Redundáns tűzfal konfiguráció OpenBSD/PF alapon

Csillag Tamás
cstamas kukac digitus.itk.ppke.hu
Pásztor Miklós
pasztor kukac ppke.hu
Pázmány Péter Katolikus Egyetem, Információs Technológiai Kar

HTML változat: http://www.ppke.hu/~pasztor/openbsd-pf.html

Tartalom

Az OpenBSD a szabad operációs rendszerek közt kit?nik biztonságával és a hálózati szolgáltatásainak fejlettségével. OpenBSD-n fejlesztik az OpenSSH-t, a legelterjedtebb SSH implementációt. Az OpenBSD programjai, dokumentációi egyszerű, áttekinthető, lényegretörő egységet alkotnak.

A PF az OpenBSD csomagszűrő szoftvere, melyet 2001-ben fejlesztett ki Daniel Hartmeier. Azóta a PF elérhető más operációs rendszereken is. A PF hatékony, egyszerűen konfigurálható és módot ad redundáns tűzfal kialakítására is. E tulajdonság folytán ezen kritikus szolgáltatás nem esik ki egy upgrade, hardver csere, vagy áramkimaradás miatt.

A PPKE-n több mint egy éve használunk OpenBSD/PF alapú tűzfalat. Az előadás ismerteti a kialakított konfigurációt, és a felgyülemlett tapasztalatokat.

Az eszközök

OpenBSD

Az OpenBSD csak egyike a manapság elérhető szabad unix operációs rendszereknek. Az OpenBSD projektet 1995-ben Theo de Raadt alapította Kanadában, mely a legnemesebb unixos hagyományokat folytatja: szikárság, egyszerűség, lényegretörő, mondhatjuk, hogy nyers hozzáállás jellemzi. Nem csak a klasszikus KISS azaz ,,Keep it small and simple" jelszót tarják, jelszavuk: ,,Shut up and hack" . Ez történik például az évenként sorrakerülő összejöveteleken a ,,hackaton"-okon. Egységes, áttekinthető, nem szószátyár és nem izgő-mozgó semmilyen dokumentáció, installációs procedúra, web lap, vagy akár

a levelezési lista, viszont kezdettől fogva a legnagyobb figyelmet fordítják a *biztonságra* és *megbízhatóságra.* Rendszeresen, évente kétszer adnak ki új változatot, 2006 májusi a 3.9. A projekt pénzt elsősorban a CD-k eladásából szerez. Az OpenBSD különösen hálózati alapszoftverekben nyújt sokat. Itt van az OpenSSH, az OpenNTPD „hazája" és kiváló routing protokoll implementációkat fejlesztenek OpenBSD alatt: OSPF, BGP.

PF (Packet Filter)

A PF az OpenBSD csomagszűrő szoftvere. Legfőképpen egy svájci programozó Daniel Hartmaier munkája. Funkciójában a népszerű iptables megfelelője. 2004-ig még a PPKE ITK-n is Linux-ot és iptables-t használtunk, arról tértünk át PF-re. Akárcsak iptables-ben, itt is definiálhatunk:

- szűrési szabályokat különböző szempontok szerint
  - forrás és/vagy cél cím
  - interfész
  - protokoll (pl. UDP, TCP)
  - forrás és/vagy cél port
- a szűrt csomagokat visszautasíthatjuk, vagy eldobhatjuk
- cserélhetünk IP címet (NAT)
- átirányíthatunk forgalmat
  - más IP címre, portra
  - helyi alkalmazáshoz
- naplózhatunk ...

A pf vezérlésének két útja:

- `/etc/pf.conf` konfigurációs fájl
- `pfctl` parancs

PF szépségek

Listák

Egy-egy szabályban felsorolásokat tehetünk például portokra, vagy IP címekre ilyesféleképpen:

```
pass out proto udp from any to 10.20.30.40 port {domain, ntp}
```

Makrók

Változókat definiálhatunk, és ez után például interfészek, vagy IP címek nevükkel adhatók meg.

```
dmz_if = "em1"
dns_server = "10.20.30.40"
#
# ...
#
pass out log $dmz_if proto udp from any to $dns_server port {domain, ntp}
```

Stateful filtering

A PF nem csak csomagokat, hanem *kapcsolatokat* lát. Ez azt jelenti, hogy például ha egy TCP kapcsolat felépítését a szabályok engedélyezik, és rendelkeztünk róla, akkor a kapcsolat többi csomagját már külön rendelkezésünk nélkül is engedélyezi a PF. A kapcsolatról egy táblázatot tart a PF, ami nem csak azt teszi lehetővé, hogy különböző paramétereket ellenőrizzen (pl. kiszűrje a window-n kívüli csomagokat), hanem jelentősen gyorsítja a további csomagok átengedését. A PF belső állapotait a `pfctl -ss`

paranccsal kérdezhetjük le.

Táblázatok

IP címek (IPv4, vagy IPv6) egy halmazát táblázatokba tehetjük. A táblázatokra aztán hivatkozhatunk a pf konfigurálásakor szűrési, átirányítási vagy NAT szabályoknál. A táblázatokat bővíthetjük vagy szűkíthetjük menet közben `pfctl` parancsokkal. Táblázatok használata nem csak a konfigurálást teszi könnyebbé, hanem a működést is: a táblázatokon végzett műveletek hatékonyabbak, mintha szabályismétlést vagy akár listákat használnánk. Példa:

```
table <goodguys> { 192.0.2.0/24, !192.0.2.5 }
pass  in on fxp0 from <goodguys> to any
```

Scrub

A scrub a csomagok normalizálására szolgáló eszköz. Segítségével megtehetjük többek közt, hogy:

- Az IP fragmentumokat egyben továbbítjuk
- A kimenő IP csomagok ID mezőjét randomizáljuk
- A kimenő IP csomagok TTL-jét normalizáljuk

Anchor-ok

Az anchor-ok segítségével a szabályok egy halmazát kezelhetjük külön. Egy anchor tartalmazhat szűrési, vagy átirányítási szabályokat, sőt, akár újabb anchor-okat. Egy anchor-ra hivatkozhatunk a konfigurációs fájlból, vagy `pfctl` paranccsal. Ilyen módon struktúráltan, és rugalmasan tudunk konfigurációt kialakítani, például eseményvezérelt módon (akár cron-ból), módosíthatjuk, bővíthetjük a szabályokat. Példa:

```
anchor horgony
load anchor horgony from "/etc/horgony.0"
```

Menet közben aztán ehhez hasonló shell parancsokkal kezelhetjük az anchor-t:

```
#pfctl -a horgony pass in quick on $dmz_if proto tcp from 11.12.3.4 to $ns_server
port domain keep state
```

Az anchor definiálásakor megadhatunk feltételeket is: például interfészek, IP címek és portok szerint. Ilyen módon az anchor-ok az iptables chain-ekhez hasonló funkciót tölthetnek be.

Common Address Redundancy Protocol (CARP)

A carp arra szolgál, hogy két vagy több számítógép közös IP címen nyújtson szolgáltatást abból a célból, hogy ha az egyik kiesne, akkor a másik gép zökkenőmentesen átvehesse a szerepét. Carp segítségével mód van terhelés megosztársa (load balancing) is. A Carp hasonlít a Cisco által használt VRRP protokollhoz, de több annál: Carp-ban például a protokoll üzenetei kriptográfiailag védettek, így nehezítve a dolgát egy imposztornak aki el akarná terelni a hálózatunkon a forgalmat hamis carp üzenetekkel.

Egy carp interfész definiálásakor meg kell adnunk, hogy melyik fizikai interfészen melyik az az IP cím, amin osztozik két vagy több számítógép. Ehhez az IP címhez azután közösen használnak egy addig nem létező ethernet címet, amivel hol egyik, hol másik gép válaszolja meg az ARP kéréseket.

A carp - konfigurálható időközönként - multicast csomagokat küld a 112-es IP protokoll kóddal. Ezek a csomagok határozzák meg, hogy a carp társak közül éppen ki is a master.

Carp interfészt ilyesfajta paranccsal definiálhatunk:

```
inet 11.12.13.14 255.255.255.0 NONE carpdev em0  vhid 5 advskew 10 pass nehez-szo
```

Ha a `hostname.carp0` fájlunk tartalma ez a sor, akkor az em0 ethernet kártyán jelentkezni fog a 11.12.13.14-es IP címen a gépünk. Carp interfészt definiálhatunk 802.1Q vlan interfészekre is.

pfsync

A pfsync segítségével a tűzfal kapcsolatainak változásait tudjuk szinkronizálni két csomagszűrő között. Célszerű a szinkronizálás céljára külön interfészt használni között amiket összeköthetünk egy keresztkábellel. A PF állapotváltozásait azután ezen keresztül szinkronizálják a tűzfalak.

A pfsync - akárcsak a carp - egy virtuális interfész. Ilyesfajta paranccsal definiálhatjuk:

```
up syncdev fxp0 syncpeer 10.1.2.3
```

Ha a `hostname.pfsync0` fájlunk tartalma ez a sor, akkor az fxp0 ethernet kártyán a 10.1.2.3 IP című géppel fogunk szinkronizálni.

A PPKE ITK-n használt konfiguráció

A karunkon használatos redundáns tűzfal konfigurációt az 1. ábra szemlélteti:

Amint látható több belső vlan-t és fizikai hálózatot, a külső interfészt és pfsync interfészt használunk. A redundáns működés lehetővé teszi, hogy az éppen működő tűzfalon operációs rendszert, cseréljünk, hardvert bővítsünk anélkül, hogy ezzel akár csak egy másodperc kiesést okoznánk. Előfordult, hogy egy belső gépre ssh-val bejeletkezve dolgozó munkatársunk észre sem vette, hogy közben a tűzfal áramellátása - egy ügyetlen mozdulat következtében - megszűnt: nem csak az IP címeket, hanem az élő ssh kapcsolatra vonatkozó állapotot is átvette a tartalék.

Problémák és megoldások

A PF használata során nem egy problémával szembesültünk, amikre elegáns egyszerűséggel adott megoldást az OpenBSD/PF páros.

Belső gépre nyitni SSH folyosót dinamikus ADSL címről

Probléma

Van a hálózatban egy belső gép, aminek semmilyen szolgáltatása nem érhető el a világból, azonban egy munkatárs otthoni gépéről mégis meg akarjuk engedni az ssh elérést. Nehézséget okoz, hogy az otthoni gép változó IP című (ADSL-lel kapcsolódik).

Megoldás

A problémát PF és PF táblázatok segítségével oldottuk meg. A változó IP címhez DNS nevet rendelünk. Erre szolgál az a.itk.ppke.hu aldomain. Az ADSL vonal végén levő gép bekapcsoláskor regisztrálja a nevét.

A tűzfalon definiálunk egy táblázatot:

```
table <ad_sl> {}
```

A táblázatban szereplő címekről engedélyezzük az ssh forgalmat:

```
pass out  quick proto tcp from <ad_sl> to  $in_here port ssh $syn keep state
```

Egy cron job azután a következő egyszerű `pfctl` paranccsal engedélyezi az ssh forgalmat:

```
pfctl -v -t ad_sl -Tr $HIP
```

Tantermi internet forgalom korlátozása

Probléma

A számítógépes tantermekben gyakran kívánatos, hogy csökkentsük a hallgatók kísértését, arra, hogy munka helyett az interneten kóboroljanak.

Megoldás

A PF egyik szép lehetősége az authpf shell. Ha például a `pista` felhasználó shell-je `/usr/sbin/authpf`, akkor amikor bejelentkezik - tipikusan ssh-val -, akkor az /etc/authpf/users/pista/authpf.rules fájlban levő pf szabályok betöltődnek az `authpf` anchor-ba, és addig maradnak érvényben, amíg a felhasználó be van jelentkezve. Ezt az eszközt jól lehet használni, ha például wifi hozzáférést csak username/jelszóval akarunk megengedni hálózatunkban, vagy ha a hálózat belsejéhez akarunk ideiglenes kapukat nyitni.

A tantermi internet böngészés letiltásához azonban éppen nem engedélyezésre, hanem tiltásra használjuk az authpf-et. Létrehozunk egy `nenet` nevű felhasználót a tűzfalon `authpf` shell-lel. Egy ehhez hasonló sort teszünk a hozzá tartozó authpf.rules fájlba:

```
block return in  log quick  proto tcp from $user_ip/24
```

Ennek hatására a tűzfal nem engedi át a felhasználó /24-es hálózatából a forgalmat. Mivel minden tanteremhez egy-egy /24-es belső hálózati címtartomány tartozik, ennek az az eredménye, hogy az egész tanteremből tiltva lesz a tűzfalon át a tcp forgalom. A módszer finomítva van azáltal, hogy a fenti sor elé beteszünk néhány elengedhetetlenül szükséges szolgáltatás engedélyezést.

Neptun túlterhelés

Probléma

Karunkon működnek az egyetemi hallgatói információs rendszer szerverei. Az ehhez való hozzáférés úgy történik, hogy windows remote desktop szerverekhez kapcsolódnak távolról a felhasználók, és ezeken a terminál szervereken indul el a Neptun kliens alkalmazás. Amikor vizsgaidőszak kezdődik, akkor az egyetem négy karáról egy időben több ezer felhasználó próbál kapcsolódni a rendszerhez. A windows terminál szervererre folytonosan jönnek tcp kérések a 3389-es portra. Ezt a terminál szerver elfogadja, de egy határon túl azonnal elbontja, mert korlátozva van az egyszerre felépíthető kapcsolatok száma. Ennek az az eredménye, hogy a TCP kapcsolat time_wait állapotban marad. Ilyenből több tízezer állapotbejegyzés keletkezik a tűzfalon. Sokkal okosabb lenne, ha a windows nem tcp fin-nel, hanem azonnal resettel bontaná a kapcsolatot, de a probléma PF segítségével is megoldható.

Megoldás

A PF többféle korlátozást is lehetővé tesz, melyek védenek a túlterhelés ellen, DoS támadások ellen. Az állapottáblázat méretét korlátozzuk egy ilyesfajta beállítással:

```
set limit states 20000
```

Az állapottáblázatban az egyes bejegyzések elévülésének idejét szabályozzuk így:

```
set timeout { adaptive.start 8000, adaptive.end 40000 }
```

Ezzel a beállítással kijelülünk egy intervallumot, és ha az állapotok száma ebbe az intervallumba esik, akkor egy lineáris skála szerint hamarabb évülnek el tűzfal állapotai, a következő szorzó szerint:

```
(adaptive.end - number of states) / (adaptive.end - adaptive.start)
```

A másik triviális bevezethető korlátozás nem egészében a tűzfalra, hanem csak egy-egy szabályra vonatkozik, például egy-egy szolgáltatásra:

```
pass out log quick on $neptun_if proto tcp to $neptun port $neptuns_server $syn keep
state (max 4000)
```

Ennek hatására legfeljebb 4000 kapcsolat épülhet fel, mely eszerint a szabály szerint létesülhetett.

FTP gond és megoldása

Probléma

Az egyetemi hálózatban levő kliens gépekről általában lehet kifele TCP kapcsolatot kezdeményezni - kivéve SMTP kapcsolatot -, befele azonban nem megengedett semmilyen kapcsolat. Az FTP protokoll két TCP csatornát használ: a vezérlő kapcsolat a kliens véletlen (ephemeral) portja és a szerver 21-es portja közt épül fel, FTP adatkapcsolat viszont a klasszikus esetben az ftp szerverről épül fel a kliens gépre (aktív ftp). Ez problémát jelent a csomagszűrő tűzfalon, amit linux alatt iptables-ben a connection-tracking old meg.

Megoldás

PF-ben a pftpx nevű eszközt használjuk. Ez nem része, csak kiegészítése a PF-nek, egy ftp proxy, amihez PF szinten átirányítjuk az FTP kéréseket:

```
rdr pass on $int_if proto tcp from $lan to any port 21 -> 127.0.0.1 port 8021
```

Bevezetünk megfelelő anchor-okat, amikbe a proxy majd az általa generált szabályokat helyezi:

```
anchor "pftpx/*"
nat-anchor "pftpx/*"
rdr-anchor "pftpx/*"
```

Ezek után az ftp kapcsolatok a külső szerverek szemével nézve a tűzfalról érkeznek, és nem koppannak a tűzfalon, mert ebbe a pftpx anchor-ba „röptében" beteszi a megfelelő szabályokat a pftpx processz.

Flow információk gyűjtése

Probléma

A netflow a Cisco által fejlesztett nyilt protokoll, ami adatgyűjtésre szolgál, hálózati kapcsolatokról készít összesítéseket. Cisco és Juniper routerekben szokásos használni: a router egy külső collector gépre küldi UDP csomagokban a gyűjtött adatokat, ahol azonnal, vagy később fel lehet dolgozni. Karunkon viszont a forgalom jelentős része nem megy át ilyen routeren: a redundáns tűzfal tölti be a router szerepet a tucatnyi belső VLAN között. Ahhoz, hogy az itt folyó forgalomról is képet nyerjünk, a PF által átengedett forgalomról is szükséges adatokat gyűjteni.

Megoldás

A PF-hez a pfflowd nevű kiegészítés képes szabványos netflow formában adatokat küldeni egy kollektor géphez. Damien Miller ausztrál programozó munkája. A pfflowd a pfsync mechanizmusát használja arra, hogy netflow alakúra alakítsa a már rendelkezésre álló flow-k információját.

Spam források fékezése

Probléma

A hálózati sávszélesség növelése a spam küldőknek is kedvez: egyre nagyobb sávszélességgel, egyre több helyről árasztanak el szeméttel.

Megoldás

A spamd nevű eszköz smtp szerverként funkcionál mint egy valódi levelező szerver, de nagyon laaasaaaan válaszol, sokáig feltartja a spam küldőt, és végül elutasítja a levelet.

A PF-ben egy táblázat segítségével a spam forrásokból jövő kapcsolatot a spamd-hez terelhetjük:

```
rdr pass on $ext_if inet proto tcp from <spamd> to any port smtp tag SPAMD ->
127.0.0.1 port 8825
```

Az OpenBSD-ben a `spamd-setup` program segítségével egyszerűen és rugalmasan konfigurálhatjuk, hogy mi is legyen a `spamd` táblázatban és egyúttal azt is megadhatjuk, hogy a spamd végül milyen üzenettel utasítsa el a levelet. A spamd-setup veheti a forrás IP címeket fájlból, vagy a hálózatról. Egyetemünkön egy cron-ból induló procedúra futtatja két óránként a spamd-setup-ot, módosítva a spamd PF táblázatot.

Spam, féreg elhanyagolt gépekről

Probléma

Sok vírus, féreg és más rosszindulatú kód települ felhasználók gépeire, és ezek gyakran automatikus levelekkel bombázzák azokat az e-mail címeket, amiket kibányásztak a fertőzött gépből. A windows95 és windows98 operációs rendszerekből érkező levelek nagy valószínűséggel ilyen levelek. Ezért nem akarunk ilyen gépekről érkező leveleket elfogadni.

Megoldás

A PF egyik szép tulajdonsága a ,,passive fingerprinting": nem csak IP címek, protokollok, vagy portok szerint tudunk szűrni, hanem operációs rendszer szerint is. Az operációs rendszerek felismerésére Michal Zalewski lengyel informatikus fejlesztett ki eszközt, ez a p0f. Az IP csomagok, pontosabban a TCP SYN csomagok sajátságait veszi figyelembe: igaz ugyanis, hogy nagy biztonsággal meg lehet állapítani, hogy milyen operációs rendszer, de még azt is, hogy milyen változata küldte. A p0f adatbázis OpenBSD-n a `pf.os` fájlban található. A http://lcamtuf.coredump.cx/p0f-help/ webhely arra szolgál, hogy bővítse, javítsa az adatbázist. Egy ilyen sor eredményesen szűri a windows9x operációs rendszerekből jövő leveleket:

```
rdr pass on $ext_if inet proto tcp from any os {"Windows 95", "Windows 98"} to any
port smtp tag SPAMD_WIN -> 127.0.0.1 port 8825
```

Amint látható az smtp kapcsolatokat a lokális gép 8825-ös portjára irányítjuk, ahol a spamd ,,fogadja" a leveleket.

Összefoglalás

Az OpenBSD/PF páros nem csak biztonságos, jól kezelhető, redundáns tűzfal felépítésére alkalmas, hanem már eddig is nagyon sok kellemes meglepetést hozott egyetemünkön. Biztos, hogy még mindig vannak kihasználatlan tartalékai. Ilyen például a QoS: PF-ben mód van prioritásos sorok kezelésére, és sávszélesség garantálására.

Irodalom

[1] http://www.benzedrine.cx/pf-paper.html
[2] http://www.openbsd.org/faq/pf/
[3] Jacek Artymiak: Building Firewalls with OpenBSD and PF
ISBN 83-916651-1-9
[4] http://www.bgnett.no/~peter/pf/en/index.html
[5] http://www.mindrot.org/pfflowd.html
[6] http://www.stearns.org/p0f/