Zastitni zid sa OpenBSD PF paket filterom

Peter N. M. Hansteen

Datadokumentasjon A/S

The document is a 'work in progress', based on a manuscript prepared for a lecture at the BLUG (see http://www.blug.linux.no/) meeting of January 27th, 2005.
i'm interested in comments of all kinds, and you may if you wish add web or other references to html or pdf versions of the manuscript. If you do, i would like, but can not require, you to send me an email message that you've done it. For communication regarding this document please use the address <peter@bgnett.no>.

Pregled sadrzaja

Pre nego sto pocnemo

Ovo predavanje[1] ce biti o zastitnim zidovima i srodnim finkcijama, pocevsi od *malo* teorije pa kroz primere filtriranja i raspodelom mreznog saobracaja. Kao i u drugim slucajevima, stvari o kojima cu

diskutovati mogu biti uradjene *na vise nacina*. Uvek i bilo kada, slobodno me prekinete ako vam nesto zatreba. To jest, ako mi dozvolite da koristim ono sto cu nauciti od vasih komentara kasnije, bilo u prepravljenoj verziji ovog predavanja ili u praksi. Ali prvo,

Ovo nije HOWTO

Ovaj document nije namenjen kao gotov recept za copy i paste.

Da bismo ovo zapamtili, molim vas ponavljajte zamnom

Zavet Mreznog Admina

Ovo je moja mreza.

moja je
ili tehnicki pripada mojim poslodavcima,
moja je duznost
i ja brinem za nju svim svojim srcem

Postoje i druge mreze koje veoma lice na moju,

Ali nijedna nije kao ona.

i kunem se

Da necu nepromisljeno kopirati iz HOWTO-a

Sustina je u tome da, iako pravila i konfiguracije koje cu pokazati rade, ta pravila mogu biti veoma jednostavna i veoma je verovatno da su barem malo zastarela i verovatno veoma pogresna za *vasu mrezu.* Molim vas, imajte na umu da je ovaj document namenjen da predstavi nekoliko korisnih stvari i inspirise vas da postignete dobre stvari.
Tezite ka razumevanju vase mreze i sta vam je potrebno da biste je napravili boljom.
Molim vas nemojte slepo kopirati iz ovog ili bilo kog drugog dokumenta.

Sada, kada smo to resili mozemo preci na sustinu stvari.

PF?

Prvo, nekoliko reci o softveru o kojem cemo diskutovati, OpenBSD PF.
PF je napisan tokom leta i jeseni 2001 od strane Daniela Hartmeiera i nekoliko OpenBSD developera, i predstavljena je kao osnovni deo OpenBSD 3.0 sistema u Decemberu 2001.
Potreba za novom vrstom zastitnog zida za OpenBSD proizasla je kada je Darren Reed objavio svetu da IPFilter, koji je do tada bio veoma blisko integrisan u OpenBSD, ustvari nije pod BSD licencom. U stvari potpuno suprotno. Sama licenca je skoro rec u rec kopija BSD licence, samo je izostavljeno pravo da se kod menja. OpenBSD verzija IPFilter-a je sadrzala prilican broj izmena, koje, kako se ispostavilo, prema licenci nisu bile dozvoljene. IPFilter je uklonjen iz OpenBSD source tree 29-tog Maja 2001 godine, i nekoliko Nedelja OpenBSD-current nije sadrzao nikakav zastitni zid.
Srecom, U Svajcarskoj Daniel Hartmeier je vec radio na ogranicenim eksperimentima ukljucujuci kernel hacking u mreznom kodu.

Njegova polazna tacka je bila povezivanje ubacivanje njegovih licnih funkcija u mrezni stack, dozvoljavajuci paketima da prolaze kroz njega, i nakog nekog vremena poceo je da razmislja o filtriranju. Onda je nastupila kriza oko licence.
IPFilter je izbacen iz source tree 29-tog Maja. Prvi kod za PF napisan je u Nedelju 24-tog Juna 2001 u 19:48:58 UTC.
Usledilo je nekoliko meseci veoma intenzivnog rada, i verzija PF-a koja je trebala da izadje sa OpenBSD 3.0 je sadrzala skoro kompletnu implementaciju filtriranja paketa, ukljucujuci mrezno prevodjenje adresa.
Kako je izgledalo, Daniel Hartmeier i ostali PF developeri su iskoristili svoje iskustvo sa kodom za IPFilter. Daniel je predstavio USENIX 2002 sa testovima ucinka koji su pokazali da se OpenBSD 3.1 PF pokazao jedanko dobro ili bolje pod opterecenjem nego IPFilter na istoj platformi ili iptables na Linuxu.
Dalje, neki testovi su sprovedeni na originalnom PF-u iz OpenBSD-a 3.0. Ovi testovi su pokazali da je kod dobio na efektivnosti od verzije 3.0 do verzije 3.1. Clanak koji sadrzi detalje je dostupan na Daniel Hartmeier web stranici, pogledajte *http://www.benzedrine.cx/pf-paper.html*.
Nisam video skorasnje uporedne testove, ali prema mom iskustvu, i prema iskustvu drugih, PF filtering zanemarljivo malo opterecuje resurse. Kao jedan podatak, kompjuter koji sluzi kao gateway izmedju Datadok mreze i ostatka sveta je Pentium III 450MHz sa 384MB RAM-a. Kad sam se setio da pogledam, nisam video da je masina ikada imala manje od 96 procenata idle procesa.

Paket filter? Zastitni zid?

Do sada sam vec upotrebio neke izraze i koncepte koje nisam objasnio i to cu ubrzo da ispravim. PF je *paket filter*, to znaci, kod koji kontrolise mrezne pakete na nivou paketa i portova i odlucuje sta ce sa njima. U slucaju PF-a ovaj kod vecinu vremena radi u kernel prostoru, unutar mreznog koda.
PF deluje u svetu koji se sastoji od paketa, protokola, konekcija i portova.
Na osnovu toga odakle paket dolazi ili gde ide, na osnovu kog protokola, konekcije porta je namenjen, PF je u mogucnosti da odluci gde da vodi paket, ili da odluci dali ce on uopste biti propusten.
Isto tako je moguce da se mrenzni saobracaj usmerava na osnovul *sadrzaja* paketa, koji se obicno tumaci kao filtriranje na aplikacionom nivou, ali ovo nije ono sto PF radi. Kasnije cemo se vratiti na neke slucajeve gde PF moze da preda ovakve zadatke drugom softveru, ali prvo da se upoznamo sa nekim osnovama.
Vec smo napomenuli koncept zastitnog zida. Jedna vazna odlika PF-a i slicnih softvera, mozda i najvaznija odlika, je da su sposobni da indentifikuju i blokiraju saobracaj koji ne zelite da propustite do vase lokalne mreze ili nezelite da ga propustite prema spoljnom svetu. U neku ruku termin ‘zastitni zid’ je logican.

NAT?

Jedna druga stvar o kojoj cemo dosta pricati su ‘unutrasnje’ i ‘spoljasnje’ adrese, ‘rutabilne’ i ‘ne-rutabilne’ adrese. Ovo u sustini se ne odnosi direktno na zastitni zid ili filtriranje paketa, ali zbog nacina na koji svet danas radi moracemo se osvrnuti na ovo. Ovo dolazi iz vremena ranih devedesetih kada je neko poceo da racuna broj kompjutera koji ce biti povezani na internet ako se nastavi sa komercijalizacijom i velike mase korisnika koji ce se povezivati u isto vreme.

Kada su internet protokoli formulisani, kompjuteri su obilno bili glomazne, skupe stvari na kojima obicno radi veliki broj simultanih korisnika, svaki na svom manje vise glupavom terminalu. Povezivanje je bilo dozvoljeno samo univerzitetima i nekolicini kompanija koje su imale ugovor sa Pentagonom. U sustini 32 bitne adrese od 4 okteta ce se odrzati veoma dugo. Bukvalno mogu da prime milione kompjutera.
A onda se desila komercijalizacija Interneta, i odjednom bilo je milon malih, jeftinih kompjutera koje cekaju da se konektuju u isto vreme. Ovaj tip razvoja je pokazivao sve znake nastavka cak i ubrzanja razvoja. Ovo je znacilo da pametni ljudi koji su nacinili mrezu treba da urade jos nekoliko novih stvari. Oni su uradili nekoliko stvari manje vise u isto vreme. Kao prvo, poceli su da rade na resenju zasnovanom na vecem adresnom prostoru – ovo je nazvano IP verzija 6, ili krace IPv6 - koja korsiti 128 bitne adrese. Ovo je naznaceno kao dugorocno resenje. Mislim da sam do sada napomenuo da je podrska za Ipv6 ugradjena u OpenBSD kao default, i PF je uvek, koliko ja znam podrzavao IPv6.
Dalje, neki tip privremenog resenja je bio potreban. Kada bi se svet prebacio na koriscenje adresa koje su cetiri puta vece za to bi trebalo dosta vremena. Proces je trenutno, kao sto mozemo da vidimo, jos uvek u veoma ranoj fazi. Naslo se privremeno resenje koje se sastoju iz dva dela. Jedan deo je mehanizam koji omogucuje da mrezne gateways prepisu adrese paketa, drugi deo se sastoji u tome da se dodele odredjeni rasponi adresa, koje nisu bile odredjene ranije za koriscenje, samo u mrezama koje nebi komunicirale direktno sa Internetom u velikom broju. Ovo bi znacilo da nekoliko razlicitih masina, na odvojenim lokacijama, mogu imati istu lokalni IP adresu. Ali ovo nebi bio problem zato sto bi adrese bile prevedene pre nego sto se saobracaj propusti van na mrezu u velikom broju. Ako bi saobracaj sa takvim ‘nerutabilnim’ adresama dolazio na Internet u velikom broju, ruteri koji vide saobracaj bi imali opravdan razlog da zabrane paketima da prolaze.
Ovo se naziva "Prevodjenje Mreznih Adresa", ponekad se naziva i “IP maskarada" ili slicno. Dva RFC-a koja definisu sve u vezi ovog su datirana 1994-te i 1996-te[2].
Postoji dosta razloga za koriscenje takozvanih "RFC 1918 adresa", ali tradicionalno i istorijski glavni razlog je da su oficijalne adrese ili nedostupne ili neprakticne.

PF danas

Obradili smo malo istorije. Od 2001 je proslo nekoliko godina, i PF u sadasnjoj OpenBSD 3.8 formi je paket filter koji je sposoban da radi dosta stvari, ako to zelite od njega.
Na primer, PF klasifikuje pakete na osnovu protokola, porta, tipa paketa, izvora ili odredisne adrese. Sa odredjenom dozom sigurnosti on je isto tako u mogucnosti da klasifikuje pakete na osnovu izvornog operativnog sistema.
i iako NAT nije neophodan deo paket filtera, iz prakticnih razloga, lepo je da se logika prepisivanje adresa obradjuje negde u blizini. Prema tome, PF sadrzi i NAT logiku.
PF moze –na osnovu kombinacije protokola, porta i ostalih podataka – da usmeri saobracaj ka ostalim odredistima u odnosu na ona odredjena od strane posiljaoca, na primer drugoj masini ili proramu za dalju obradu kao npr. daemonu koji slusa na portu, lokalno ili na drugoj masini.
Pre nego sto je PF napisan, OpenBSD je vec sadrzao altq kod za kontrolisanje opterecenja i stanja saobracaja. Nakon nekog vremena, altq je integrisan sa PF-om. Prvenstveno iz prakticnih razloga. Kao rezultat ovoga, sve ove funkcije su dostupne vama preko jednog, veoma jednostavnog, konfiguracionog fajla, koji se obicno naziva pf.conf, koji se cuva u /etc/ direktorijumu.
Ovo je sada dostupno kao deo osnovnog sistema na OpenBSD-u, u FreeBSD-u gde je PF, od verzije 5.3, jedan od tri zastitna zida koja se mogu koristiti po zelji, u NetBSD-u i DragonFlyBSD-u. Sa zadnja dva sistema nisam puno radio zbog sredstava. Nekada se nema potrebog hardvera i vremena u isto vreme. Ali svi su indikatori da su samo male ralike u detaljima izmedju ovih sistema, barem sto se tice PF-a.

BSD vs Linux - Konfiguracija

Pretpostavljam da danas postoje oni koji su vise upoznati sa konfigurisanjem Linux-a ili ostalih sistema nego BSD-a, tako da cu nakratko pomenuti nekoliko stvari o BSD konfiguraciji.
BSD mrezni interfejsi nisu obelezeni kao eth0 i tako dalje. Interfejsima su data imena koja se podudaraju sa imenom drajvera plus broj sekvence, tako da se 3Com kartice koje koriste xl drajver pojavljuju kao xl0, xl1, i tako dalje, dok ce Intel kratice verovatno imati em0, em1, SMC kartice sn0, i tako dalje.
U globalul, BSD sistemi su organizovani da citaju konfiguraciju iz /etc/rc.conf, koji je citan od /etc/rc skripte pri podizanju sistema. OpenBSD preporucuje koriscenje /etc/rc.conf.local za lokalno prilagodjavanje, zbog toga sto rc.conf sadrzi default vrednosti, dok FreeBSD koristi /etc/defaults/rc.conf da bi sacuvao default opodesavanja, cineci /etc/rc.conf pravim mestom za unosenje promena.
PF se kofigurise editovanjem /etc/pf.conf fajla i koriscenjem pfctl alata u komandnoj liniji. pfctl aplikacija sadrzi *veliki* proj opcija.Pogledacemo blize neke od njih.
U slucaju da ste se zapitali, postoje web interfejsi dostupni za admin zadatke, ali oni nisu delovi osnovnog sistema. PF razvojni tim ne gleda neprijateljski na ove alate, ali u stvari nisu videli da je bilo koji graficki interfejs za PF konfiguraciju bez sumnje bolji od pf.conf u tekst editoru, potpomognut prizivanjem pfctl i nekoliko ostalih unix trikova.

Najjednostavnije moguce podesavanje (OpenBSD)

Ovo nas, konacno, dovodi do prakticnog dela gde podesavamo PF u najjednostavnijoj konfiguraciji. Radicemo na jednoj masini koja komunicira sa mrezom koja moze biti i Internet.
Da bi startovali PF, kao sto smo predhodno napomenuli, morate da kazete rc-u da zelite da pokrenete servis. U OpenBSD-u, ovo se radi u /etc/rc.conf.local, sa magicnom linijom

```
pf=YES          # enable PF
```

veoma jednostavno. Dalje, mozete, ako to zelite, da odredite gde ce PF da nadje svoja pravila.

```
pf_rules=/etc/pf.conf # specify which file contains your rules
```

Pri sledecem pokretanju sistema PF ce biti pokrenut. Ovo mozete preoveriti ako potrazite poruku PF enabled u konzoli. Fajl /etc/pf.conf koji dolazi sa normalnom instalacijom OpenBSD-a, FreeBSD-a ili NetBSD-a sadrzi izvestan broj korisnih predloga, ali su oni svi iskljuceni stavljanjem # karaktera na pocetku linije.
Nemorate da restartujete svoj kompjuter da biste pokrenuli PF. Ovo mozete lako da uradite koristeci pfctl. Ne zelimo da restartujemo kompjuter bez nekog dobrog razloga, zato upisujemo komandu

```
peter@skapet:~$ sudo pfctl -e ; sudo pfctl -f /etc/pf.conf
```

koja startuje PF[3][4]. Za sada nemamo skup pravila, sto znaci da ustvari PF ne radi nista.

Verovatno je vazno napomenuti da ako restartujete vas kompjuter sada, rc skripta na OpenBSD-u ce ucitati standardni skup pravila, koji je ustvari ucitan *pre* nego sto su bilo koji od mreznih interfejsa pokrenuti.
Ovaj standardni skup pravila sluzi kao bezbednosna mera u slucaju da se vas gateway podigne sa nevalidnom konfiguracijom. On vam omogucava da se ulogujete i da prepravite ono sto je izazvalo syntax gresku zbog koje se vas skup pravila nije ucitao. Osnovni skup pravila omogucava osnovni skup servisa: ssh pristup sa bilo kog mesta, osnovni name resolution i NFS mounts.
Neke ranije verzije PF portova si izgleda nisu donosili osnovna pravila sa njima.

Najjednostavnije moguce podesavanje (FreeBSD)

Na FreeBSD-u vam treba malo vise magije u vasem /etc/rc.conf, posebno prema FreeBSD Handbook-u[5]

```
pf_enable="YES"                 # Enable PF (load module if required)
pf_rules="/etc/pf.conf"         # rules definition file for PF
pf_flags=""                     # additional flags for pfctl startup
pflog_enable="YES"              # start pflogd(8)
pflog_logfile="/var/log/pflog"  # where pflogd should store the logfile
pflog_flags=""                  # additional flags for pflogd startup
```

Na FreeBSD-u, PF je standardno kompajliran kao modul koji se ucitava u kernel. Ovo znaci da mozete poceti sa $ sudo kldload pf, pa onda $ sudo pfctl -e da bi pokrenuli PF. Ako ste ubacili ove linije u /etc/rc.conf, mozete koristiti PF rc skriptu da startujete PF:

$ sudo /etc/rc.d/pf start

Najjednostavnije moguce podesavanje (NetBSD)

Na NetBSD-u 2.0 i novijim PF je dostupan kao modul koji se moze ucitati u kernel, instaliran preko paketa kao pkgsrc/security/pflkm ili kompajliran u konfiguraciju statickog kernela. U NetBSD-u 3.0 i nadalje, PF je deo osnovnog sistema.
Ako zelite da omogucite PF u vasoj kernel konfiguraciji (radije nego da ucitate kernel modul), dodajte sledece linije vasoj konfiguraciji kernela:

```
pseudo-device  pf                # PF packet filter
pseudo-device  pflog                        # PF log interface
```

U /etc/rc.conf vam trebaju ove linije

```
pf=YES
pflogd=YES
```

da biste omogucili PF i PF log interfejs.

Ako ste instalirali modul, ucitajte ga sa NetBSD$ sudo modload /usr/lkm/pf.o, pa zatim NetBSD$ sudo pfctl -e da omogucite PF. Alternativno, mozete pokrenuti rc skripte, NetBSD$ sudo /etc/rc.d/pf start da omogucite PF i NetBSD$ sudo /etc/rc.d/pflogd start da omogucite logiranje.
Da ucitate modul automatski pri podizanju sistema, dodajte sledecu liniju u /etc/lkm.conf:

```
/usr/lkm/pf.o - - - - AFTERMOUNT
```

Ako ste do sada uradili sve kako treba, spremni ste da kreirate vas prvi skup pravila.

Prvi skup pravila – jedan kompjuter

Ovo je najjednostavnije podesavanje, za jedan kompjuter na kojem nece raditi nijedan servis, i koji ce biti povezan sa jednom mrezom koja moze biti Internet. Za sada, koristicemo /etc/pf.conf koji izgleda ovako:

```
block in all
pass out all keep state
```

to znaci, zabrani sav dolazni saobracaj, dozvoli saobracaj koji mi kreiramo, i odrzavaj informacije o stanju na nasim konekcijama. Odrzavanje informacija o stanju omogucava prolaz do nas povratnom saobracaju za sve konekcije koje smo mi zapoceli. Ovo radite u slucaju ako je ovo masina kojoj mozete da verujete. Ako ste spremni da koristite skup pravila ucitavate ih sa

```
$ sudo pfctl -e ; sudo pfctl -f /etc/pf.conf
```

Malo strozije

Za malo slozenije i kompletnije podesavanje, pocecemo tako sto cemo zabraniti sve i onda dozvoliti prolaz samo onim stvarima za koje znamo da nam trebaju[6]. Ovo nam daje priliku da se upoznamo sa dvema funkcijama koje cine PF tako izvanredim alatom - liste i makroi.
Napravicemo neke izmene u /etc/pf.conf, pocevsi sa

```
block all
```

Onda malo zastanemo. Makroi treba da se definisu pre upotrebe:

```
tcp_services = "{ ssh, smtp, domain, www, pop3, auth, pop3s }"
udp_services = "{ domain }"
```

Sada smo prikazali nekoliko stvari odjednom – kako makroi izgledaju, pokazali smo da makro moze biti lista, i da PF razume pravila koristeci imena portova jednako dobro kao sto to radi sa brojevima portova. Imena su ona koja su izlistana u /etc/services. Ovo nam daje mogucnosti koje mozemo iskoristiti u nasim pravilima, koje smo malo izmenili da bi izgledali ovako:

```
block all
pass out proto tcp to any port $tcp_services keep state
pass proto udp to any port $udp_services keep state
```

Sada ce neki od nas da istaknu UDP nemoze odrzavati konekcije, ali PF u stvari uspeva da odrzi informacije o stanju konekcije uprkos ovom. Kada pitate server imena za ime domena, verovatno zelite da dobijete odgovor.
Posto smo napravili izmene u pf.conf, ucitacemo nova pravila:

```
peter@skapet:~$ sudo pfctl -f /etc/pf.conf
```

i nova pravila ce se primeniti. Ako nema sintaksnih gresaka, pfctl nece izbaciti nikakvu poruku prilikom ucitavanja pravila. Ova oznaka –v ce dati opsirnije pfctl izlazne podatke.
Ako ste napravili dosta izmena u vasem skupu pravila, mozete da proverite vasa pravila pre nego sto ih ucitate. Komanda za to je, pfctl -nf /etc/pf.conf. Opcija -n omogucava da se pravila samo prevedu bez njihovog ucitavanja. Ovo vam omogucava da ispravite eventualne greske. Zadnji validni skup pravila koji je ucitan ce se primenjivati dok ne ugasite PF ili ucitate novi skup pravila.

Stastitisticki podaci iz pfctl

Mozda cete pozeleti da proverite dali ustvari PF radi, i mozda u isto vreme da pogledate neke statisticke podatke. pfctl nudi niz razlicitih tipova informacija ako koristite pfctl -s, dodavsi tip informacija koje zelite da prikazete. Sledeci primer je preuzet sa mog kucnog gateway-a dok sam pripremao ovo predavanje:

```
peter@skapet:~$ sudo pfctl -s info
Status: Enabled for 17 days 00:24:58         Debug: Urgent

Interface Stats for xl0              IPv4             IPv6
  Bytes In                     9257508558                0
  Bytes Out                     551145119              352
  Packets In
    Passed                        7004355                0
    Blocked                         18975                0
  Packets Out
    Passed                        5222502                3
    Blocked                            65                2

State Table                          Total         Rate
  current entries                       15
  searches                         19620603        13.3/s
  inserts                           173104         0.1/s
  removals                          173089         0.1/s
Counters
  match                             196723         0.1/s
  bad-offset                             0         0.0/s
  fragment                              22         0.0/s
  short                                  0         0.0/s
  normalize                              0         0.0/s
  memory                                 0         0.0/s
  bad-timestamp                          0         0.0/s
  congestion                             0         0.0/s
  ip-option                             28         0.0/s
  proto-cksum                          325         0.0/s
  state-mismatch                       983         0.0/s
  state-insert                           0         0.0/s
  state-limit                            0         0.0/s
```

```
src-limit                   26          0.0/s
synproxy                     0          0.0/s
```

Prva linija nam prikazuje da je PF startovan i da je aktivan nesto vise od dve nedelje, sto je jednako vremenu od kada sam upgrade-ovao do tada najnovijeg 3.9-beta snapshot. `pfctl -s all` daje meoma detaljne informacije. Probajte je i pogledajte, i dok ste tamo, pogledajte i neke druge pfctl opcije. `man 8 pfctl` vam daje potpune informacije.
Za sada imate jedan kompjuter koji moze da komunicira veoma dobro i u dovoljno sigurnom obliku sa ostalim kompjuterima povezanim sa Internetom.
Ipak nekoliko stvari jos uvek nedostaje. Na primer, verovatno zelite da propustite barem neki ICMP i UDP saobracaj, ako nista drugo barem zbog svojih potreba u otklanjanju gresaka.
Iako postoje modernije i sigurnije opcije, verovatno cete morati da se pozabavite sa ftp servisom. Ubrzo cemo se vratiti na ove stvari.

Jednostavan gateway sa NAT-om

Sada se napokon pomeramo ka ralisticnijim ili barem vise uobicajenim podesavanjima, gde kompjuter konfigurisan kao zastitni zid ima ulogu i gateway-a za barem jos jedan kompjuter. Ostali kompjuteri mogu naravno imati svoj zastitni zid, ali cak i ako imaju, to uopste nije vazno za ono sto nas ovde interesuje.

Gateway i zamke za in, out i on

U podesavanju za jedan kompjuter, zivot je relativno jednostavan. Saobracaj koji vi kreirate bice ili propusten ili ne ka spoljnom svetu, i vi odredjujete sta cete propustiti unutra. Kada podesavate gateway, vasa perspektiva se menja. Idete od podesavanja tipa “ja protiv ostatka sveta” do “Ja sam taj koji odlucuje sta da propustim za ili od svih mreza sa kojima sam povezan”. Kompjuter ima nekoliko, ili najmanje dve, mrezne kartice, svaku povezanu na posebnu mrezu.
Veoma je razumno razmisljati da ako zelite da vas mrezni saobracaj bude propusten od mreze povezane na `$int_if` prema hostu na mrezi povezanoj na `$ext_if`, onda ce vam trebati pravilo koje bi izgeldalo nesto kao

```
pass in on $int_if from $int_if:network to $ext_if:network \
    port $ports keep state
```

koje isto tako odrzava konekciju. Ipak, jedna od najcescih kao i greske na koje se najcesce zale u konfiguraciji zastitnog zida jeste neshvatanje da rec “to” ne garantuje prolazak do odredista. Pravilo koje smo upravo napisali samo propusta saobracaj do gateway-a na unutrasnjem interfejsu. Da bi se paketu dozvolilo da ode malo dalje treba vam pravilo koje kaze

```
pass out on $ext_if from $int_if:network to $ext_if:network \
    port $ports keep state
```

Ova pravila ce raditi, ali to ne mora da znaci da cete postici ono sto ste trazili.
Ako postoje dobri razlozi zasto vam trebaju ovako specificna pravila u vasem skupu pravila, onda znate da vam trebaju i zasto. Za osnovnu konfiguraciju gateway-a o kojoj cu ja pricati ovde, ono sto bi ste hteli da koristite je verovatno pravilo koje kaze

```
pass from $int_if:network to any port $ports keep state
```

da biste dozvolili vasoj lokalnoj mrezi pristup Internetu i ostavite detektivskii posao *antispoof* i *scrub* kodu. Oba su prilicno dobri i kasnije cemo se vratiti njima. Za sada samo prihvatimo cinjenicu da za jednostavna podesavanja, pravila vezana za interfejse sa in/out pravilima teze da dodaju vise galame neko sto je to potrebno za vas skup pravila.
Za zauzetog mreznog administratora, citak skup pravila znaci bezbedan skup pravila.
U ostatku ovog tutorijala, sa nekim izuzecima, trudicemo se da pravila budu sto je moguce jednostavnija zbog citljivosti.

Podesavanje

Pretpostavimo da smo kompjuteru dodali jos jednu mreznu karticu ili da ste podesili mreznu konekciju sa vasom lokalnom mrezom preko PPP ili na drugi nacin. Necemo razmatrati specificnu konfiguraciju interfejsa. Radi diskusije i primera ispod, samo ce se imena interfejsa razlikovati izmedju PPP podesavanja i podesavanja za Ethernet, i dacemo sve od sebe da se sto pre otrasimo pravih imena mreznih interfejsa.
Prvo, moramo da ukljucimo gatewaying da bi kompjuter mogao da prosledjuje mrezni saobracaj, koji prima na jednom interfejsu, prema ostalim mrezama preko posebnog interfejsa. Za pocetak uradicemo ovo u komandoj liniji sa sysctl, za tradicionalnu *IP verziju cetiri*

```
# sysctl net.inet.ip.forwarding=1
```

a ako trebamo da prosledjujemo saobracaj *IP verzije sest*, onda je komanda

```
# sysctl net.inet6.ip6.forwarding=1
```

Da bi ovo ponovno radilo i kada restartujete kompjuter, treba da unesete ova podesavanja u relevantne konfiguracione fajlove.
U OpenBSD-u i NetBSD-u, ovo radite tako sto editujete /etc/sysctl.conf, menjajuci linije koje vam trebaju, ovako

```
net.inet.ip.forwarding=1
net.inet6.ip6.forwarding=1
```

U FreeBSD-u, ovo obicno radite unosenjem odgovarajucih promena dodajuci ove linije u vas /etc/rc.conf

```
gateway_enable="YES" #for ipv4
ipv6_gateway_enable="YES" #for ipv6
```

Efekat na mrezi je indentican, FreeBSD rc skripta postavlja dve vrednosti preko sysctl komande. Ipak, veci deo FreeBSD konfiguracije je se nalazi u rc.conf fajlu.
Dali su oba interfejsa koje nameravate da koristite podesena i rade? Koristite ifconfig -a, ili ifconfig *ime_interfejsa* da biste saznali.
Ako jos uvek zelite da dozvolite sav saobracaj zapocet od kompjutera sa vase mreze, vas /etc/pf.conf moze otprilike izgledati ovako[7]:

```
ext_if = "xl0" # makro za spoljni interfejs – koristite tun0 za PPPoE
int_if = "xl1"  # makro za unutrasnji interfejs
```

```
# ext_if IP adresa moze biti dinamicka, zbog toga ($ext_if)
nat on $ext_if from $int_if:network to any -> ($ext_if)
block all
pass from { lo0, $int_if:network } to any keep state
```

Primetili ste da smo koristili makroe da bi dodelili logicka imena mreznim interfejsima. Ovde su korsicene 3Com kartice, ali ovo je zadnji put tokom predavanja da ce ovo biti od nekog znacaja. U jednostavnim podesavanjima ko sto su ova, mozda ne mozemo dobiti mnogo koriscenjem ovakvih makroa, ali kada nam skup pravila naraste, naucucete da cenite citkost koju oni daju skupu pravila.
Pogledajte i nat pravilo. Ovo je mesto gde se vrsi mrezno prevodjenje adresa sa ne-rutabilnih adresa unutar vase lokalne mreze u jedinstvenu oficijalnu adresu koja vam je dodeljena.
Zagrade koje okruzuju zadnji deo nat pravila ($ext_if) sluze u slucaju da je IP adresa spoljnjeg intefejsa dinamicki dodeljena. Ovaj detalj ce osigurati da vas mrezni saobracaj protice bez ozbiljnih prekida cak iako se spoljna IP adresa menja.
Ali, ovaj skup pravila verovatno dozvoljava vise mreznog saobracaja nego sto vi zelite da izadje sa vase mreze. Gde ja radim, slican makro je

```
client_out = "{ ftp-data, ftp, ssh, domain, pop3, auth, nntp, \
               https, 446, cvspserver, 2628, cvsup, 8000, 8080 }"
```

sa pravilom

```
pass inet proto tcp from $int_if:network to any port $client_out \
    flags S/SA keep state
```

Ovo moze biti individualna selekcija portova, ali to je tacno ono sto meni i mojim kolegama treba. Neki od nabrojanih portova su potrebni za sisteme o kojima ne mogu dalje da pricam. Vase se potrebe verovatno razlikuju u nekim pojedinostima, ali ovo bi trebalo da pokrije barem neke od korsinih servisa.
Dalje, imamo jos nekoliko drugih pass pravila. Veoma brzo cemo se vratiti na neke od najinteresantnijih. Jedno od pass pravila koje je korisno onima koji zele mogucnost administriranja svojih kompjutera sa udaljene lokacije je

```
pass in inet proto tcp from any to any port ssh
```

ili konkretnije

```
pass in inet proto tcp from any to $ext_if port ssh
```

koje god zelite. Na kraju moramo da dozvolimo rad servisa imena za nase klijente

```
udp_services = "{ domain, ntp }"
```

dopunjeno pravilom koje dozvoljava saobracaj koji nam treba kroz nas zastitni zid:

```
pass quick inet proto { tcp, udp } to any port $udp_services keep state
```

Primetite rec **quick** u ovom pravilu. Poceli smo da pisemo pravila koja se sastoje od vise pravila, i vreme je da pogledamo povezanost pravila u skupu pravila. Pravila se proveravaju odozgo nadole, u nizu u kojem su napisani u konfiguracionom fajlu. Za svaki paket ili konekciju proverenu od strane PF-a*, zadnje pravilo koje se podudara* u skupu pravila je ono koje se primenjuje. Rec *quick* nudi izlaz iz uobicajene rutine. Kada se paket podudari sa quick pravilom, paket se tretira prema datom pravilu. Provera kroz pravila se zaustavlja bez rezmatranja dali ce neka sledeca pravila da se podudaraju sa

paketom. Voma zgodno ako vam treba nekoliko izolovanih izuzetaka u vasim pravilima. Ovo vodi racuna i o ntp-u, koji se koristi za sinhronizaciju vremena. Jedna zajednicka stvar za oba protokola je da oni mogu, pod izvesnih okolnostima, da komuniciraju nezimenicno preko TCP-a i UDP-a.

Taj stari tuzni ftp

Kratka lista TCP portova koje smo dotakli malopre su sadrzala, izmedju ostalih, i FTP. FTP je star i problematican, posebno za one koji pokusavaju da kombinuju FTP i zastitne zidove. FTP je star i cudan protocol, sa mnogo toga nepozeljnog. Neka od najcescih stvari protiv su

- Lozinke se prenose nezasticene
- Protokol zahteva koriscenje barem dve TCP konekcije (control i data) na odvojenim portovima
- Kada se uspostavi konekcija, podaci komuniciraju preko portova koji su izabrani nasumice

Sve ovo navedeno cini da se obrati paznja na sigurnost, cak i pre razmisljanja o potencijalnoj slabosti u klijentskom ili serverskom softveru koja moze dovesti do bezbednostih problema.. Ovakve stvari su se desavale.
Postoje moderije i sigurnije opcije za transfer fajlova, kao sftp ili scp, koji imaju opcije autentifikacije i prenos podataka preko enkriptovanih konekcija. Kompetentni IT profesionalci bi trebalo da koriste neke od ostalih oblika transfera podataka a ne FTP.
Bez obzira na nasu profesionalnost ili afinitete, svi smo svesni da nekada moramo da radimo neke stvari koje ne zelimo. U slucaju FTP-a kroz zastitne zidove, glavni deo se sastoji u tome da se saobracaj prosledjuje malom programu koji je napisan posebno za ovu namenu.

FTP kroz NAT: ftp-proxy

Samo OpenBSD 3.8 ili raniji

Novembra 2005, stari ftp-proxy (/usr/libexec/ftp-proxy) je zamenjen u OpenBSD-current sa novim ftp-proxy, koji je u /usr/sbin. Ovo je softver koji je ukljucen u OpenBSD 3.9 i novijim. Pogledajte *Odeljak ftp-proxy, Novi stil* za vise detalja.

ftp-proxy je deo osnovnog sistema na OpenBSD i ostalim sistemima koji imaju PF, i obicno se poziva preko inetd "super server" preko odredjenog unosa u /etc/inetd.conf._
Ovo pod navodnicima odredjuje da ftp-proxy radi u NAT modu na loopback interfejsu, lo0:

```
127.0.0.1:8021 stream tcp nowait root /usr/libexec/ftp-proxy \
  ftp-proxy –n
```

Ova linija je standardno podrazumevana u vasem inetd.conf, iskljucena sa # karakterom na pocetku linije. Da biste omogucili vase promene. restartujte inetd.
U FreeBSD, NetBSD i ostalim BSD-ovima zasnovanim na rcng ovo radite komandom

```
FreeBSD$ sudo /etc/rc.d/inetd restart
```

Ili slicnom. Pogledajte man 8 inetd ako niste sigurni.
OpenBSD rc sistem je vise tradicionalniji, i komanda koja vam treba je

```
OpenBSD$ sudo kill -HUP `cat /var/run/inetd.pid`
```

Sada je inetd pokrenut, sa vasim novim podesavanjima.
A sada konkretno o preusmeravanju. Pravila za preusmeravanje i NAT pravila spadaju u istu klasu pravila. Druga pravila mogu direktno pozivati ova pravila, i pravila za filtriranje mogu zavisiti od ovih pravila. Logicki, rdr i nat pravila treba da se definisu pre pravila za filtriranje.
Stavicemo nase rdr pravilo odmah nakon nat pravila u nasem /etc/pf.conf

```
rdr on $int_if proto tcp from any to any port ftp -> 127.0.0.1 \
        port 8021
```

Sada i preusmereni saobracaj treba da se pruposti van. Ovo postizemo sa

```
pass in on $ext_if inet proto tcp from port ftp-data to ($ext_if) \
    user proxy flags S/SA keep state
```

Sacuvajte pf.conf, onda ucitajte nova pravila sa

```
$ sudo pfctl -f /etc/pf.conf
```

Verovatno cete imati korisnike koji su primetili da FTP radi pre nego sto uspete da im objasnite sta ste uradili.
Ovo je primer koriscenja NAT-a na gateway-u sa ne-rutabilnim adresama unutar mreze.

FTP kroz pf sa rutabilnim adesama: ftpsesame, pftpx i ftp-proxy!

U slucajevima kada lokalna mreza koristi javne, rutabilne adrese unutar zastitnog zida, moram priznati da sam imao posteskoca da nateram da ftp-proxy radi ispravno. Kada sam potrosio previse vremena na ovom problemu, laknulo mi je kada sam pronasao resenje za ovaj specifican problem koje je ponudio Holandjanin po imenu Camiel Dobbelaar u formi daemona nazvanog ftpsesame. Lokalne mreze koje koriste javne adrese unutar zastitnog zida su veoma retke tako da cu ovo preskociti. Ako vam ovo treba i radite na OpenBSD-u ili ranijim ili na nekim od operativnih sistema koji imaju PF, mozete da skinete ftpsesame sa Sentia-e na http://www.sentia.org/projects/ftpsesame/.
ftpsesame se uvlaci u vas skup pravila preko anchor, to je ime za pod-skup pravila. Dokument se sastoji od man stranice sa primerima koje cete verovatno samo prekopirati.
ftpsesame nikada nije ubacen u osnovni sistem, i Camiel je napisao novo resenje za iste probleme. Novi program, prvo nazvan pftpx, 0.8 je dostupan na http://www.sentia.org/downloads/pftpx-0.8.tar.gz. Sledeca razvojna verzija, preimenovana u novi ftp-proxy, bice deo OpenBSD 3.9 kao /usr/sbin/ftp-proxy.

---

ftp-proxy, novi stil

Za OpenBSD 3.9 i novije

Ako vrsite upgrejd do OpenBSD 3.9 ili radite sa svezeg 3.9 ili novije instalacije, ovo je verzija ftp-proxy koju treba da koristite.

Kao i njen predhodnik, konfiguracija ftp-proxy, naslednika pftpx, je samo stvar kopiranja sa man stranice.
Ako vrsite upgrade do novog ftp-proxy sa starije verzije, morate da uklonite ftp-proxy liniju iz vaseg inetd.conf fajla i da restartujete inetd.
Sledece, pokrenite ftp-proxy tako sto cete dodati sledecu liniju vasem /etc/rc.conf.local ili /etc/rc.conf
ftpproxy_flags=""
Mozete startovati proxy manuelno ako zelite tako sto cete pokrenuti /usr/sbin/ftp-proxy.
Prelazimo na pf.conf fajl, trebaju vam dve definicije anchor u NAT delu:

```
nat-anchor "ftp-proxy/*"
rdr-anchor "ftp-proxy/*"
```

Oba su potrebna, cak iako vasa podesavanja ne korsite NAT. Ako prelazite sa predhodne verzije, vas skup pravila verovatno vec sadrzi odgovarajuca preusmerenja. U suprotnom, dodajte:

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

Idemo dole prema pravilima filtriranja, dodajte anchor za proxy,

```
anchor "ftp-proxy/*"
```

i konacno pravilo za propustanje da bi dozvolili paketima prolazak od proxy-ja do ostatka sveta.

```
pass out proto tcp from $proxy to any port 21 keep state
```

gde se $proxy siri do adrese za koje je proxy daemon vezan.
Ovaj primer pokriva jednostavna podesavanja sa klijentima koji kontaktiraju razne FTP servere. Za ostale varijacije i komplikovanija podesavanja, pogledajte ftp-proxy man stranicu.

Vasa mreza bez problema

Podizanje mreze na kojoj ce se lako resavati problemi je potencijalno velika stvar. U vecini slucajeva, debugging ili lakoca razresavanje problema vase TCP/IP mreze zavisi od toga kako se odnosite prema Internet protokolu koji je posebno dizajniran sa debugging-om na umu, *Internet Control Message Protocol*, ili *ICMP* kako se obicno naziva skraceno.
ICMP je protocol za slanje i primanje *kontrolnih poruka* izmedju hostova i gateways, vecinom da bi se omogucile povratne informacije posiljaocu o neobicnim uslovima ili poteskocama na putu to hosta kome je namenjen. Postoji dosta ICMP saobracaja koji se obicno odigrava u pozadini dok recimo surfujete web-om, citate postu ili saljete podatke. Ruteri (svesni ste da upravo pravite jedan zar ne?) koriste ICMP da bi saznali velicinu paketa i ostale prenosne parametre u procesu koji se cesto naziva *path MTU discovery*.
Mozda ste culi administratore koji govore o ICMP-u kao “cistom zlu”, ili, ako malo su malo napredniji, “nuznom zlu”. Razlog za ovakav stav je cisto istorijski. Odgovor se moze naci nekoliko godina unazad kada je otkriveno da je nekoliko operativnih sistema sadrzalo kod u njihovom mreznom stack-u koji moze da obori system, ili da u nekim slucajevima radi neke cudne stvari, sa dovoljno velikim ICMP zahtevom.

Jedna od kompanija koje su primile tezak udarac je bio Microsoft, i mozete naci dosta materijala na temu “ping-of-death” bug-a koristeci vas pretrazivac. Ovo se sve desilo u drugoj polovini devedesetih, i svi moderni operativni sistemi su od tada detaljno sanirali njihom mrezni kod. To je barem ono u sta treba da verujemo.
Jedno od ranijih resenja je bilo da se jednostavno blokira sav ICMP saobracaj ili barem ICMP ECHO, sto je ono sto ping koristi. Ovo su pravila koja su tu otprilike deset godina, i ljudi koji su ih tu stavili su jos uvek uplaseni.

Onda, hocemo li sve propustiti?

Ocigledno pitanje onda postaje, ako je ICMP tako dobra i korisna stvar, zar je necemo propustiti kompletno, svo vreme? Odgovor je, “Zavisi”.
Propustajuci dijagnosticki saobracaj bez pogovora naravno olaksava debugging, ali isto tako relativno lako omogucava ostalima da dobiju informacije o vasoj mrezi. To znaci da pravilo

```
pass inet proto icmp from any to any
```

mozda nece biti optimalno ako zelite da obavijete interna desavanja vase mreze u malo misterije. Ako smo vec iskreni, treba reci i da mozete naci da neki ICMP saobracaj veoma bezopasno prolazi sa vasim keep state pravilima.

Lako resenje: brbljanje ovde prestaje

Najjednostavnije resenje moze biti da da dozvolite sav izlazni ICMP saobracaj koji dolazi od vase lokalne mreze a da spoljna istrazivanja blokirate na vasem gateway-u:

```
pass inet proto icmp icmp-type $icmp_types from $int_if:network \
        to any keep state
pass inet proto icmp icmp-type $icmp_types from any to $ext_if \
        keep state
```

Zaustavljajuci istrazivanja na gateway-u mogu biti atraktivna opcija, ali da pogledamo nekoliko drugih opcija koje ce nam pokazati fleksibilnost koju PF ima.

Propustanje komande ping

Skup pravila koji smo sastavili do sada ima jednu ociglednu manu: uobicajene komande za otklanjanje problema kao sto su ping i traceroute nece raditi. To mozda nece mnogo znaciti vasim korisnicima, i posto je ping komanda bila ta koja je preplasila ljude koji su bili za filtriranje ili blokiranje ICMP saobracaja, ocigledno postoje oni koji misle da nam je bolje bez nje. Ako vi spadate u onu grupu koja me razume, bicete vise nego naklonjeni tome da imate te alate za otklanjanje gresaka dostupne. Sa nekoliko malih dodataka u nasem skupu pravila i bicete. ping koristi ICMP, da bi nas skup pravila bio idalje sredjen, pocecemo definisanjem jos jendog makroa:

```
icmp_types = "echoreq"
```

i pravila koje koristi definiciju,

```
pass inet proto icmp all icmp-type $icmp_types keep state
```

Ako vam zatreba da propustite vise ili druge tipove ICMP paketa, onda mozete da prosirite icmp_types u listu tih paketa koje zelite da propustite.

Pomazuci komandi traceroute

traceroute je jos jedna komanda koja je veoma korisna kada vam vasi korisnici kazu da Internet ne radi. Podrazumevano, Unix traceroute koristi UDP konekcije prema setu formula na osnovu destinacije. Pravilo ispod radi sa traceroute komandom na svim unix-ima na kojima sam imao prilike da je isprobam, ukljucujuci GNU/Linux:

```
# allow out the default range for traceroute(8):
# "base+nhops*nqueries-1" (33434+64*3-1)
pass out on $ext_if inet proto udp from any to any \
        port 33433 >< 33626 keep state
```

Dosadasnje iskustvo pokazuje da implementacija traceroute na ostalim operativnim sistemima radi veoma slicno. Osim, naravno, Microsoft Windows. Na toj platformi, TRACERT.EXE koristi ICMP ECHO za ovu namenu. Tako da ako zelite da propustite Windows traceroute, treba vam samo prvo pravilo. Unix traceroutes moze biti poducen da koristi i druge protokole, i ponasace se isto kao njegova Microsoft kopija ako koristite njegovu -i opciju u komandnoj liniji. Mozete pogledati traceroute man stranicu (ili njegov izvorni kod) za vise detalja.
Ovo resenje je uzeto sa openbsd-misc posta. Mislim da su liste, i pretrazive arhiva listova (dostupne na razlicitim mestima sa http://marc.theaimsgroup.com/), veoma dragocen izvor kad god vam trebaju informacije vezane za OpenBSD ili PF.

Path MTU discovery

Zadnja stvar o kojoj cu vam pricati kada se radi o otklanjanju problema je 'path MTU discovery'. Internet protokoli su dizanirani da budu nezavisni bez obzira na uredjaj, i jedna od posledica toga je da nemozete uvek predvideti sa sigurnoscu koja je optimalna velicina paketa za datu konakciju. Glavna prepreka za vasu velicinu paketa se naziva *Maksimalna Transmisiona Jedinica*, ili *MTJ* (MTU*)* koja odredjuje gornju granicu velicine paketa za interfejs. Komanda ifconfig pokazace MTJ za vas mrezni interfejs.
Moderne TCP/IP implementacije ocekuju da budu u mogucnosti da mogu da odrede tacnu velicinu paketa za konekciju kroz process koji jednostavno ukljucuje slanje paketa razlicitih velicina sa “Ne fragmentiraj” oznakama, ocekujuci ICMP povratni paket oznacavajuci “tip 3, kod 4”, kada je gornja granica postignuta. Te trebate odmah da trazite odgovor u RFC-u. Tip 3 znaci “nedostupno odrediste”, dok kod 4 je ukratko “fragmentacija potrebna, ali je ne fragmentiraj oznaka postavljena”. Znaci ako vam se cini da su vase konekcije prema mrezama koje mogu imati drugaciji MTJ neoptimalne, i ne morate da budete toliko odredjeni, pokusajte da promenite malo listu tipova ICMP da biste dozvolili i pakete sa nedostupnim odredistem:

```
icmp_types = "{ echoreq, unreach }"
```

kao sto vidimo, ovo znaci da ne moramo da menjamo pass pravilo uopste:

```
pass inet proto icmp all icmp-type $icmp_types keep state
```

PF vam omogucava da filtrirate sve varijacije ICMP tipova i kodova, i ako zelite da kopate po tome koji ICMP saobracaj da propustite a koji ne, lista mogucih tipova i kodova su dokumentovani u icmp(4) i icmp6(4) man stranicama. Dalje informacije su dostupne u RFC-ima[8] .

Higijena: block-policy, scrub i antispoof

Nas gateway se ne cini kompletnim bez jos nekoliko drugih stvari u konfiguraciji koje ce ga ciniti da se ponasa malo razumnije prema hostovima na Internetu i vasoj lokalnoj mrezi.
block-policy je opcija koja se moze podesiti u options delu skupa pravila, koji se nalazi ispred pravila preusmeravanja i filtriranja. Ova opcija odreduje koje povratne informacije, ako ih bude, ce PF dati hostovima koji pokusavaju da zapocnu konekcije koje su blokirane. Opcija ima dve moguce vrednosti, drop koji ispusta blokirane pakete bez ikakvih povratnih informacija, i return koja se vraca sa statusnim kodom kao npr. Konekcija odbijena i slicno.
Pravilna strategija politike blokiranja je bila tema dosta diskusija. Mi smo odabrali da budemo ljubazni i da uputimo nas zastitni zid da izdaje povratne informacije:

```
set block-policy return
```

scrub je rec koja omogucuje normalizaciju mreznih paketa, sastavljajuci fragmentirane pakete i uklanjajuci dvosmislenost. Ukljucujuci scrub pruzate odredjenu meru zastite protiv odredjenih tipova napada koji se zasnivaju na nepravilnom manipulisanju fragmentiranim paketima. Dodatne opcije su isto dostupne, ali smo mi odabrali najednostavniji oblik koji je pogodan za vecinu konfiguracija.

```
scrub in all
```

Neki servisi, kao sto je NFS, traze specificne opcije za manipulisanje fragmentima. Ovo je opsezno objasnjeno u PF uputstvu, a i man stranice pruzaju sve informacije koje vam mogu zatrebati.
antispoof je cest poseban slucaj filtriranja i blokiranja. Ovaj mehanizam stiti od delovanja laznih ili nepostojecih IP adresa, pretezno blokirajuci pakete koji dolaze na interfejs i u pravcima koji logicki nisu moguci. Odredicemo da hocemo da iskorenimo lazni saobracaj koji dolazi spolja i bilo koji lazni paket koji, a to je malo verovatno, potice sa nase mreze:

```
antispoof for $ext_if
antispoof for $int_if
```

Ovim kompletiramo nas jednostavi zastitni zid koji prevodi mrezne adrese za malu lokalnu mrezu.

Web server i mail server unutar mreze

Vreme prolazi, i potrebe se menjaju. Veoma cesto se javlja potreba za servisima koji trebaju biti dostupni spolja. Ovo cesto postaje malo teze zato sto adrese vidljive spolja su ili nedostupne ili preskupe, a pokretanje nekoliko drugih servisa na masini cija je osnovna uloga zastitni zid nije bas pozeljna opcija.

Mehanizam preusmeravanja u PF-u omogucava da vrlo lako imate server na unutrasnjoj mrezi. Ako pretpostavimo da nam treba web server koji sadrzi podatke u cistom tekstu (http) i enkriptovano (https) i plus zelimo mail server koji salje i prima e-mail poruke, a da dozvolite klijentima unutar i izvan lokalne mreze da koriste neke od poznatih protokola za slanje i primanje, sledece linije mogu biti ono sto vam je potrebno kao dopuna skupu pravila koji smo ranije razvili:

```
webserver = "192.168.2.7"
webports = "{ http, https }"
emailserver = "192.168.2.5"
email = "{ smtp, pop3, imap, imap3, imaps, pop3s }"

rdr on $ext_if proto tcp from any to $ext_if port \
      $webports -> $webserver
rdr on $ext_if proto tcp from any to $ext_if port \
      $email -> $emailserver

pass proto tcp from any to $webserver port $webports \
  flags S/SA synproxy state
pass proto tcp from any to $emailserver port $email \
  flags S/SA synproxy state
pass proto tcp from $emailserver to any port smtp \
  flags S/SA synproxy state
```

Primetite oznaku 'synproxy' u novim pravilima. Ovo znaci da ce PF upravljati konekcijom (trostruko rukovanje) umesto servera ili klijenta pre nego prosledi konekciju do aplikacije. Ovo omogucava odredjenu dozu zastite protiv pojedinih vrsta napada.
Skupovi pravila za konfiguraciju sa DMZ mrezama izolovanih iza odvojenih mreznih interfejsa i u nekim slucajevima servisa koji rade na alternativnim portovima se nece mnogo razlikovati u odnosu na ove.

Briga o svojima - unutrasnjost

Sve sto sam do sada rekao je tacno dokle god je ono sto vas interesuje kako da saobracaj sa hostova izvan vase lokalne mreze prosledite do vasih servera.
Ako zelite da hostovi u vasoj lokalnoj mrezi budu u mogucnosti da koriste servise na ovim masinama, ubrzo cete videti da saobracaj koji potice iz vase lokalne mreze nikada nece stici do spoljnjeg interfejsa. Spoljni interfejs je taj gde se desavaju sva preusmeravanja i prevodjenja, i prema tome ne mozete preusmeravati od iznutra. Ovaj problem je toliko cest da PF dokumentacija daje cetiri razlicita resenja za ovaj problem.[9]
Opcije koje su navedene u PF prirucniku su

- ‘Odvojeni Horizont' DNS, sto znaci konfigurisanje vaseg servisa imena da bi omogucio jedan set odgovora na zahteve koji poticu iz lokalne mreze i jos jedan za ostale zahteve
- Koriscenjem proxy softvera kao npr nc (NetCat)
- Odnoseci se prema lokalnoj mrezi kao prema posebnom slucaju za preusmeravanje i NAT. Razmotricemo ove opcije ubrzo.
- Ili jednostavno pomerajuci vase servere na odvojenu mrezu, takozvanu 'DMZ', sa samo manjim promenama u vasim PF pravilima.

Moramo da presretnemo mrezne pakete koji dolaze sa lokalne mreze i da se ispravno upravlja tim konekcijama, vodeci racuna da se sav povratni saobracaj prosledjuje komunikacionom partneru koji je zapravo i zapoceo konekciju.
Vracajuci se na nas predhodni primer, ovo postizemo tako sto dodajemo ova specijalna pravila:

```
rdr on $int_if proto tcp from $int_if:network to $ext_if \
      port $webports -> $webserver
rdr on $int_if proto tcp from $int_if:network to $ext_if \
      port $email -> $emailserver
no nat on $int_if proto tcp from $int_if:network to $int_if:network
nat on $int_if proto tcp from $int_if:network to $webserver \
      port $webports -> $int_if
nat on $int_if proto tcp from $int_if:network to $emailserver \
      port $email -> $int_if
```

Tabele vam olaksavaju zivot

Mozete pomisliti da ovo postaje odvratno staticno i kruto. Desavace se da postoje neki podaci koji su bitni za filtriranje i preusmeravanje u datom momentu, ali ne zasluzuju da budu stavljeni u konfiguracioni fajl! Veoma tacno, i PF nudi mehanizme za razresenje i ovakvih situacija. Tabele su jedna od tih opcija, prvenstveno korisne kao liste kojima se moze manipulisati bez potrebe za ucitavanjem celog skupa pravila, i gde je pozeljno brzo pretrazivanje. Imena tabela su uvek unutar zagrada < >, kao npr:

```
table <clients> { 192.168.2.0/24, !192.168.2.5 }
```

ovde, mreza 192.168.2.0/24 je deo tabele, osim adrese 192.168.2.5, koja je iskljucena koriscenjem ! znaka (logicko NE). Isto tako je moguce da se ucitaju tabele iz fajlova gde je svaki deo na posebnoj liniji, kao sto je fajl /etc/clients

```
192.168.2.0/24
!192.168.2.5
```

koji se koristi da pokrene tabelu u /etc/pf.conf:

```
table <clients> persist file /etc/clients
```

Onda, na primer, mozete promeniti jedno od vasih ranijih pravila da izgleda kao

```
pass inet proto tcp from <clients> to any port $client_out \
       flags S/SA keep state
```

da biste upravljali odlaznim saobracajem sa kompjutera vasih klijenata. Sa ovim, mozete manipulisati sadrzajem tabela na zivo, kao

```
$ sudo pfctl -t clients -T add 192.168.1/16
```

Ovo menja samo kopiju tabele koja se nalazi u memoriji, sto znaci da ako dodje do pada sistema ili ponovnog pokretanja promene se nece odrzati dok ne sacuvate promene.

Ako zelite da odrzavate kopiju tabele na disku mozete koristiti cron job koji stavlja sadrzaj tabele na disk u odredjenim intervalima, koristeci komandu pfctl -t clients -T show >/etc/clients. Alternativno, mozete editovati /etc/clients fajl i zameniti sadrzaj tabele u memoriji sa podacima iz fajla:

```
$ sudo pfctl -t clients -T replace -f /etc/clients
```

Za operacije koje cesto izvrsavate, pre ili kasnije moracete da pocnete da pisete shell skripte za zadatke kao sto su ubacivanje ili brisanje, ili menjanje sadrzaja tabela. Jedino pravo ogranicenje lezi u vasim potrebama ili vasoj kreativnosti.
Ubrzo cemo se vratiti na ostale korisne primene tabela.

Logiranje

Nismo pominjali mnogo logiranje do sada. PF daje mogucnosti da belezite tacno ono sto vi zelite dodajuci rec ‘log’ pravilima koje zelite da belezite. Mozda zelite da malo ogranicite kolicinu podataka tako sto cete odrediti jedan interfejs gde ce se logiranje odvijati. Ovo postizete tako sto dodajete

```
set loginterface $ext_if
```

i onda editujete pravila koja zelite da logujete, kao

```
pass out log from <client> to any port $email \
        label client-email keep state
```

Ovim se saobracaj belezi u binarnom formatu koji je samo namenjen da se koristi kao unos za tcpdump.
label deo kreira novi skup brojaca za razlicite statisticke podatke u pravilu. Ovo moze biti veoma zgodno ako npr. nadgledate ostale koliko protoka trose.

Kratak pogled sa tcpdump

Kada ste ukljucili belezenje za jedno ili vise pravila, PF belezi preko pflog0 interfejsa, i cuva binarni log podataka u log fajlu /var/log/pflog. Log fajl je koristan za stalno belezenje i u slucajevima kada zelite da s vremena na vreme konvertujete neke podatke u druge formate. Ipak, ako zelite da posmatrate vas saobracaj u realnom vremenu, mozete reci tcpdump-u da umesto toga posmatra pflog0 interfejs. Evo kako mogu izgledati izlazni podaci nekoliko pravila letnjeg poslepodnevnog Cetvrtka:

```
peter@skapet:~$ sudo tcpdump -n -e -ttt -i pflog0
tcpdump: WARNING: pflog0: no IPv4 address assigned
tcpdump: listening on pflog0, link-type PFLOG
Feb 16 16:43:20.152187 rule 0/(match) block in on xl0: 194.54.59.189.2559 >
194.54.107.19.139: [|tcp] (DF)
Feb 16 16:48:26.073244 rule 27/(match) pass in on xl0: 61.213.167.236 >
194.54.107.19: icmp: echo request
Feb 16 16:49:09.563448 rule 0/(match) block in on xl0: 61.152.249.148.80 >
194.54.107.19.55609: [|tcp]
Feb 16 16:49:14.601022 rule 0/(match) block in on xl0: 194.54.59.189.3056 >
194.54.107.19.139: [|tcp] (DF)
Feb 16 16:53:10.110110 rule 0/(match) block in on xl0: 68.194.177.173 >
```

194.54.107.19: [|icmp]
Feb 16 16:55:54.818549 rule 27/(match) pass in on xl0: 61.213.167.237 >
194.54.107.19: icmp: echo request
Feb 16 16:57:55.577782 rule 27/(match) pass in on xl0: 202.43.202.16 >
194.54.107.19: icmp: echo request
Feb 16 17:01:27.108404 rule 0/(match) block in on xl0: 194.54.59.189.4520 >
194.54.107.19.139: [|tcp] (DF)
Feb 16 17:11:02.137310 rule 0/(match) block in on xl0: 222.73.4.154.80 >
194.54.107.19.55609: [|tcp]
Feb 16 17:14:05.739403 rule 0/(match) block in on xl0: 194.54.174.246.3970 >
194.54.107.19.135: [|tcp] (DF)
Feb 16 17:14:08.715163 rule 0/(match) block in on xl0: 194.54.174.246.3970 >
194.54.107.19.135: [|tcp] (DF)
Feb 16 17:14:09.308355 rule 0/(match) block in on xl0: 194.54.174.246.3970 >
194.54.107.19.135: [|tcp] (DF)
Feb 16 17:19:01.853730 rule 27/(match) pass in on xl0: 203.84.214.5 >
194.54.107.19: icmp: echo request
U PF prirucniku postoji deo posvecen belezenju koji sadrzi veliki broj veoma korisnih predloga. U kombinaciji sa, izmedju ostalog, tcpdump man stranicama, mocicete da izvucete bilo koje log podatke koji vam mogu biti korisni.

Ali postoje ogranicenja (anegdota)

Isprva moze vam izgledati privlacno da stavite nesto kao

```
block log all
```

- za svaki slucaj da nesto ne propustite.
PF prirucnik sadrzi detaljan opis kako napraviti PF log u tekst formatu koriteci syslog, i ovo zvuci veoma privlacno. Prosao sam kroz proceduru opisanu tamo kada sam podesavao moju prvu PF konfiguraciju na poslu, i brzo shvatio: Belezenje je korisno, ali budite selektivni. Nakon malo vise od sat vremena, PF tekst log je narastao na vise od gigabajta, na masini sa manje od deset gigabajta ukupnog prostora na disku.
Objasnjenje lezi u tome da, cak i kada se Internet ne koristi u velikoj meri, na kraju obicne ADSL veze i dalje postoji neverovatna kolicina nekontrolisanog Windows saobracaja kao sto je deljenje fajlova i razni tipovi pretraga koji pokusavaju da dodju do vas. Sigurno je da ni Windows masine unutar mreze nisu tako mirne. Bilo kako bilo: razumno ogranicite sta belezite, ili obezbedite dovoljno prostora na disku.

Drzati stvari na oku komandom pftop

Ako vas interesuje sta ulazi i izlazi sa vase mreze, alatka pftop Can Erkin Acara je veoma korisna. Ime veoma asocira na to sta ona radi - pftop prikazuje radni snimak vaseg saobracaja u formatu koji je veoma inspirisan top-om(1):

```
pfTop: Up State 1-21/67, View: default, Order: none, Cache: 10000       19:52:28

PR   DIR SRC                DEST               STATE   AGE   EXP  PKTS BYTES
tcp  Out 194.54.103.89:3847   216.193.211.2:25      9:9     28    67    29  3608
tcp  In  207.182.140.5:44870  127.0.0.1:8025        4:4     15 86400    30  1594
```

```
tcp  In  207.182.140.5:36469  127.0.0.1:8025       10:10   418    75   810 44675
tcp  In  194.54.107.19:51593  194.54.103.65:22      4:4    146 86395   158 37326
tcp  In  194.54.107.19:64926  194.54.103.65:22      4:4    193 86243   131 21186
tcp  In  194.54.103.76:3010   64.136.25.171:80      9:9    154    59    11  1570
tcp  In  194.54.103.76:3013   64.136.25.171:80      4:4     4 86397     6  1370
tcp  In  194.54.103.66:3847   216.193.211.2:25      9:9    28    67    29  3608
tcp  Out 194.54.103.76:3009   64.136.25.171:80      9:9    214     0     9  1490
tcp  Out 194.54.103.76:3010   64.136.25.171:80      4:4     64 86337     7  1410
udp  Out 194.54.107.18:41423  194.54.96.9:53        2:1     36     0     2   235
udp  In  194.54.107.19:58732  194.54.103.66:53      1:2     36     0     2   219
udp  In  194.54.107.19:54402  194.54.103.66:53      1:2     36     0     2   255
udp  In  194.54.107.19:54681  194.54.103.66:53      1:2     36     0     2   271
```

Vase konekcije mogu biti razvrstane po raznim kriterijumima, izmedju ostalog i po PF pravilu, velicini, starosti i tako dalje.
Ovaj program nije u samom osnovnom sistemu, ali je u portovima u OpenBSD-u i FreeBSD-u kao /usr/ports/sysutils/pftop, a na NetBSD-u preko pkgsrc kao sysutils/pftop.

Nevidljivi gateway - most

Most u nasem kontekstu je masina sa dva ili vise interfejsa, lociranim izmedju Interneta i jedne ili vise unutrasnjih mreza, i mreznim interfejsima nije dodeljena IP adresa. Ako ta masina radi na OpenBSD-u ili slicnom operativnom sistemu, jos uvek je moguce da se filtrira i preusmerava saobracaj.
Prednost takve structure je da je napad na zastitni zid tezi. Nedostatak je da se svi administrativni zadaci moraju obavljati u konzoli zastitnog zida, osim ako ne konfigurisete mrezni interfejs koji je dostupan preko nekog tipa osigurane mreze, ili cak serijske konzole.
Prava metoda za konfigurisanje mostova se razlikuje u nekim detaljima izmedju operativnih sistema. Dole je kratak recept za koriscenje na OpenBSD-u, koji u dobroj meri blokira sav saobracaj koji nije Internet protocol saobracaj. Podesavanje mosta sa dva interfejsa:

```
/etc/hostname.xl0
up
/etc/hostname.xl1
up
/etc/bridgename.bridge0
  add xl0 add xl1 blocknonip xl0 blocknonip xl1 up
/etc/pf.conf
ext_if = xl0
int_if  = xl1
interesting-traffic = { ... }
block all
pass quick on $ext_if all
pass log on $int_if from $int_if to any port $interesting-traffic \
    keep state
```

i komplikovanija podesavanja su moguca. Iskusniji preporucuju da se odabere jedan interfejs na kojem ce se izvoditi sva filtriranja i preusmeravanja. Svi paketi prolaze kroz PF dva puta, stvarajuci tako potencijalno_ekstremono komplikovana pravila.
i OpenBSD brconfig komanda daje svoj skup opcija za filtriranje ako su vam potrebne dodatne konfiguracione opcije. bridge(4) i brconfig(8) man stranice daju dodatne informacije.
FreeBSD koristi malo drugaciji skup komandi za konfigurisanje mostova, dok NetBSD implementacija PF-a ne podrzava mostove.

Usmeravanje saobracaja preko ALTQ

ALTQ – skracenica za ALTernate Queueing – je veoma felksibilan mehanizam za upravljanje mreznim saobracajem koji je ziveo svoj zivot pre nego sto je integrisan u PF. Altq je bio jedna od onih stvari koje su integrisane u PF zbog dodatne pogodnosti koju je pruzao.
Altq koristi termin *queue* za glavni mehanizam kontrole saobracaja. Nizovi su definisani sa odredjenom kolicinom mreznog protoka ili specificnim delom dostupnog protoka, i njima se mogu dodavati podnizovi raznih tipova.
Da bismo ovo shvatili, pisacete filter pravila koja dodeljuju pakete specificnim nizovima ili selekciji podnizova gde ce paketi prolaziti prema odredjenim kriterijumima.
Nizovi se kreiraju sa jednom od nekoliko *disciplina* nizova. Osnovna disciplina niza bez ALTQ-a je FIFO (prvi ulazi, prvi izlazi). Malo interesantnija disiplina je disciplina koja se zasniva na klasi (CBQ), sto prakticno znaci da definisete protok niza kao zadatu kolicinu podataka u sekundi, u procentima ili u kilobajtima, megabajtima i tako dalje, sa opcijom dodatnog prioriteta, ili zasnovanog na prioritetu (priq), gde dodeljujete samo prioritete. Prioriteti mogu biti postavljeni od 0 do 7 za cbq nizove, od 0 do 15 za priq nizove, zadajuci veci prioritet i povlascenja vecim brojem. Dodatno, algoritamski hijerarhijski niz "Hierarchical Fair Service Curve" ili HFSC je dostupan. Ukratko, pojednostavljena sintaksa je

```
altq on interface type [options ... ] main_queue { sub_q1, sub_q2 ..}
  queue sub_q1 [ options ... ]
  queue sub_q2 [ options ... ]
[...]
pass [ ... ] queue sub_q1
pass [ ... ] queue sub_q2
```

Ako korsitite ove opcije u vasim skupovima pravila, obavezno procitajte pf.conf man stranice i PF prirucnik. Ova dokumentacija daje voma detaljna i razumljiva objasnjenja sintaksi i opcija.[10] [11]

ALTQ – rasporedjivanje prema procentima

Sada u prakticnom primeru, koji sam sa prakticnim namerama uzeo sa unix.se. Nizovi su podeseni na spoljnem interfejsu. Ovo je verovatno najcesci pristup, zato sto su ogranicenja protoka obicno veca na spoljnjem interfejsu. U principu, ipak, rasporednjivanje nizova i oblikovanje saobracaja moze biti izvedeno na bilo kojem mreznom interfejsu. Ovde, podesavanje ukljucuje cbq niz za ukupan protok od 640KB sa sest podnizova.

```
altq on $ext_if cbq bandwidth 640Kb queue { def, ftp, udp, http, \
        ssh, icmp }
queue def bandwidth 18% cbq(default borrow red)
queue ftp bandwidth 10% cbq(borrow red)
queue udp bandwidth 30% cbq(borrow red)
queue http bandwidth 20% cbq(borrow red)
queue ssh bandwidth 20% cbq(borrow red) { ssh_interactive, ssh_bulk }
queue ssh_interactive priority 7
queue ssh_bulk priority 0
queue icmp bandwidth 2% cbq
```

Vidimo da je podniz def sa 18 percenata protoka oznacen kao osnovni niz, to znaci da ako saobracaj nije eksplicitno dodeljen nekom drugom nizu zavrsava ovde. borrow i red znace da niz moze 'pozajmiti' protok od maticnog niza, dok sistem pokusava da izbegne zagusenje tako sto primenjuje RED (Random Early Detection) algoritam. Ostali nizovi manje ili vise prate isti sablon, do podniza ssh, koji sam ima dva podniza sa individualnim prioritetima.
Na kraju, pass pravila koja pokazuju koji ce saobracaj biti dodeljen nizovima, i njihovim kriterijumima:

```
pass log quick on $ext_if proto tcp from any to any port 22 flags S/SA \
   keep state queue (ssh_bulk, ssh_interactive)
pass in quick on $ext_if proto tcp from any to any port 20 flags S/SA  \
   keep state queue ftp
pass in quick on $ext_if proto tcp from any to any port 80 flags S/SA \
   keep state queue http
pass out on $ext_if proto udp all keep state queue udp
pass out on $ext_if proto icmp all keep state queue icmp
```

Mozemo pretpostaviti da ovakav raspored zadovoljava potrebe sajta.

ALTQ – prioritet po tipu saobracaja

Dolazimo do sledeceg primera, uzetog sa web stranice Daniela Hartmeiera. Kao i nekolicina nas, i Daniel je na asimentricnoj konekciji, i naravno zeleo je da dobije bolje korsicenje protoka.
Jedan stvar je posebno indicirala da postoji mesto za poboljsanje. Dolazeci saobracaj (download) je ocigledno usporavao izlazni saobracaj.
Analizirajuci podatke koji su pokazivali da ACK paketi za svaki preneti paket podataka prouzrokuju neproporcionalno veliko usporenje, verovatno zbog postojece FIFO (Prvi ulazi, prvi izlazi) discipline niza na odlaznom saobracaju.
Hipoteza se formirala – ako bi ACK paketi, koji su prakticno bez podataka, bili sposobni da se uvuku izmedju vecih paketa podataka, ovo ce voditi efikasnijem iskoriscenju dostupnog protoka. Cilj su bila dva niza sa razlicitim prioritetima. Vazni delovi skupa pravila slede:

```
ext_if="kue0"

altq on $ext_if priq bandwidth 100Kb queue { q_pri, q_def }
queue q_pri priority 7
queue q_def priority 1 priq(default)

pass out on $ext_if proto tcp from $ext_if to any flags S/SA \
       keep state queue (q_def, q_pri)

pass in  on $ext_if proto tcp from any to $ext_if flags S/SA \
       keep state queue (q_def, q_pri)
```

Rezultat je zaista bio poboljsanje performansi. Danielov clanak je dostupan na njegovoj web stranici http://www.benzedrine.cx/ackpri.html

---

ALTQ – upravljanje nezeljenim saobracajem

Nas zadnji altq primer je jedan koji je ziveo u vremenu jedne od mnogih spam ili virus oluja ciji smo svedoci bili tokom zadnjih nekoliko godina. Poznato je da su masine koje prouzrokuju ovakve izlive

email poruka uglavnom sve Windows masine. PF ima veoma pouzdan fingerprint mehanizam koji detektuje operativni system na drugom kraju mrezne konekcije. Jednom OpenBSD korisniku je bilo preo glave ovog besmislenog saobracaja, i postovao je neke delove njegovog pf.conf fajla u njegovom blogu:

```
altq on $ext_if cbq queue { q_default q_web q_mail }

   queue q_default cbq(default)
   queue q_web (...)

    ## all mail limited to 1Mb/sec
    queue q_mail bandwidth 1Mb { q_mail_windows }
    ## windows mail limited to 56Kb/sec
    queue q_mail_windows bandwidth 56Kb

    pass in quick proto tcp from any os "Windows" to $ext_if port 25 \
          keep state queue q_mail_windows
    pass in quick proto tcp from any to $ext_if port 25 label "smtp" \
          keep state queue q_mail
" i can't believe i didn't see this earlier. Oh, how sweet. ...
  Already a huge difference in my load. Bwa ha ha. "
```

Randal L. Schwartz, 29. Januar 2004, http://use.perl.org/~merlyn/journal/17094
Ovde je svom email saobracaju dodeljen jedan megabit protoka, dok email saobracaj koji potice sa Windows hostova dobija da deli podniz od ukupno 56 kbit. Bez iznenadjenja, ukupno opterecenje se snizilo i blog se zavrsava necim sto lici na zlo cerekanje.
Moram da priznam da je ovo nesto sto sam ja veoma zeleo da uradim, ali se nikada nisam usudio. Veliki broj nasih musterija je izabralo, iz nekih njihovih razloga, da podignu mail servis na Windows-u, i mi zelimo da primimo vecinu njihove poste. Ubrzo, pogledacemo drugaciji PF pristup koji ce postici skoro isti efekat.

CARP i pfsync

CARP i pfsync su bile dve glavne nove stvari u OpenBSD 3.5. CARP je skracenica za Redundantni Protokol Javnih Adresa (Common Address Redundancy Protocol). Protokol je razvijen kao patentom neopterecena alternativa za VRRP (Virtual Router Redundancy Protocol, RFC 2281, RFC 3768), koji je jos uvek bio daleko da postane IETF potvrdjeni standard, iako moguci problem sa patentom nije bio resen. Pomenuti patenti su drzani od strane Cisco, IBM i Nokia.
Oba protokola imaju nameru da osiguraju rendundantnost sustinskih mreznih karakteristika, sa automatskim oporavkom pada.
CARP se zasniva na podesavanju grupe masina kao jedan ‘master’ i jedan ili vise radundantnih ‘robova’, svi sa mogucnostima da se nose sa javnim IP adresama. Ako master padne, jedan od robova ce naslediti IP adresu, i ako je sinhronizacija odradjena pravilno, aktivne konekcije ce biti prenete. Prenos se moze autorizovati koristeci kriptografske kljuceve.
Jedna od glavnih namena CARP-a je da obezbedi da mreza i dalje funkcionise kao i obicno cak i u slucaju da zastitni zid ili drugi servis padne zbog gresaka ili planiranih aktivnosti odrzavanja kao na primer upgrejd.
U slucaju PF zastitnog zida, pfsync moze da obavlja sinhronizaciju. pfsync je tip virtualnog mreznog interfejsa posebno dizajniranog za sinhronizaciju informacija o stanju izmedju PF zastitnih zidova. pfsync interfejsi se dodeljuju fizickim interfejsima preko ifconfig. Na mrezama gde su uptime zahtevi

toliko striktni da diktiraju automastki oporavak, verovatno je da postoji nekoliko simultanih mreznih konekcija dovoljno velikih da bi bilo dobro da pfsync mrezi dodelimo svoju fizicku mrezu.
Nadam se da cu istraziti ove uzbudljive i napredne mogucnosti u buducem naprednom delu ovog tutorijala. Za sada, najbolje reference za pfsync i CARP su OpenBSD FAQ, man stranice i clanak Ryan McBridet-a http://www.countersiege.com/doc/pfsync-carp/.

Pojednostavljenje bezicne mreze

Zelim da kazem da u BSD-u, i posebno u OpenBSD-u, ne postoji potreba za ‘pojednostavljenjem bezicnog umrezavanja’, zato sto to vec jeste. Postavljanje bezicne mreze u sustini se ne rezlikuje od pokretanja jednog umrezenog, ali postoje i problemi koji se javljaju jednostavno zato sto se ovde radi o radio talasima a ne kablovima. Nakratko cemo pogledati neke od problema pre nego sto krenemo sa prakticnim problemima podesavanja.

Malo IEEE 802.11 istorije

Prelazak na bezicne mreze pruza vam mogucnost posmatranja bezbednosti na raznim nivoima u mreznom stack-u iz nove perspective. Kratko cemo pogledati dva osnovna IEEE 802.11 bezbednosna mehanizma.[12]
Ne treba ni reci da ce vam trebati dodatne sigurnosne mere, kao na primer SSH ili SSL enkripcija, da biste ocuvali odredjeni nivo privatnosti za vas protok podataka.

WEP (Wireless Equivalent Privacy)

Jedna od posledica koriscenja radio tralasa umesto zica je da je podjednako lako da se uhvate vasi podaci tokom prenosa. Dizajneri 802.11 familije bezicnih mreznih standarda su izgleda bili svesni ovog, i dosli su do resenja koje su izneli pod imenom *Wired Equivalent Privacy*, ili *WEP*.
WEP je enkricpiona shema nivoa koji se smatra primitivnom kucnom radinoscu medju kripstografskim profesionalcima. Nije bilo iznenadjujuce kada je WEP enkripcija probijena nekoliko meseci nakon sto je prvi proizvod pusten. Iako postoje besplatni alati koje mozete skinuti za razbijanje sifre WEP enkodiranog saobracaja bukvalno za nekoliko minuta, zbog nekoliko razloga jos uvek je veoma rasprostranjena i koriscena. Na saobracaj zasticen samo WEP-om trebate gledati kao na malo sigurniji u odnosu na prenos podataka bez ikakve zastite. Ali opet, samim tim da je trud potreban da bi se neko ubacio u WEP mrezu moze biti dovoljan da se obeshrabre lenji i nevesti napadaci.

WPA (WiFi Protected Access)

Dizajnerima 802.11 je vrlo brzo svanulo da ustvari njihov Wired Equivalent Privacy sistem nije bas ono sto je trebao da bude, tako da su dosli sa preradjenim i malo opseznijem resenjem koje je nazvano *WiFi Protected Access*, ili *WPA*.
WPA izgleda bolje nego WEP, barem na papiru, ali specifikacija je dovoljno komplikovanija da nije ni izbliza toliko rasprostranjena niti implementirana kao sto su to dizajneri hteli. i WPA je bio kritikovan zbog njegovih problema dizajna i bagova. U kombinaciji sa pozatim problemima pristupa

dokumentaciji i hardveru, besplatna podrska za softver varira. Ako specifikacija vaseg projekta ukljucuje WPA, pazljivo pogledajte dokumentaciju za vas operativni sistem i drajvere.

Podesavanje jednostavne bezicne mreze

Prvo proverite dali je vasa kartica podrzana i proverite izlazne podatke komandom dmesg da biste se uverili da se drajver pokrece i podize karticu pravilno[13] Sa uspesno konfigurisanom karticom trebalo bi da vidite nesto kao

```
ath0 at pci1 dev 4 function 0 "Atheros AR5212" rev 0x01: irq 11
ath0: AR5212 5.6 phy 4.1 rf5111 1.7 rf2111 2.3, ETSI1W, address
00:0d:88:c8:a7:c4
```

Sledece, trebate konfigurisati interfejs za TCP/IP. U OpenBSD-u, ovo znaci da /etc/hostname.ath0 otprilike ovako:

```
up media autoselect mediaopt hostap mode 11b chan 6 nwid unwiredbsd \
 nwkey 0x1deadbeef9
inet 10.168.103.1
```

Primetite da je konfiguracija podeljena u dve linije. Prva linija generise ifconfig komandu koja podesava interfejs sa pravilnim parametrima za fizicku bezicnu mrezu, druga komanda, koja se izvrsava samo ako se prva komanda zavrsi, postavlja IP adresu. Primetite da smo izricito odredili kanal, i omogucili slabu WEP enkripciju tako sto smo stavili nwkey parametar.
U FreeBSD i NetBSD verzijama trebate stavili sledece linije u vas /etc/rc.conf:

```
ifconfig_ath0="up media autoselect mediaopt hostap mode 11b chan 6 \
  nwid unwiredbsd nwkey 0x1deadbeef9"
ifconfig_ath0="inet 10.168.103.1"
```

Onda verovatno zelite da pokrenete dhcpd da daje adrese i ostale vazne mrezne informacije do klijenata. Vasim klijentima ce trebati /etc/hostname.ath0 konfiguracija

```
up media autoselect mode 11b chan 6 nwid unwiredbsd nwkey 0x1deadbeef9
dhcp
```

ili na FreeBSD-u i NetBSD-u,

```
ifconfig_ath0="up media autoselect mode 11b chan 6 nwid unwiredbsd \
  nwkey 0x1deadbeef9"
ifconfig_ath0="DHCP"
```

u /etc/rc.conf.

Pod pretpostavkom da vas gateway radi NAT, verovatno cete zeleti da postavite NAT i za bezicnu mrezu, praveci neke male izmene u vasem /etc/pf.conf:

```
air_if = "ath0"
i
nat on $ext_if from $air_if:network to any -> ($ext_if) static-port
```

Trebace vam slicna, skoro ista, linija za vasu ftp-proxy konfiguraciju, i ukljucite $air_if u vasim pass pravilima.
To je sve. Ova konfiguracija vam daje funkcionalnu BSD pristupnu tacku, sa barem simbolicnom bezbedboscu preko WEP enkripcije.

Otvorena, ali cvrsto cuvana bezicna mreza sa authpf

Kao i uvek, postoje i drugi nacini da se konfigurise bezbednost vase bezicne mreze u odnosu na ovaj koji smo upravo videli. Zastita koju WEP enkripcija nudi, a bezbednosni profesionalci se uglavnom slazu, je jedva dovoljna da se signalizira napadacu da ne zelite da dopustite svima i da na razliciti nacin koriste vase mrezne resorse.
Drugaciji pristup se pojavio jednog dana u mom emailu u formi poruke od mog prijatelja Vegard Engen-a, koji mi je rekao da je pripremao authpf. authpf je korisnicka komandna linija koja vam omogucava da ucitate PF pravila prema korisniku, efektivno odlucujuci koji korisnik dobija sta da radi. Sa umerenim podesavanjem, samo saobracaj koji potice od autorizovanih korisnika ce biti propusten. Beleske Vegard-vog podesavanja su dole navedene. Njegova bezicna mreza je konfigurisana bez WEP enkripcije, preferirajuci da prepusti bezbednost PF-u i authpf:
Pocnite tako sto cete kreirati prazni fajl /etc/authpf/authpf.conf. Treba se nalaziti tamo da bi authpf radio, ali ustvari ne treba da sadrzi nikakve vrednosti.
Drugi vazni deo fajla /etc/pf.conf sledi. Prvo, makroi za interfejse:

```
int_if="sis1"
ext_if="sis0"
wi_if = "wi0"
```

Koriscenje ove adrese ce biti ociglednije malo kasnije:

```
auth_web="192.168.27.20"
```

Tradicionalna authpf tabela

```
table <authpf_users> persist
```

Mozemo staviti NAT deo u authpf.rules, ali nista ne skodi da ga zadrzimo u glavnom fajlu pf.conf:

```
nat on $ext_if from $wi_if:network to any -> ($ext_if)
```

Preusmerite saobracaj da bi stigao do servera na unutrasnjoj mrezi. i ovo moze biti stavljeno u authpf.rules, ali posto ustvari ne omogucavaju pristup bez pass pravila, nece skoditi ako ih zadrzimo ovde.

```
rdr on $wi_if proto tcp from any to $myaddr port $tcp_in -> $server
rdr on $wi_if proto udp from any to $myaddr port $udp_in -> $server
```

Sledece preusmerenje salje sav web saobracaj od neautorizovanih korisnika na port 80 na $auth_web. U Vegard-ovom podesavanju, ovo je web server koji prikazuje kontakt informacije osoba koji su greskom na bezicnoj mrezi. U komercijalnom podesavanju, ovo bi bilo mesto gde bi stavili nesto sto bi upravljalo kreditnim karticama i kreiralo korisnike.

```
rdr on $wi_if proto tcp from ! <authpf_users> to any \
 port 80 -> $auth_web
```

Da biste aktivirali nat, binat ili preusmerenja u authpf

```
nat-anchor "authpf/*"
binat-anchor "authpf/*"
rdr-anchor "authpf/*"
```

Vracamo se na filter pravila, pocecemo sa pametnom osnovom

```
block all
```

U globalu, nezavisna pravila korisnika ce ici ovde. Sledece za authpf anchor, postaracemo se da neautorizovani korisnici koji se konektuju na bezicni interfejs budu preusmereni na $auth_web

```
anchor "authpf/*" in on wi0

pass in on $wi_if inet proto tcp from any to $auth_web \
 port 80 keep state
```

Postoje tri stvari koje zelimo na nasem bezicnom interfejsu: Servis imena (DNS), DHCP i SSH sa prolaskom do gateway-a. Jednostavno resavamo koristeci tri pravila

```
pass in on $wi_if inet proto udp from any port 53 keep state

pass in on $wi_if inet proto udp from any to $wi_if port 67

pass in on $wi_if inet proto tcp from any to $wi_if \
 port 22 keep state
```

Sledece, definisemo pravila koja se ucitavaju za sve korisnike koji se uloguju preko njihove komandne linije podesene na /usr/sbin/authpf. Ova pravila idu u /etc/authpf/authpf.rules,

```
int_if = "sis1"
ext_if = "sis0"
wi_if = "wi0"
server = "192.168.27.15"
myaddr = "213.187.n.m"

# Services which live on the internal network
# and need to be accessible
tcp_services = "{ 22, 25, 53, 80, 110, 113, 995 }"
udp_services = "{ 53 }"
tcp_in = " { 22, 25, 53, 80, 993, 2317, pop3}"
udp_in = "{ 53 }"

# Pass traffic to elsewhere, that is the outside world
pass in on $wi_if inet from <authpf_users> to ! $int_if:network \
   keep state

# Let authenticated users use services on
# the internal network.

pass in on $wi_if inet proto tcp from <authpf_users> to $server \
  port $tcp_in keep state
```

```
pass in on $wi_if inet proto udp from <authpf_users> to $server \
  port $udp_in keep state

# Also pass to external address. This means you can access
# internal services on external addesses.

pass in on $wi_if inet proto tcp from <authpf_users> to $myaddr \
    port $tcp_in keep state
pass in on $wi_if inet proto udp from <authpf_users> to $myaddr \
    port $udp_in keep state
```

Sada imamo otvorenu mrezu gde se svi mogu konektovati i dobiti adresu od DHCP. Svi HTTP zahtevi se presusmeravaju na port 80 na 192.168.27.20, sto je ustvari server na unutrasnjoj mrezi gde se svim zahtevima odgovara sa istim stranicom, koja prikazuje kontakt informacije u slucaju da se zelite registrovati i da vam se dozvoli koriscenje mreze.
Dozvoljeno vam je da se poveze preko ssh do vaseg gateway-a. Korisnici sa validnim korisnickim ID-jem i lozinkom dobijaju pravila sa odgovarajucim pass pravilima ucitanim za njihovu dodeljenu IP adresu.
Mozemo ovo jos finije podesavati tako sto cemo napraviti specificna pravila za korisnike u /etc/authpf/users/$user/authpf.rules. Pravila za svakog korisnika mogu koristiti $user_ip makro za korisnikovu IP adresu. Na primer, ako zelim da sebi dam neogranicen pristup, kreiracu sledece

/etc/authpf/users/vegard/authpf.rules:

```
wi_if="wi0"
pass in on $wi_if from $user_ip to any keep state
```

Odbijanje brute force napada

Ako ste koristitili Secure Shell login servis bilo gde sa pristupom Internetu, verovatno ste imali prilike da vidite ovakve stvari u vasem logovima za autorizaciju:

```
Sep 26 03:12:34 skapet sshd[25771]: Failed password for root from
200.72.41.31 port 40992 ssh2
Sep 26 03:12:34 skapet sshd[5279]: Failed password for root from
200.72.41.31 port 40992 ssh2
Sep 26 03:12:35 skapet sshd[5279]: Received disconnect from
200.72.41.31: 11: Bye Bye
Sep 26 03:12:44 skapet sshd[29635]: Invalid user admin from
200.72.41.31
Sep 26 03:12:44 skapet sshd[24703]: input_userauth_request:
invalid user admin
Sep 26 03:12:44 skapet sshd[24703]: Failed password for invalid user
admin from 200.72.41.31 port 41484 ssh2
Sep 26 03:12:44 skapet sshd[29635]: Failed password for invalid user
admin from 200.72.41.31 port 41484 ssh2
Sep 26 03:12:45 skapet sshd[24703]: Connection closed by 200.72.41.31
Sep 26 03:13:10 skapet sshd[11459]: Failed password for root from
200.72.41.31 port 43344 ssh2
Sep 26 03:13:10 skapet sshd[7635]: Failed password for root from
200.72.41.31 port 43344 ssh2
Sep 26 03:13:10 skapet sshd[11459]: Received disconnect from
200.72.41.31: 11: Bye Bye
Sep 26 03:13:15 skapet sshd[31357]: Invalid user admin from 200.72.41.31
```

```
Sep 26 03:13:15 skapet sshd[10543]: input_userauth_request: invalid
user admin
Sep 26 03:13:15 skapet sshd[10543]: Failed password for invalid user
admin from 200.72.41.31 port 43811 ssh2
Sep 26 03:13:15 skapet sshd[31357]: Failed password for invalid user
admin from 200.72.41.31 port 43811 ssh2
Sep 26 03:13:15 skapet sshd[10543]: Received disconnect from
200.72.41.31: 11: Bye Bye
Sep 26 03:13:25 skapet sshd[6526]: Connection closed by 200.72.41.31
```

Nakon toga se ponavlja. Ovako izgleda brute force napad. Zasigurno neko, ili verovatnije, napadacev kompjuter negde, pokusava da dodje do kombinacije korisnickog imena i lozinke pomocu brute force napada koji ce mu omoguciti ulaz u vas sistem.
Najjednostavnija reakcija bi bila da se napise pf.conf pravilo koje blokira sve prilaze. Ovo nas dovodi do nove klase problema, ukljucujuci ono sto radite da bi dozvolili pristup osobama sa legitimnim poslom na vas sistem. Mozete razmisljati da pomerite servis na neki drugi port, ali opet, oni koji vas flood-uju na portu 22 verovatno mogu da skeniraju i do porta 22222.
Od OpenBSD 3.7, PF je ponudio malo elegantnije resenje. Mozete pisati vasa pass pravila tako da odrzavaju odredjene granice sta hostovi koji se konektuju mogu da urade. U dobroj meri, mozete staviti prekrsitelje u tabelu sa adresama kojima onda mozete zabraniti neki ili ceo pristup. Mozete cak i izabrati da ugasite sve postojece konekcije koje dolaze sa masina koje prelaze vasa ogranicenja, ako zelite. Evo kako se to radi:
Prvo kreirajte tabelu. U vasem delu za tabele dodajte

```
table <bruteforce> persist
```

Onda negde blize pocetku vaseg skupa pravila podesite da blokirate bruteforcere

```
block quick from <bruteforce>
```

i na kraju, vase pass pravilo.

```
pass inet proto tcp from any to $int_if:network port $tcp_services \
        flags S/SA keep state \
                (max-src-conn 100, max-src-conn-rate 15/5, \
         overload <bruteforce> flush global)
```

Ovo je veoma slicno onome sto smo vec videli zar ne? Ustvari, prvi deo je indentican delu koji smo sastavili ranije. Deo u zagradama je nova stvar koja ce jos vise rasteretiti vase mrezno opterecenje.
max-src-conn oznacava broj simultanih konekcija koje su dozvoljene od jednog hosta. U ovom primeru, ja sam ga podesio na 100, u vasim podesavanjima mozda cete zeleti malo vecu ili nizu vrednost.
max-src-conn-rate predstavlja odnost novih konekcija koje su dozvoljene za jednog hosta, ovde 15 konekcija za 5 sekundi. Opet, vi odlucujete sta odgovara vasim podesavanjima.
overload <bruteforce> znaci da se adresa svakog hosta koji premasuje ove granice dodaje tabeli bruteforce. Nas skup pravila blokira sav saobracaj od adresa u bruteforce tabeli.
konacno, *flush global* kaze da kada host dostigne granicu, konekcija tog hosta se prekida (flushed). Opsti deo kaze da se u dobroj meri ovo odnosi na konekcije koje se poklapaju i sa ostalim pass pravilima.
Efekat je dramatican. Bruteforcers koji mene skeniraju veoma cesto zavrsavaju sa "Fatal: prekid pre autentifikacije" porukama, sto je upravo ono sto mi zelimo.
Jos jednom, imajte na umu da je ovaj primer pravila prvenstveno sluzi kao ilustracija. Veoma je moguce da ce vasoj mrezi trebati drugacija pravila ili kombinacija pravila.

Ako, na primer, zalite da dozvolite dobar broj konekcija uopste, ali zelite da budete malo stegnutiji kada se radi o ssh, mozete dodati pravilo iznad sa necim sto smo koristili ranije u nasem skupu pravila:

```
pass quick proto { tcp, udp } from any to any port ssh \
        flags S/SA keep state \
        (max-src-conn 15, max-src-conn-rate 5/3, \
        overload <bruteforce> flush global)
```

Trebalo bi da mozete da nadjete skup parametara koja su odgovarajuci za vasu situaciju tako sto cete procitati odgovarajuce man stranice i PF Prirucnik, i mozda malo eksperimentisanja.

Otezajte spamerima

Do sada smo presli dosta gradiva, i veoma sam srecan da vam mogu predstaviti nesto veoma korisno: PF kao sredstvo da otezate zivot spamera. Na osnovu naseg skorasnjeg izlaganja PF pravila, trebalo bi da odmah shvatite sledece /etc/pf.conf delove:

```
table <spamd> persist
table <spamd-white> persist file "/var/mail/whitelist.txt"
rdr pass on $ext_if inet proto tcp from <spamd> to \
        { $ext_if, $int_if:network } port smtp -> 127.0.0.1 port 8025
rdr pass on $ext_if inet proto tcp from !<spamd-white> to \
        { $ext_if, $int_if:network } port smtp -> 127.0.0.1 port 8025
```

Imamo dve tabele, i jedna od njih se puni od fajla negde na sistemu. SMTP saobracaj od adresa iz prve tabele plus one koje nisu u drugoj tabeli su preusmerene na demona koji slusa na portu 8025.

Zapamtite, niste sami: crna lista

Glavna stvar osnovnog spamd dizajna je cinjenica da spameri salju veliki broj poruka, i verovatnoca da ste vi prva osoba koja prima odredjenu poruku je veoma mala. Spam se pretezno salje preko nekoliko udruzenih spamerskim mreza i velikog broja otetih masina. i individualne poruke i masine ce veoma brzo biti prijavljene na crnu listu, i ovo su podaci koji na kraju zavrsavaju u prvoj tabeli u nasem primeru.

Lista crnog i sivog, i lepljivi tarpit

Ono sto spamd radi SMTP konekcijama koje dolaze od adresa u crnoj listi je da prezentuje svoj baner i odmah prelazi u mod gde odgovara na SMTP saobracaj jedan po jedan bajt. Ova tehnika, cija je namera da izgubi sto je moguce vise vremena onome koji salje poruke a prakticno ne kosta nista onog koji prima poruke, se naziva *tarpitting*. Specificna implementacija sa SMTP odgovorima od 1 bajta se cesto naziva *stuttering*.
spamd podrzava i *greylisting*, koji radi tako sto odbija poruke od nepoznatih hostova privremeno sa 45n kodovima, propustajuci poruke sa hostova koji pokusaju ponovo za neko prihvatljivo vreme.

Saobracaj sa hostova koji se dobro ponasaju, to jest, posiljaoci koji su podeseni da se ponasaju u granicama postavljenim u odgovarajucim RFCs[14], ce biti propusten.
Siva lista kao tehnika je predstavljena 2003 u clanku Evan Harris-a[15], i implementacije su usledilie tokom narednih nekoliko meseci. spamd OpenBSD-a je dobio podrsku za sivu listu sa verzijom OpenBSD 3.5, koja je izasla u Maju 2004.
Zapanjujuca stvar koja se tice sive liste, osim svoje jednostavnosti, je da jos uvek radi. Spameri i pisci malware-a se sporo adaptiraju. Videcemo nekoliko primera kasnije.

Podesavanje spamd

Konfigurisanje spamd je veoma jednostavno[16]. Jednostavno editujte /etc/spamd.conf prema vasim potrebama. Sam fajl daje dosta objasnjenja, a man stranica daje dodatne informacije. Preporuceni default stavlja na crnu listu dosta toga, ukljucujuci Korejske adrese. Ja radim u kompaniji koja je ustvari u saradnji sa Korejom, i kao posledicu imam to da sam morao da izbacim te odgovarajuce unose u nasoj konfiguraciji. Vi odlucujete koje cete izvore da koristite, i koriscenje drugih lista u odnosu na osnovne je moguce.
Stavite linije za spamd i pocetne parametre koje zelite u vas /etc/rc.conf ili /etc/rc.conf.local. Za primer da kazemo da mozete dalje podesiti nekoliko parametara koji se ticu sive liste preko spamd parametara u komandnoj liniji.
Kada ste zavrsili editovanje podesavanja, pokrecete spamd sa vasim opcijama, i kompletirate konfiguraciju koristeci spamd-setup. Na kraju, kreirajte cron job koji update-uje tabele za neki odgovarajuci interval vremena.
Jednom kada se tabele napune, mozete izlistati i manipulisati njihovim sadrzajem koristeci pfctl, kao i za sve druge tabele.
Primetite da ovaj primer koristi rdr pravila koja su isto pass pravila. Ako vasa rdr pravila ne sadrze 'pass' deo, moracete kreirati pass pravila da biste omogucili prolaz saobracaju do preusmerenja. Trebate i kreirati pravila da propustite legitimni email. Ako vec imate email servis u vasoj mrezi, verovatno mozete koristiti vasa stara SMTP pass pravila.

Nase iskustvo sa spamd

Kakav je spamd u prakticnoj upotrebi? Ozbiljnije smo poceli korsititi spamd pocetkom Decembra 2004, nakon sto smo vec imali spamassassin i clamav kao delove exim procesa isporuke neko vreme. Nas exim je konfigurisan da oznacava i dostavlja poruke sa spamassassin rezultatom u intervalu od 5 do 9.99 poena, dok odbacuje poruke sa 10 i vise poena kao iporuke koje sadrze malware. Dok se prolece priblizavalo, uspesnost spamassassin-a je bio u opadanju, propustajuci sve vise spama.
Kada smo ukljucili spamd, ukupan broj upravljanih poruka i broj poruka kojima je spamassassin manipulisao su se drasticno smanjile. Broj spam poruka koje polaze neobelezene se sada stabilizirao na oko pet poruka dnevno, na osnovu korisnih izvestaja korisnika.
Ako pokrenete spamd sa -v opcijom u komandnoj liniji za verbose logiranje, log se pokrece sa jos nekoliko informacija pored informacija o IP adresi. Sa verbose logiranjem, tipican izvod loga izgleda ovako:

```
Oct  2 19:55:05 delilah spamd[26905]: (GREY) 83.23.213.115:
<gilbert@keyholes.net> -> <wkitp98zpu.fsf@datadok.no>
Oct  2 19:55:05 delilah spamd[26905]: 83.23.213.115: disconnected after
```

```
0 seconds.
Oct  2 19:55:05 delilah spamd[26905]: 83.23.213.115: connected (2/1)
Oct  2 19:55:06 delilah spamd[26905]: (GREY) 83.23.213.115:
<gilbert@keyholes.net> -> <wkitp98zpu.fsf@datadok.no>
Oct  2 19:55:06 delilah spamd[26905]: 83.23.213.115: disconnected after
1 seconds.
Oct  2 19:57:07 delilah spamd[26905]: (BLACK) 65.210.185.131:
 <bounce-3C7E40A4B3@branch15.summer-bargainz.com> -> <adm@dataped.no>
Oct  2 19:58:50 delilah spamd[26905]: 65.210.185.131: From: Auto
Insurance Savings <noreply@branch15.summer-bargainz.com>
Oct  2 19:58:50 delilah spamd[26905]: 65.210.185.131: Subject: Start
SAVING M0NEY on Auto Insurance
Oct  2 19:58:50 delilah spamd[26905]: 65.210.185.131: To: adm@dataped.no
Oct  2 20:00:05 delilah spamd[26905]: 65.210.185.131: disconnected after
404 seconds. lists: spews1
Oct  2 20:03:48 delilah spamd[26905]: 222.240.6.118: connected (1/0)
Oct  2 20:03:48 delilah spamd[26905]: 222.240.6.118: disconnected after
0 seconds.
Oct  2 20:06:51 delilah spamd[26905]: 24.71.110.10: connected (1/1),
lists: spews1
Oct  2 20:07:00 delilah spamd[26905]: 221.196.37.249: connected (2/1)
Oct  2 20:07:00 delilah spamd[26905]: 221.196.37.249: disconnected after
0 seconds.
Oct  2 20:07:12 delilah spamd[26905]: 24.71.110.10: disconnected after
21 seconds. lists: spews1
```

Prve tri linije kazu da se masina konektuje, kao druga aktivna konekcija, sa jednom konekcijom od hosta sa crne liste. (GREY) i (BLACK) pre adresa indicira status sive i crne liste. Nakon 404 sekunde (ili 6 minuta, 44 sekunde), host sa crne liste se diskonektuje bez zavrsetka dostave. Sledece linije mogu predsravljati mozda i prvi kontakt sa masine, koja je nakon toga ubacena u sivu listu.
Nas prvi zakljucak je da spamd zaustavlja veoma dosta spama. Nazalost, videli smo i nekoliko laznih potvrda. Po meni, ovo su sve bile konekcije sa upisima u spews2 (spews level 2) listi. Prestali smo sa koriscenjem ove liste kao crne liste, bez nekog primetnog povecanja u obimu spama.
A sada ono sto je bilo vrhunac mog iskustva sa spamd. Jedan upis u logu se dugo izdvajao:

```
Dec 11 23:57:24 delilah spamd[32048]: 69.6.40.26: connected (1/1),
lists: spamhaus spews1 spews2
Dec 12 00:30:08 delilah spamd[32048]: 69.6.40.26: disconnected
after 1964 seconds. lists: spamhaus spews1 spews2
```

Ovaj upis se tice posiljaoca na wholesalebandwidth.com. Ova masina je imala 13 pokusaja dostave za period od 9-tog do 12-tog Decembra 2004. Zadnji pokusaj je trajao 32 minuta, 44 sekunde, bez zavrsetka dostave.
S vremena na vrema obnavljam ovaj tutorijal, i skoro sam nasao jos nekoliko novih upisa koje su ovo prekoracile:

```
peter@delilah:~$ grep disconnected /var/log/spamd | awk '{print $9}' \
| sort -rn | uniq -c | head
   1 36099
   1 14714
   1 5495
   1 2193
   1 1964
   1 1872
   1 1718
   1 1677
   1 1617
```

1 1594
Prvi, 36099 sekundi, sto je vise od deset sati,
Aug 26 10:10:17 delilah spamd[26905]: 146.151.48.74: connected (1/0)
Aug 26 20:11:56 delilah spamd[26905]: 146.151.48.74: disconnected
after 36099 seconds.

se isgleda tice masine na University of Wisconsin, Madison kamp, koji je verovatno bio inficiran izuzetno naivnim crvom koji salje spam, koji je cekao dugo vremena na ostatak SMTP dijaloga. Sledeci upisi, 4 sati, 5 minuta i 14 sekundi i 1 sat, 31 minuta i 35 sekundi, su se izgleda ponasali na slican nacin.

Zakljucak iz naseg iskustva sa spamd

Sumirajuci, ako se selektivno koristi, crne liste u kombinaciji sa spamd su mocne, precizne i efikasne alatke za borbu protiv spama. Uticaj na opterecenje spamd masine je minimalan. Sa druge strane, spamd nikada nece raditi bolje nego njegov najslabiji izvor podataka, sto znaci da trebate da posmatrate vase logove i da koristite bele liste kada je to potrebno.

PF - Haiku

Na kraju, samo mali prikaz koliko osecanja mogu imati korisnici koji su inspirisani PF-om. Na PF mailing listi, poruka sa naslovom “Stvari koje pf nemoze da uradi?” se pojavila u Maju 2004. Poruka je napisana od nekog ko nije imao dosta iskustva sa zastitnim zidovima, i ko ko je imao problema da dodje do podesavanja koje je on zeleo.
Ovo je, naravno, pokrenulo dosta diskusije, sa nekoliko ucesnika koji su rekli da je PF tezak za pocetnike, ali ni alternative sigurno nisu bile bolje. Tema se zavrsila sa sledecim pohvalnim haikuom od strane Jason Dixon-a, koja je data u netaknutoj formi zajedno sa Jason-ovim komentarima:

Compared to working with iptables, PF is like this haiku:

A breath of fresh air,
floating on white rose petals,
eating strawberries.

Now i'm getting carried away:

Hartmeier codes now,
Henning knows not why it fails,
fails only for n00b.

Tables load my lists,
tarpit for the asshole spammer,
death to his mail store.

CARP due to Cisco,
redundant blessed packets,
licensed free for me.

Jason Dixon, na PF email listi, 20-tog Maja, 2004 (http://www.benzedrine.cx/pf/msg04702.html)

Reference

OpenBSDs web http://www.openbsd.org/
OpenBSDs FAQ, http://www.openbsd.org/faq/index.html
PF User Guide http://www.openbsd.org/faq/pf/index.html
Daniel Hartmeier's PF pages, http://www.benzedrine.cx/pf.html
Daniel Hartmeier: Design and Performance of the OpenBSD Stateful Packet Filter (pf), http://www.benzedrine.cx/pf-paper.html (presented at Usenix 2002)
Nate Underwood: HOWTO: Transparent Packet Filtering with OpenBSD, http://ezine.daemonnews.org/200207/transpfobsd.html
Randal L. Schwartz: Monitoring Net Traffic with OpenBSD's Packet Filter, http://www.samag.com/documents/s=9053/sam0403j/0403j.htm
Unix.se: Brandvägg med OpenBSD, http://unix.se/Brandv%E4gg_med_OpenBSD
Randal L. Schwartz: Blog for Thu, Jan 29, 2004, http://use.perl.org/~merlyn/journal/17094
RFC 1631, "The IP Network Address Translator (NAT)", May 1994 http://www.ietf.org/rfc/rfc1631.txt?number=1631
RFC 1918, "Address Allocation for Private Internets", February 1996 http://www.ietf.org/rfc/rfc1918.txt?number=1918
The FreeBSD PF home page, http://pf4freebsd.love2party.net/
Peter Postma's PF on NetBSD pages, http://nedbsd.nl/~ppostma/pf/
Marcus Ranum: The Six Dumbest Ideas in Computer Security, September 1, 2005
Kjell Jørgen Hole WiFi courseware, http://www.kjhole.com/Standards/WiFi/WiFiDownloads.html, also see wifinetnews.com; also The Unofficial 802.11 Security Web Page comes higly recommended.
Greylisting.org greylisting.org
Evan Harris: The Next Step in the Spam Control War: Greylisting (the original greylisting paper)

Gde naci tutorijal na web-u

Rad u toku na:
Nekoliko formata http://www.bgnett.no/~peter/pf/
Ovo je mesto gde ce se updejtovane verzije pojavljivati, izmedju dogadjanja. Molim vas obavestite me ako zelite da budete obavesteni o buducim izmenama.

Ako ste uzivali u ovome: Kupite OpenBSD CD-ove i ostale stvari, donirajte!

Ako ste uzivali u ovom tutorijalu ili vam je bio od koristi, molim vas idite na OpenBSD.org Stranica za narudzbine i kupite komplete CD-ova, ili u tom duhu, potpomognite dalji razvojni rad OpenBSD projekta preko donacije.
Ako ste ovo slusali na predavanju, mozda postoji OpenBSD prodavnica u blizini gde mozete kupiti CD-ove, majce ili ostale predmete.
Zapamtite, cak i slobodnom softveru treba pravi rad i pravi novac da bi se razvijao i odrzavao.

Beleske

[1] Ovaj rukopis je malo izmenjena verzija rukopisa pripremljenog za predavanje koje je najavljeno kao (prevedeno sa Norveskog): "Ovo predavanje je o zastitnim zidovima i srodnim funkcijama, sa primerima iz realnog zivota sa PF (Paket Filterom) iz OpenBSD projekta. PF nudi zastitni zid, NAT, kontrolu saobracaja i upravljanje protokom u jednom, felskibilnom i za sysadmina lakom sistemu. Peter se nada d ace vam ovo predavane dati neke ideje o tome kako da kontrolisete vas mrezni saobracaj na nacin na koji zelite – zadrzavajuci neke stvari izvan vase mreze, prosledjivanje saobracaja do specificnih hostova ili servisa, i naravno, otezati spamerima."

Osobe koje su dale znacajnu i korisnu ideje za ovaj rukopis su Eystein Roll Aarseth, David Snyder, Peter Postma, Henrik Kramshøj, Vegard Engen, Greg Lehey, Ian Darwin, Daniel Hartmeier, Mark Uemura, Hallvor Engen i verovatno nekoliko onih koji su se izgubili u mojoj mail arhivi dok ih ne nadjem i grep-ujem odatle.

[2] Dva dokumenta su RFC 1631, "The IP Network Address Translator (NAT)", koji datira od Maja 1994 i RFC 1918, "Address Allocation for Private Internets", koji datira od Februara 1996. Pogledajte *Poglavlje Reference*

[3] Uvek pokusavam da koristim sudo kada treba da uradim neto sto zahteva privilegije. Sudo je u osnovnim sistemu na OpenBSD-u, i lako je dostupan kao port ili paket. Ako jos uvek niste poceli da koristite sudo, trebalo bi da pocnete. Tako cete izbeci da sami sebi pucate u nogu samo zato sto ste zaboravili da ste root u tom terminalu.

[4] Zbog podesnosti - pfctl moze da upravlja sa nekoliko operacijama u jednoj komandnoj liniji. Mozete, na primer, da omogucite PF i da ucitate pravila komandom `sudo pfctl -e -f /etc/pf.conf`

[5] Postoje neke razlike izmedju FreeBSD 4.n i 5.n i novijih izdanja u vezi PF-a. Obratite se FreeBSD Handbook-u, posebno PF poglavlju da vidite koje informacije vaze u vasem slucaju.

[6] Mozete se zapitati zasto pisem pravila sa pocetnim deny? Kratak odgovor je, daje vam bolju kontrolu uz malo razmisljanja. Cilj packet filteringa je da preuzmete kontrolu, a ne da jurite za onim sta losi momci rade. Marcus Ranum je napisao veoma zabavan i informativan clanak o ovome, Sest najglupljih ideja u kompjuterskoj sigurnosti, koji je dobro procitati.

[7] Za dialup korisnike i vecinu ADSL korisnika, posebno za one koji koriste PPP preko Etherneta (PPPoE), spoljni interfejs ce biti `tun0`, umesto Ethernet interfejsa.

[8] Glavni internet RFCs koji opisuju ICMP i slicne tehnike su RFC792, RFC950, RFC1191, RFC1256, RFC2521, rfc2765, dok se neophondi updejti za ICMP za IPv6 mogu naci u RFC1885, RFC2463, RFC2466. Ovi dokumenti su dostupni na vise mesta na mrezi, kao npr ietf.org i faqs.org web stranicama, i verovatno i preko vaseg paket sistema.

Veoma je moguce da cu se vratiti na ICMP filtering u buducem naprednijem delu ovog tutorijala.

[9] Pogledajte Redirection and Reflection u PF Prirucniku.

[10] Na FreeBSD-u, ALTQ su potrebne ALTQ i queue discipline opcije za discipline koje zelite da kompajlirate u tekucem kernelu. Pogledajte PF poglavlje u FreeBSD Handbook-u za dodatne informacije.

[11] Kada je ovo pisano, ALTQ nije bio integrisan u PF implementaciji u NetBSD-u. Peter Postma odrzava NetBSD patch da omoguci PF/ALTQ funkcije. Sveze informacije u vezi ovog su dostupne na Peter Postmas PF na NetBSD stranicama, http://nedbsd.nl/~ppostma/pf/

[12] Za kompletniji pregled problema koji se ticu bezbednosti u beziciom mrezama, pogledajte clanke i slajdove eg Professor Kjell Jørgen Hole-a na www.kjhole.com. Za sveze podatke o razvoju WiFI, preporuceni sajtovi su wifinetnews.com i The Unofficial 802.11 Security Web Page.

[13] Podrska za bezicnu mrezu u OpenBSD i BSD u globalu sve vise ide nabolje, ali to ne znaci da je lako sklopiti sve delove koji su vam potrebni. Kratka istorija moje kucne mreze isgleda ovako: Poceo sam tako sto sam kupio dve CNet CWP-854 kartice, koje bi trebalo da su podrzane u

OpenBSD 3.7 preko novog ral drajvera. Jednu sam stavio u novu novcatu Dell masinu koja koristi neslobodan operativni system koji dolazi snjim. Moj gateway, koji je bez problema radio jos od verzije 3.3, je bio malo problematicniji. Kartica je prepoznata i konfigurisana, ali kada je Dell pokusao da dobije IP adresu, gateway se srusio sa kernel panicom. Detalji su dostupni kao OpenBSD PR number 4217. Obecao sam da cu testirati karticu ponovo sa novom verzijom snapshot-a – cim budem nasao karticu ponovo. Od Dell-a mogli smo videti izuzetan broj mreza, skoro sve neosigurane, ali to je potpuno druga prica..

Odlucio sam da probam ath kartice, i kupio sam D-Link DWL-G520, kja se negde zagubila kada sam se selio. Sledece, kupio sam DWL-G520+, mizleci da znak plus + znaci da je bolja. Nazalost, plus je znacio da je koriscen sasvim drugi cipset, TI ACX111, koji dolazi sa niskom cenom ali bez dostupne dokumentacije za developere slobodnog softvera. Srecom prodavnica mi je dozvolila da vratim karticu bez problema. Tada sam vec postajao isfrustriran i presao ceo grad do prodavnice koja je imala nekoliko DWL-AG520 kartica na lageru. Bile su malo skuplje u odnosu na ostale kartice,ali su odmah proradile.Nekoliko Nedelja kasnije pojavio se G520 i naravno i ona je radila. Moj laptop (koji radi na FreeBSD-u) je dosao sa Realtek 8180 wireless mini-PCI karticom, ali iz nekog razloga nisam je mogao naterati da radi. Zavrsio sam kupujuci DWL-AG650 cardbus karticu, koja je bez problema radial sa ath drajverom. Moj savet je, ako kupujete online, drzite man stranice u drugom prozoru, a ako idete u prodavnicu, kazite osoblju da cete koristiti BSD, i ako niste sigurni u vezi kartice koju pokusavaju da vam prodaju, pozajmite neku masinu da biste posetili man stranice. Ako odmah kazete osoblju mozda ce biti lakes da vam kasnije vrate novac u slucaju da karica ne radi.

[14] Bitne RFCs su uglavnom RFC1123 i RFC2821.

[15] Originalan Harris-ov papir i drugi korisni clanci i resorsi se mogu naci na greylisting.org web stranici.

[16] Na FreeBSD-u, spamd je port, /usr/ports/mail/spamd/. Ako koristite PF na FreeBSD 5.n ili novijem, morate instalirati port, pratite uputstva i vratite se ovde.