ki a belső hálózatot. Így előfordulhat az, hogy bár 20 kolléga nézi meg ugyanazon híroldal nyitólapját, amelynek mérete felhasználói letöltésenként kb. 5 MB lenne, azonban mégsem 20x5 MB a bejövő forgalom, mivel a proxy lokálisan szol-

[1] http://en.wikipedia.org/wiki/Proxy_server#Forward_proxies
[2] https://www.torproject.org/

gál ki 19 felhasználót, és csak az első felhasználó tölti le teljesen a tartalmat, ezzel is kímélve a sávszélességet. Az internetszolgáltatók régebben hatalmas proxy szervereket tartottak fenn, ugyanezen okokból, csak ott a belső hálózatot a betárcsázós modemes és ISDN/bérelt vonali felhasználók alkották, amíg a külső, spórolni kívánt vonal a nagyon drága nemzetközi sávszélesség volt, amely a régebbi időkben műholdas és mikrohullámú rendszerek segítségével jutott el többnyire hazánkba. Egy ilyen proxy rengeteg pénzt tudott megspórolni, és a felhasználói élményt is nagyban tudta javítani, hiszen a gyakran látogatott tartalmak szinte gyorsabban jöttek be mint a közelebbi, kevésbé látogatott itthoniak. Kezdetben számos kereskedelmi web proxy termék uralta a piacot, olyan cégek termékei mint a Netscape vagy a Cisco. A szabad szoftverek között a legelterjedtebb mind a mai napig a Squid proxy[3].

Squid

Alapjai az 1996-ban lezárult Harvest projekt keretein belül készültek el. Természetesen az elmúlt több mint 1,5 évtized alatt a kódja teljesen átalakult. A szoftver fejlesztésért felelős Duane Wessels az NLANR[4] munkatársa fogja össze a többi fejlesztőt. Alapja egyetlen blokkolásmentes I/O alapon vezérelt processz, ez végzi el a feladatok nagy részét. Sokszor ez is jelenti a legnagyobb problémát, pl. a memóriafoglalások tekintetében. Elterjedésének alapja, hogy funkcionalitása a kezdetek óta olyan bő, hogy a legtöbb kereskedelmi termék színvonalával nem csak felveszi a versenyt, hanem sok esetben meg is haladja azokat. Ennek köszönhető az is, hogy számtalan kereskedelmi dobozos termék, vagy eszköz felhasználja a Squidet.

A teljes körű szolgáltatási palettán olyan dolgok szerepelnek, mint: HTTP és FTP protokollok teljes körű kiszolgálása, az SSL kapcsolatok kezelése és gyorsítása, DNS cache, transzparens működési lehetőség. Részletesebb szakmai ismeret nélkül konfigurálható a tartalomszabályozás, és olyan népszerű gyorsítótárazási protokollok[5] széles körét támogatja, mint az ICP, HTCP, CARP és WCCP. Egyszóval egy igazi szabad szoftveres sikertörténet, főként a tudása és a rendkívül széles körű skálázhatósága miatt: ugyanúgy használható egy 10-20 fős iroda kiszolgálójaként, mint egy 100 000 ügyféllel rendelkező ISP[6] szervereként – természetesen más beállításokkal és merőben más hardverelemekkel. A Squidre ugyan úgy jellemző, mint a Postfixre és az SSH-ra is (hogy csak két példát ragadjunk ki), hogy a tényleges tudásának csak a felszínét kapargatjuk átlagos kis- és középvállalati üzemeltetés során.

A Squid telepítése

```
sudo apt-get install squid3
```

Ahogy az már megszokott, ez a parancs a függőségekkel együtt letölti a tárolókból, telepíti azt, majd alapbeállítás szerint létrehozza a cache dir környezetet és el is indítja a Squidet. Mivel az alapbeállítások nem feltétlenül felelnek meg nekünk, ezért a legjobb, ha telepítés után rögtön le is állítjuk:

```
sudo service squid3 stop
```

A konfigurációs állomány az /etc/squid3/squid.conf útvonalon található.

Nézzük tehát a squid.conf legfontosabb lehetőségeit:

```
http_port 3128
```

[3] http://www.squid-cache.org/
[4] http://www.nlanr.net/
[5] http://en.wikipedia.org/wiki/Web_Cache_Communication_Protocol
[6] http://en.wikipedia.org/wiki/Internet_service_provider

Definiálhatjuk, hogy a Squid melyik TCP porton fog hallgatni. A felhasználók ide fognak csatlakozni a klienseikkel (böngésző stb.). Jellemzően érdemes 3128-as alapbeállításon hagyni.

```
hierarchy_stoplist cgi-bin \?
acl QUERY urlpath_regex cgi-bin \?
cache deny QUERY
```

Jelentése: A dinamikus tartalmat nem gyorsítótárazzuk.

```
access_log /var/log/squid/access.log squid
```

A naplót itt fogjuk tárolni, ezért nagy terheltségű proxy esetében figyeljünk arra, hogy bár a logrotate fogja ezeket a naplókat is kezelni, de azzal együtt se nőhessenek túl nagyra.

```
cache_mem 256 MB
```

A Squid által használt memóriaterület nagyságát határozhatjuk meg. Érdemes itt[7] tájékozódni arról, hogy mennyit érdemes beállítani. Minden rendszer egyedi, és főképpen a felhasználók számától és viselkedésüktől függ. Az úgynevezett „hot object” mérettől is és a maximálisan tárolható objektum méretétől is függ, amely a következő opcióval állítható. Azaz képzeljünk el egy olyan hálózatot, ahol vegyesen Ubuntu kliensek és Windows kliensek használják a proxy szervert. Jellemzően az Ubuntu frissítések 0,5-5 MB méretűek, esetenként egy kernel 30-50 MB méretű is lehet. Azaz legalább ennek érdemes belefférnie a memóriába, hiszen ahogy a kliensek felkapcsolódnak a hálózatra, majdnem egyszerre fognak frissíteni. Windows frissítéseknél ugyan így kell ügyelni a méretezése, azaz:

```
maximum_object_size 128 MB
```

Ide azt a legnagyobb objektumméretet írjuk be, amely jellemző a környezetünkre.

Fontos az is, hogy a maximum_object_size beleférjen a cache_mem területbe is. A Squid a cache_mem terület sokszorosát is lefoglalhatja terhelés alatt. Jellemzően egy gyorsítótár-szerverbe legalább 4 GB memóriát érdemes beépíteni már 10-15 felhasználó esetében is, vagy ha van rá lehetőség, akkor ennél akár többet is, valamint ezzel párhuzamosan lehet növelni a cache_mem méretét is.

```
cache_dir ufs[8] /var/spool/squid3 1024 16 256
```

Ezzel a beállítással megadjuk, hogy hol helyezze el az átmeneti tárolókat, valamint sorban a gyorsítótár mérete MB-ban, a könyvtárak és az alkönyvtárak száma. Már egy alap proxy esetében is fontos, hogy a napló és a lemezgyorsítótár jól elkülönüljön, lehetőleg más partíción, még inkább más lemezen legyenek tárolva, hiszen ha ugyanabban az időben fogjuk a naplót írni és a tárolt adatot írni/olvasni, akkor a diszk olvasófejének mozgatása két távoli rész között fölöslegesen lassítana. Nagyobb rendszerek esetében érdemes egy külön vezérlőn RAID0-ba összefűzni egy nagyobb partíciót, hiszen itt az adatvesztés maximum átmeneti fennakadást okozhat csak, amíg újra megtelnek a tárolók. Érdemes továbbá a „noatime” mount-opciót használni a külön lemezgyorsítótárat tartalmazó fájlrendszer csatolásához, ezzel is nagyban gyorsítva a hozzáférést. További jó megoldás hagyományos merevlemezek esetében, ha minél gyorsabb (ma: 15k-val pörgő) lemezeket választunk és a fájlrendszer foglaltságát 50% közelébe kalibráljuk be. Ugyanakkor a folyamatosan egyre megfizethetőbb SSD-tárolók térnyerésével érdemes elgondolkodni egy sima SSD cache_dir megoldáson, hiszen ezt közel memória sebességgel fogja a rendszer elérni. Természetesen egy SSD Squid cacheként való használata erősen csökkenti az SSD élettartamát, hiszen folyamatos írásnak/olvasásnak van kitéve, így az egész rendszert nem érdemes SSD-re telepíteni, hanem kifejezetten csak a lemezgyorsítótár részt. Valamint az SSD állapotát folyamatosan monitorozni kell, például bekötve Nagi-

[7] http://wiki.squid-cache.org/SquidFaq/SquidMemory
[8] http://www.squid-cache.org/Doc/config/cache_dir/

os/Icinga alá, vagy smartmoontols-szal[9] figyelni. A fájlrendszer elkészítése során (mkfs) egy normál nem túl nagy cég kiszolgálásra alkalmas proxy beállítása esetén használhatjuk a default paramétereket, azonban ha nagy terheltségű gyorsítót akarunk üzemeltetni akkor érdemes utánaolvasni a blokkméretnek és úgy hozzuk létre az FS-t, hiszen jellemzően a Squid állományai pár KB-os méretűek.

```
acl[10] manager proto cache_object
acl localhost src 127.0.0.1/32 ::1
acl to_localhost dst 127.0.0.0/8 0.0.0.0/32 ::1
acl SSL_ports port 443          # https
acl Safe_ports port 80        # http
acl Safe_ports port 21        # ftp
acl CONNECT method CONNECT
http_access allow manager localhost
http_access deny manager
http_access deny !Safe_ports
http_access deny CONNECT !SSL_ports
http_access allow localhost
http_access deny all
```

A fenti beállításokkal egyben határozzuk meg egyrészt a belső hálózatunkat (jelen példában csak a localhost van engedve), illetve határozzuk meg azokat a portokat, amelyeket biztonságosnak ítélünk. Majd az első szabályok segítségével mindent engedünk a localhost-ról, minden Safe portot engedünk és végül minden mást kizárunk. Itt sorolhatjuk fel az alhálózatainkat, úgy mint:

```
acl localnet src 192.168.2.0/255.255.255.0
```

De ebben az esetben ne feledjük a http_access allow localnet segítségével azt engedni:

```
http_access allow localnet
```

Roppant fontos, hogy a belső proxy, ha van külső lába semmiképpen se látszódjon, vagy engedjen forgalmat kívülről, ugyanis előbb vagy utóbb a szemfüles, direkt ilyen hiányosságokat, azaz kilátszó anonymous proxykat kereső emberek vagy robotok ezt meg fogják találni és onnantól gyakorlatilag a nevünkben indíthatnak kéréseket a nagyvilág felé.

A Squid3 összekapcsolható LDAP-ból való azonosítással vagy Active Directory-val[11] is, így a már meglévő felhasználó-azonosítási rendszerbe is integrálható.

Transzparens üzemmódban

Ha nem szeretnénk, hogy a felhasználó maga dönthessen, hogy használja-e a proxyt, vagy sem, akkor kikényszeríthetjük annak használatát a helyi tűzfal és egy proxy szerver segítségével. A már beállított paramétereken kívül a Squid3-ban igazán csak a következő opcióra van szükség ahhoz, hogy átlátszó elemként beépüljön a hálózatunkba:

```
http_port 3128 transparent
```

Természetesen azért a tűzfalat módosítani kell, hogy a HTTP és az FTP forgalom átmenjen a proxy gépünkön. Egy aktuális LTS Ubuntuhoz készített leírást találhatunk itt[12], de álljon itt egy egyszerű példa amelyet a tűzfal gépünkön kell beállítanunk, hogy a forgalom átmenjen a proxy gépen:

[9] http://sourceforge.net/apps/trac/smartmontools/wiki
[10] http://wiki.squid-cache.org/SquidFaq/SquidAcl
[11] http://wiki.squid-cache.org/ConfigExamples/Authenticate/WindowsActiveDirectory
[12] http://ubuntuserverguide.com/2012/06/how-to-setup-squid3-as-transparent-proxy-on-ubuntu-server-12-04.html

```
iptables -t nat -A PREROUTING -i eth1 -p tcp -m tcp --dport 80 -j DNAT --to-destination 192.168.1.1:3128
iptables -t nat -A PREROUTING -i eth0 -p tcp -m tcp --dport 80 -j REDIRECT --to-ports 3128
```

Jelen esetben a Proxy gép IP címe a 192.168.1.1 és az eth0 a WAN felöli oldal.

A transzparens proxy beállításával lehetőség nyílik arra, hogy a teljes, titkosítatlan közegben történő felhasználói HTTP forgalmat monitorozzuk, majd elemezzük a Sarg[13] segítségével. Ez sok esetben célravezetőbb, mint tiltani és arra sarkalni a felhasználót, hogy keressen kiutat. Természetesen figyelembe kell venni a hazai törvényi környezetet, azaz a felhasználóval megfelelő módon közölni kell, hogy megfigyelés alatt áll.

Proxy beállítások központosítása:

A böngészők támogatnak egy úgynevezett automatikus proxy beállítást (Proxy AutoConfiguration - PAC). Ezt abban az esetben érdemes használni, ha nem tudunk vagy nem akarunk transzparens proxyt beállítani, de nem szeretnénk a klienseken az esetleges proxy IP-cím, port stb. változásokat kézzel állítgatni. Az ilyen esetekben egyszer kell csak a kliens böngészőben az automatikus proxy beállítást kiválasztani, majd megadni a proxy beállító szkript URL-jét. A hálózaton elhelyezett (HTTP- vagy HTTPS-protokollon elérhető) PAC[14] állományból olvassa majd fel a böngésző, hogy hol található a proxy szerver. Valamint a PAC[15] szkriptben azt is ki lehet kötni, hogy amennyiben a proxy szerver nem elérhető, akkor direktben menjenek ki az internetre a gépek.

Weboldalak blokkolása

Gyakori igény, hogy az ismertebb „sávszélességrabló” oldalakat szükség esetén blokkoljuk, erre a Squid többféle megoldást is kínál. A legegyszerűbb egy acl létrehozása és kitiltása:

```
acl block_site dstdomain "/etc/squid3/blocked-sites"
http_access deny block_site
```

Értelemszerűen a /etc/squid3/blocked-sites állomány tartalmazza a blokkolni kívánt oldalak listáját.

Ennél kifinomultabb megoldás kínál a **squidguard**[16], amely beépülő modulként hívható a Squidból, természetesen telepítés után:

```
apt-get install squidguard
```

Tudása messze túlmutat az egyszerű blokkolási lehetőségeken[17], hiszen adatbázisból dolgozik, így eleve mások által készített feketelistákat is felhasználhatunk[18]. A Squid-del való integrációja igen egyszerű, a Squid konfigurációját a következő sorral kell kiegészíteni:

```
redirect_program /usr/sbin/squidGuard -c /etc/squid3/squidGuard.conf
```

Majd a Squid újraindítása után használatba vehetjük:

```
sudo service squid restart
```

[13] http://sarg.sourceforge.net/
[14] http://wiki.squid-cache.org/SquidFaq/ConfiguringBrowsers
[15] http://en.wikipedia.org/wiki/Proxy_auto-config
[16] http://www.squidguard.org/
[17] http://www.squidguard.org/about.html
[18] http://www.squidguard.org/about.html

Az /etc/squid3/squidGuard.conf fájlban beállíthatunk mások által definiált adatbázisokat, kategóriákat is létrehozhatunk:

```
        dbhome /usr/local/squidGuard/db
logdir /usr/local/squidGuard/log

dest gambling {
    log         gambling
    domainlist  gambling/domains
    urllist     gambling/urls
}

dest warez {
    log         warez
    domainlist  warez/domains
    urllist     warez/urls
}

acl {
        default {
                pass !gambling !warez all
                redirect 302:http://gnu.hu
        }
}
```

Így ha a példában szereplő gambling/domains, gambling/urls, és warez/domains és warez/urls fájlok megfellelően ki vannak töltve tartomány- vagy számítógépnevekkel illetve URL-ekkel, akkor pl. a szerencsejátékos és a warezoldalak egy jelentős részét tudjuk blokkolni, vagy átirányítani akár egy meglévő külső akár egy saját hálózatban lévő belső weboldalra, amely természetesen lehet egy figyelmeztetés is.

Sarg[19]

Ez az igen hasznos alkalmazás kiegészíti a Squid naplózási képességeit. Természetesen a proxy szerver naplóállománya igen részletes, így nagy felhasználói szám mellett még a gyakorlott naplóelemző számára is tökéletesen átláthatatlan. A Sarg ebben nyújt segítséget, mivel grafikusan elemzi számunkra a felhasználók szokásait. Használata és telepítése egyszerű, felépítése és konfigurációs állománya nagyon hasonlít a Webalizerhez[20]. Praktikusan éjszaka lefut, és az aznapi tevékenységet grafikonok és táblázatok segítségével elhelyezi egy kiértékelés könyvtárban, amelyet aztán akár weblapszerűen (webszerver szükséges) vagy lokálisan (a tartalmat saját gépre átmásolva, pl időzített scp, stb) egy böngészőből kiértékelhetünk különböző rendezési elvek alapján. Nagyon könnyű összegezni vele a legnagyobb letöltőt, illetve mérni vele az esetleges nem tiltott, de nem ajánlott tartalmakon eltöltött időt. Éppen ezért komoly visszaélésekre is lehet használni, hiszen a felhasználók teljes napi/heti/havi weben töltött naplóját könnyen értelmezhetően mutatja. Tehát kezeljünk szenzitív adatként, és mindenképpen védjük/tároljuk megfelelően a kiértékelt adatokat.

[19] http://sarg.sourceforge.net/
[20] http://en.wikipedia.org/wiki/Webalizer