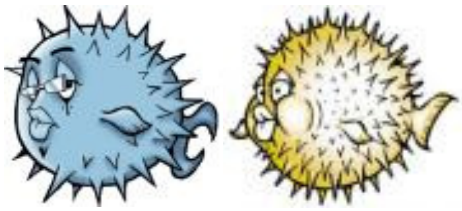
OPENBSD - PAKET SÜZGECİ
PF - PACKET FILTER
V1_1.0



TÜRKÇE KULLANMA KLAVUZU

BÖLÜM -1-

TEMEL AYARLAR

Yusuf UZUNAY
Orta Doğu Teknik Üniversitesi
Enformatik Enstitüsü
Ankara TÜRKİYE
yuzunay(at)ii.metu.edu.tr

Cahit GÜNGÖR
Hacettepe Üniversitesi
Bilgisayar Mühendisliği Bölümü
Ankara TÜRKİYE
b20122177(at)cs.hacettepe.edu.tr

Son Güncelleme: 10.05.2005

İÇİNDEKİLER

1. BİLGİLENDİRME

Bu dökümanın hemen hemen tamamı OpenBSD sitesindeki PF Kullanıcı Rehberi (http://www.openbsd.org/faq/pf/index.html) dökümanından çevrilmiştir. Amacımız bir çok yerde en güvenli işletim sistemi olarak telaffuz edilen OpenBSD'nin varsayılan olarak sunduğu PF isimli ateş duvarını tanıtıcı bir Türkçe döküman oluşturup, Türk bilişimine katkıda bulunmaktır. Çeviri esnasında birebir çeviri bazı durumlarda çok zor olmaktadır. Bu yüzden zorluk yaşanan yerlerde bire bir çeviriden ziyade Türkçe'de daha açıklayıcı olacak anlatımlar tercih edilmiş, bazı noktalarda da ek açıklamalarla konu pekiştirilmeye çalışılmıştır. Dökümanda gerek çeviri yönünden, gerekse başka hususlarda hatalar bulunabilir. Bu dökümanın olgunlaştırılmasında herkesin katkısı olabilir. Tespit ettiğiniz hatalar olursa veya şu bölümde şu şekilde bir anlatım yapılsaydı Türkçe olarak daha iyi anlaşılırdı veya şunu da ekleyelim gibi önerilerinizi bize e-posta yoluyla iletirseniz, dökümanın bir sonraki versiyonunda görüşleriniz değerlendirilecektir.

Dökümanı orijinalindeki gibi 3 bölümde yayınlamayı düşündük. Bu bölümler şunlardır:

- Temel Ayarlar
- İleri Düzey Ayarlar
- PF ile İlgili Ek Konular (Performans, kayıt alma v.b.)

Bu versiyonda sadece Temel Ayarlar ile ilgili olan bölüm yer almaktadır. 2. Bölümün çalışmaları hala devam etmektedir.

Yine çeviri esnasında zorlandığımız bir nokta da, bilişimde bazı kelimelerin İngilizce'sinin Türkçe'sinden daha yoğun kullanılması ve bu yüzden de aslında tam olarak Türkçe karşılıklarının bulunamamasıdır. Döküman okunurken sıkıntı yaratmaması açısından, bu tip çevirdiğimiz bazı kelimeleri aşağıda veriyoruz:

Block : Engellemek
Conditioning : İyileştirme
Defragment : Birleştirme
Deny : Reddetmek
Driver : Sürücü
Filter : Süzgeç, süzme
Firewall : Ateş Duvarı
Forwarding : Aktarma
Gateway : Geçit
Handshake : El sıkışma
Host : İstemci
Interface : Arayüz, arabirim
Kernel : Çekirdek
Normalizing : Normalleştirme
Port : Kapı
Prioritization : Önceliklendirme
Proxy : Vekil
Proxying : Vekalet
Redirection : Yönlendirme
Script : Betik
Server : Sunucu

(Not: "File" sözcüğünün Türkçe karşılığı aslında "kütük" olmasına rağmen, bunun yerine "dosya" kullanımın çok yaygın olması ve bir çok kişinin "kütük" sözcüğüne yabancı olması sebebiyle, belgede "dosya" kelimesi "file" kelimesinin karşılığı olarak kullanılmıştır.)

2. GİRİŞ

PF, TCP/IP trafiğini süzme ve ağ adres dönüşümü (NAT – Network Address Translation) sağlayan bir OpenBSD sistemidir. PF, İngilizce "packet filter" yani paket süzgeci ifadesinden gelmektedir ve bu döküman boyunca kısaca PF olarak kullanılacaktır. PF, aynı zamanda TCP/IP trafiği üzerinde normalleştirme ve iyileştirme de yapabilir, bant genişliğini kontrol eder ve paket önceliklendirmeyi de destekler. PF, OpenBSD 3.0'dan bu yana ana OpenBSD çekirdeğinin bir parçasıdır.

PF, ilk olarak Daniel Hartmeier tarafından geliştirilmiş olup, şu anda Daniel ile birlikte diğer OpenBSD takımı üyeleri de PF'in geliştirilmesine katkı sağlamaktadır.

PF'i aktif etmek ve başlangıçta ayar dosyasının okunmasını sağlamak için, `/etc/rc.conf.local` dosyasına
`pf=YES`
satırını eklemek gerekir. Bir sonraki açılışta PF aktif hale geçecektir.

Aynı zamanda `pfctl` komutu da kullanılarak PF aktif veya pasif hale getirilebilir:

Aktif:
```
#pfctl -e
```

Pasif:
```
#pfctl -d
```

Bu komutlar sadece, PF'i aktif veya pasif hale getirir, herhangi bir kural kümesini yüklemez. Kural kümesi PF'in aktif hale gelmesinden önce veya sonra ayrı olarak yüklenir.

PF, açılış esnasında varsayılan olarak kurallarını `rc` betikleri tarafından yüklenen `/etc/pf.conf` dosyasından okur. Bu dosya aslında `pfctl(8)` tarafından okunan, yüklenen ve `pf(4)`'e girişi yapılan basit bir metin dosyasıdır. PF'in bu dosyadan değil de başka bir dosyadan kuralları okuması ve yüklemesi sağlanabilir. İyi dizayn edilmiş birçok UNIX sisteminde olduğu gibi, PF de oldukça büyük oranda esneklik sağlamaktadır.

PF ayar dosyası (`pf.conf`) 7 bölümden oluşmaktadır.

1. **Makrolar:** IP adresleri ve arayüzler benzeri şeyleri tutan, kullanıcı tanımlı değişkenlerdir.
2. **Tablolar:** IP adres listelerini tutmaya yarayan yapıdır.
3. **Seçenekler:** PF'in nasıl çalıştığı kontrol eden çeşitli seçeneklerdir.
4. **Scrub:** Paketleri birleştirme (defragment) ve normalleştirme (normalization) için tekrar işleme alma
5. **Queueing:** Paket önceliklendirme ve bant genişliği kontrolünü sağlar
6. **Translation:** NAT'ı ve paket yönlendirmesini kontrol eder.
7. **Süzme Kuralları:** Paketler bir arayüzden geçerken seçici geçirgenlik ve engelleme sağlar.

Bu dosyadaki boş satırlar önemsenmeyeceği gibi, # ile başlayan satırlar da açıklama satırı olarak ele alınır.

3. KONTROL

PF işlemleri pfctl komutuyla kontrol edilir. Komutun en sık kullanılan seçenekleri şu şekildedir:

`# pfctl -f /etc/pf.conf`	`pf.conf` dosyasını yükler
`# pfctl -nf /etc/pf.conf`	dosyayı tarar (parse), fakat yüklemez.
`# pfctl -Nf /etc/pf.conf`	dosyadan sadece NAT kurallarını yükler.
`# pfctl -Rf /etc/pf.conf`	dosyadan sadece süzme kurallarını yükler.
`# pfctl -sb`	Geçerli NAT kurallarını gösterir.
`# pfctl -sr`	Geçerli süzme Kurallarını gösterir.
`# pfctl -ss`	Geçerli durum tablosunu gösterir.
`# pfctl -si`	Geçerli süzme durumunu ve sayaçları gösterir.
`# pfctl -sa`	Gösterebileceği her şeyi gösterir.

Bütün komutları görebilmek için `pfctl(8)`'in `man` sayfasına bakılabilir.

4. LİSTELER

Benzer kriterlere sahip kuralları tek bir kural altında toplamaya yarar (örneğin çoklu protokolleri, IP Adreslerini, kapı numaralarını v.b.). Listeler küme parantezleri {} ile tanımlanır. `pfctl`, kuralları tararken bir liste tanımı ile karşılaştığında, aslında öncelikle bu listeyi açar ve kuralları tek tek çalıştırır.

```
block in on $EXT_IF from 192.168.0.1 to any
block in on $EXT_IF from 192.168.0.7 to any
block in on $EXT_IF from 192.168.0.12 to any
```

Bu üç kural liste kullanılmak suretiyle, şu şekilde tek bir kural olarak belirtilebilmektedir:

```
block in on $EXT_IF from "{ 192.168.0.1, 192.168.0.7, 192.168.0.12
}" to any
```

Ayrıca tek tek IP yazmaktan ziyade, şu şekilde bir liste tanımı da yapılabilir:

```
SERVERS="{192.168.0.15, 192.168.0.16 }"
block in on $EXT_IF from $SERVERS  to any
```

Çoklu listeler tek bir kural içerisinde tanımlanabilmektedir ve ayrıca sadece süzme kuralları için geçerli değildir.

```
rdr on fxp0 proto tcp from ant to any port { 22 80 } -> 192.168.0.6
block out on fxp0 proto { tcp udp } from {192.168.0.1, 10.5.32.6 }
to any port { ssh telnet }
```

NOT: Listelerin öğeleri arasında virgül (,) kullanımı tercihe bağlıdır.

Listelerde sıkça yapılan bir yanlış şu şekilde bir negatif gösterimdir:

```
pass in on fxp0 from { 10.0.0.0/8, !10.1.2.3.}
```

Bunun yerine aşağıdaki gibi iki kural yazılması daha doğru olacaktır:

```
pass in on fxp0 from 10.0.0.0/8
pass in on fxp0 from !10.1.2.3
```

5. MAKROLAR

Makrolar, IP adresleri, kapı numaraları, arabirim adları gibi verileri tutabilen, kullanıcıların tanımlayabildiği değişkenlerdir. Makrolar PF kural kümesinin karmaşıklığını azaltır ve kural kümesinin daha kolay bir şekilde yönetilebilmesini sağlar.

Makro isimleri bir harf ile başlamalı ve harf, sayı veya alt çizgi ile devam etmelidir. Ayrıca Makro isimleri `pass, out, block` gibi rezerve edilmiş kelimelerden biri olamaz.

Ör:

```
EXT_IF="dc0"
pass in on $EXT_IF any
```

Makro tanımlandıktan sonra kullanılırken yukarıda da görüldüğü üzere önüne "`$`" işaretini alır ve ayrıca liste şeklinde de tanım yapılabilir:

```
Friends = "{ 192.168.1.1, 10.0.2.5, 192.168.43.53 }"
```

Makrolar tekrar tekrar tanımlanabilmektedir:

```
host1 = "192.168.1.1"
host2 = "192.168.1.2"
all_hosts = "{" $host1 $host2 "}"
```

6. TABLOLAR

Tablolar IP Adreslerini tutmak için kullanılır. Listelere göre daha az bellek ve işlemci zamanı tüketirler. Tablolar şu şekillerde kullanılmaktadır:

- Süzme, *Scrub*, NAT ve yönlendirmelerde kaynak veya hedef adres olarak.
- NAT kurallarında dönüşüm adresi olarak
- Yönlendirme kurallarındaki adreslerde
- `Route-to`, `reply-to` ve `dup-to` süzme kurallarında hedef adres olarak.

Tablolar, `pf.conf` dosyasında veya pfctl komutu kullanılarak oluşturulabilirler.

6.1 Tablo Tanımları

Tablolar, `pf.conf` dosyasında "`table`" kelimesiyle tanımlanırlar. Her bir tablo için aşağıdaki nitelikler belirtilebilir:

- `const` – Tablo bir kez yaratıldığında bir daha içeriği değiştirilemez. Bu şekilde bir tanım yapılmadığı takdirde sistem güvenlik seviyesi (*securelevel*) 2 veya daha yukarı bir seviyede çalışsa dahi `pfctl` herhangi bir anda tabloya ekleme veya tablodan silme gibi işlemleri yapabilir.
- `persist` – Herhangi bir kural gönderimde bulunmasa bile, çekirdeğin tabloyu bellekte tutmasını sağlar. Eğer bu nitelik olmazsa, çekirdek tabloya gönderimde bulunan son kural silindiği zaman, tabloyu otomatik olarak silecektir.

Örnek:

```
table <arkadaslar> { 192.0.2.0/24 }
table <rfc1918> const { 192.168.0.0/16, 172.16.0.0/12, 10.0.0.0/8 }
table <spammers> persisit
block in on fxp0 from { <rfc1918>, <spammers> } to any
pass in on fxp0 from <arkadaslar> to any
```

Adresler tablolarda aynı zamanda negatif tanımlayıcı ile de kullanılabilirler.

```
table <arkadaslar> { 192.0.2.0/24, !192.0.2.5 }
```

Bu tanıma göre, arkadaşlar tablosu `192.0.2.5` nolu adres dışındaki `192.0.2.0/24` ağında bulunan IP adreslerini kapsamaktadır.

Tablo isimleri örneklerden de anlaşılacağı üzere daima <> işaretleri arasında belirtilir.

PF'de tablolar, aynı zamanda dosyalar içindeki girdilerden de oluşturulabilir.

```
table <spammers> persist file "/etc/spammers"
block in on fxp0 from <spammers> to any
```

/etc/spammers dosyasının her bir satırı IP Adreslerinden veya CIDR gösteriminde belirtilmiş ağ bloklarından oluşabilir. Satırın önemsenmemesi için başına # yerleştirilebilir.

6.2 PFCTL Kullanarak Tabloları Düzenleme

Tablolar pfctl komutu kullanarak anında düzenlenebilir. Örneğin <spammers> tablosuna ekleme yapmak için şu komut kullanılabilir:

```
# pfctl -t spammers -T add 218.70.0.0/16
```

Eğer daha önceden spammers isminde bir tablo yoksa bu komut eklenmeden önce spammers isminde bir tablo oluşturulur.

Bir tablodaki adresleri listelemek için ise şu komut verilebilir:

```
# pfctl -t spammers -T show
```

Her tablo girdisi için istatistikleri görüntülemek için -T show ile birlikte -v parametresi kullanılabilir. Aynı zamanda tablodan bir adres silmek için ise aşağıdaki komutu verebiliriz:

```
# pfctl -t spammers -T delete 218.70.0.0/16
```

6.3 Adres Belirtme

Bilgisayarlar, IP Adreslerinin yanı sıra istemci isimleri ile de belirtilebilirler. İstemci ismi IP adresine dönüştürüldüğünde, çıkan IPv4 ve IPv6 adresleri, tabloya yerleştirilir. IP adresleri geçerli bir arabirim adı veya "self" kelimesiyle belirtilerek de tabloya yerleştirilebilirler. Bu, tablonun o arabirime veya makineye atanan bütün IP Adreslerini (loopback adresi dahil) kapsaması anlamına gelmektedir.

Tablolarda dikkat edilmesi gereken başka bir husus da 0.0.0.0/0 ve 0/0 gibi ifadelerin tabloda kullanılamamasıdır fakat makro tanımlamak suretiyle kullanılabilirler.

6.4 Adres Eşleştirme

Eğer tabloda birden fazla adres alanları mevcutsa, IP'ye en yakın olan IP alınarak kurala tabi tutulur:

```
table <arkadaslar> { 172.16.0.0/16, !172.16.1.0/24, 172.16.1.100 }
block in on dc0 all
pass in on dc0 from <arkadaslar> to any
```

Dolayısıyla bu kurallara göre paketlerin nasıl değerlendirileceğine göz atalım:

- 172.16.50.5 – en yakın eşleşmesi 172.16.0.0/16 olduğundan paket tabloyla eşleşir ve geçer.
- 172.16.1.25 – en yakın eşleşmesi !172.16.1.0 olduğundan ve burada negatif bir eşleşme olduğundan paket engellenir.
- 172.16.1.100 – bu doğrudan 172.16.1.100 ile eşleştiğinden, paket geçer.
- 10.1.4.55 – tablo ile eşleşmez ve engellenir.

7. PAKET SÜZME

Paket Süzme, paketler bir arabirimden geçerken seçici geçirgenlik ve engelleme sağlar. PF'in paketleri incelerken kullandığı kriter paketlerin 3. Katman (IPv4 ve IPv6) ve 4. Katman (TCP, UDP, ICMP ve ICMPv6) başlıklarıdır. En sık kullanılan kriterler kaynak ve hedef IP Adresi, kaynak ve hedef kapı numaraları ve protokoldür.

Süzme Kuralları, paketler kurallarla eşleştikten sonra engellenecek mi yoksa geçirilecek mi buna karar verir. Dosyadaki kuralların sırası çok önemlidir. PF ayar dosyası baştan sona doğru taranır ve "quick" kelimesi yazılmamışsa eğer, daima en son yazılı olan kural geçerlidir. Bütün kurallara **quick** kelimesi eklediğimiz takdirde ise sıralamada başta yazılan kurallar geçerli kılınacaktır.

```
pass in quick proto tcp from any to 10.1.2.3
block in all
```

Bu kural dizilimine göre, `10.1.2.3` adresine giden TCP paketleri dışında bütün paketler engellenecektir.

7.1 Kural Sözdizimi

Süzme Kurallarının söz dizimi genel olarak (basitleştirilmiş bir şekilde) şu şekildedir.

Action [direction] [log] [quick] [on interface] [af] [proto protocol] [from src_addr [port src_port]] [to dst_addr[port dst_port]] [flags tcp_flags] [state]

action (eylem) : İki çeşit eylem vardır. Bunlar ya "geç-`pass`" yada "engelle-`block`"dir. `Pass` eylemi paketi çekirdeğe işlenmesi için gönderirken, `block` komutu ise verilen engelleme politikasına bakarak engelleme sağlar. Normal davranışın üzerine yazan engelleme politikaları ya "`block drop`" ya da "`block return`"dür.

direction (yön) : Paketler bir arabirime ya girerler (`in`) ya da çıkarlar (`out`).

log (kayıt) : paketlerin `pflogd` tarafından işlem günlüğünün tutulacağını (`log`) belirtir. Eğer kural `keep state`, `modulate state` veya `synproxy state` seçeneklerini içeriyorsa, sadece durumu (`state`) gerçekleştiren paket kayıt edilir. Ne olursa olsun bütün paketleri kayıt etmek için "`log-all`" kullanılır.

quick (hızlı) : Eğer bir kuralda "`quick` " kelimesi geçiyorsa, o kural, dizinin son kuralı olarak algılanıp işleme koyulur.

interface (arabirim) : Paketin içinden geçtiği ağ arabiriminin veya grubunun ismidir. Grup arabirim ismine sayı eklenmemiş halidir. Böylelikle aynı marka olan bütün arabirimleri desteklemesi anlamına gelmektedir. Örneğin `ppp` veya `fxp` yazıldığında `ppp` veya `fxp` arayüzüne gelen bütün paketler ele alınacaktır.

af : Paketin bağlı olduğu adres ailesi ya inet (IPv4 için) ya da inet6 (IPv6 için)'dır.

protocol : Paketin 4. Katman protokolünü belirtir. Şunlardan birisi olabilir:

- tcp
- udp
- icmp

- icmp6
- `/etc/protocols` dosyasından geçerli bir protokol.
- 0 ile 255 arasında bir protokol numarası
- liste kullanarak belirtilen bir protokol kümesi

`src_addr, dst_addr :` IP başlığındaki kaynak ve hedef adresidir. Adresler şu şekillerde belirtilebilir:

- Tek bir IPv4 veya IPv6 adresi
- Bir CIDR ağ bloğu
- Kural kümesi yüklendiğinde tam yetkilendirilmiş domain ismi. Sonuç olarak, elde edilen IP Adresleri kurallarda yerine koyulacaktır.
- Bir Ağ arabiriminin ismi. Arabirime atanmış herhangi bir IP Adresi kurallarda yerine koyulacaktır.
- `/netmask` (Ör: /24) ile izlenen bir ağ arabiriminin ismi. Arabirimdeki her bir adres CIDR ağ bloğu oluşturabilmek için ağ maskesi ile birleştirilir.
- Parantez ( ) içinde belirtilmiş ağ arabiriminin ismi. Bu tanımlama belirtilen arabirimin IP Adresinin değişmesi durumunda PF'e ilgili arabirimle ilgili kuralı güncelleştirmesi gerektiğini bildirir. Bu tanımlama genellikle DHCP veya çevirmeli bağlantı üzerinden IP alan arabirimler için oldukça faydalıdır. IP adresi her değiştiğinde PF'in kurallarının tekrar yüklenmesine gerek kalmayacaktır.
- Aşağıda belirtilen kelimelerin sonunda bulunduğu ağ arabirim isimleri.
  - **`:network`** – CIDR ağ bloğunu temsil eder. (Ör: 192.168.0.0/24)
  - **`:broadcast`** – Ağ broadcast adresini temsil eder.
  - **`:peer`** – noktadan noktaya (point to point) bağlantıda bir noktanın IP adresini temsil eder.

Ayrı yeten "**`:0`**" belirteci bir arabirim isminin arkasına veya yukarıda belirtilen belirteçlerden sonra getirilirse, bu PF'in takma IP adresi kullanmayacağını gösterir.

`src_port, dst_port :`

4. Katman paket başlığındaki kaynak/hedef kapıyı belirtir. Kapılar şu şekillerde belirtilebilir:
   - 1 ile 65535 arasındaki bir sayı
   - `/etc/services` dosyasından geçerli bir servis ismi
   - listeler kullanılarak belirtilmiş kapı kümesi
   - belirli bir aralık:
     - != (eşit değil)
     - < (den küçük)
     - > (den büyük)
     - <= (den küçük veya eşit)
     - >< (aralık)
     - <> (ters aralık)

   Bu sembollerden son ikisi ikili (binary) operatörlerdir ve iki argüman alırlar.

     - **:** (Özel aralık)

Örnekler:

```
# 100'den küçük kapılar
pass in proto tcp from any port < 100

# Sadece 102 nolu kapı
block in proto tcp tcp to any port = 102
```

```
#1001-2000 arasındaki kapılar
pass out proto udp from any port 1000 >< 2000

# 53 dışındaki bütün kapılar
block out proto udp to any port != 53

#0-99 Arası ve 1001'den büyük kapılar
pass out proto tcp to any port 100 <> 1000
```

tcp_flags : "proto tcp" yi kullanırken, TCP başlığında aktif olması gereken bayrakları (flags) belirtir. Bayraklar şu şekilde belirtilir *flags check/mask* . Örneğin: *flags S/SA* ifadesi PF'e sadece S ve A (SYN ve ACK) bayraklarına bakmasını ve bunların içinden S bayrağı aktif olanları eşleştirmesini söyler.

state : Belirli bir kuralla eşleşen paketin durum bilgisinin tutulup tutulmayacağını belirtir.

- **keep state :** TCP, UDP ve ICMP için geçerlidir..
- **modulate state :** sadece TCP için geçerlidir. PF bu tanımla ile birlikte eşleşen kurallar için güçlü başlangıç sıra numarası (ISNs- Initial Sequence Numbers) oluşturacaktır.
- **synproxy state :** Sunucuları TCP SYN seli saldırılarına karşı korumaya yardım etmek için içeri doğru gelen TCP bağlantılarını belleğinde bir vekil (proxy) işlemine tabi tutar. (Burada "proxies" gibi bir kavram kullanılmıştır, bizim çevirimize göre burada yapılan işlem gelen bütün SYN paketlerini olduğu gibi hedefe iletmeyip, 3 yollu el sıkışmanın gereğini yerine getiren bağlantıları hedefe iletmektedir şeklindedir. Dolayısıyla SYN seli dediğimiz devamlı SYN paketi gönderilerek oluşturulan saldırıların da önüne geçilmiş olur.) synproxy state seçeneği hem keep state hem de modulate state seçeneklerini kapsar.

7.2 Varsayılan Olarak Reddet

Ateş duvarını oluştururken önerilen yaklaşım genellikle "varsayılan olarak reddet – default deny" yaklaşımıdır. Bu ilk olarak bütün her şeyi engelleyip daha sonradan belirli paket tiplerine izin verme anlamına gelmektedir. Bu tip bir yaklaşımda ilk kurallar şu şekildedir:

```
block in all
block out all
```

Bu kurallarla bütün arabirimlere hem gelen hem de giden bütün trafik engellenmiş olur.

7.3 Trafiğe İzin Verme

Yukarıdaki iki kural tanımlandıktan sonra geçmesini istediğimiz trafik açık bir şekilde belirtilir. Belirtilmeyen trafik varsayılan olarak reddedilir. Bu safha paketlerin Kaynak/Hedef IP adreslerinin, kapılarının ve kullandıkları protokollerin önem kazandığı safhadır. İzin verilecek kurallar tanımlanırken dikkat edilecek husus mümkün olduğunda kısıtlayıcı bir kural kümesi oluşturmaktır. Bu sayede sadece ve sadece hedeflenilen trafiğe izin verilmiş olur.

```
# dc0 arbirimindeki  yerel ağdan, 192.168.0.0/24,
# IP adresi 192.168.0.1 olan OpenBSD makinesine doğru gelen trafiği geçir.
# Aynı zamanda dc0 arabiriminden dışarıya doğru çıkan geri dönüş trafiğine
# de izin ver.
pass in  on dc0 from 192.168.0.0/24 to 192.168.0.1
pass out on dc0 from 192.168.0.1 to 192.168.0.0/24
```

```
# fxp0 arabiriminde OpenBSD üzerinde çalışan Web Sunucuya doğru gelen
#paketleri geçir. Fxp0 arabirimi hedef adres olarak kullanılmıştır
#dolayısıyla sadece hedefi OpenBSD makinesi olan paketler bu kuralla
#eşleşecektir.
pass in on fxp0 proto tcp from any to fxp0 port www
```

7.4 "quick" Kelimesi

Daha önce de bahsettiğimiz gibi PF ayar dosyasında daima **en son yazılan kurallar** geçerlidir. Yani aşağıdan-yukarıya doğru bir önem sırası mevcuttur. Bu sırayı **quick** kelimesi ile değiştirmemiz mümkündür. Bütün kurallara quick kelimesi eklediğimiz takdirde sıralamada başta yazılan kurallar geçerli kılınacaktır.

```
pass in quick proto tcp from any to 10.1.2.3
block in all
```

Bu kural dizilimine göre `10.1.2.3` adresine giden TCP paketleri dışında bütün paketler engellenecektir.

7.5 Durum Koruma

Paket Süzgeçleri'nin en önemli özelliklerinden bir tanesi "`keeping state`" veya "`stateful inspection`" olarak bilinen durum koruma özelliğidir. Durum koruması bir ağ bağlantısının durumu ve ilerleyişinden haberdar olmak anlamına gelir. PF her bağlantının durumunu bir durum tablosunda tutarak, çok hızlı bir şekilde paketin ateş duvarı kayıtlarından geçip geçmeyeceğine karar verir.

Durum koruması daha basit kural yazılımı, daha iyi paket süzme özellikleri gibi bir çok avantaj sağlar. Örneğin PF hem gidiş hem de geliş olmak üzere iki farklı yönde de eşleştirme yapabilmektedir. Durum koruması sayesinde dönen trafik için ayrı kurallar yazmaya gerek kalmayacaktır. Dolayısıyla bir paket için PF'in işleme süresi de düşeceğinden, performans da oldukça artacaktır.

Kurallardan biri "`keep state`" ifadesini içerdiğinde, bu kurala uyan ilk paket gönderici ve alıcı arasında bir "durum" oluşturur. Bu sayede sadece göndericiden alıcıya giden paketler durum tablosundaki girdiyle eşleşmez, aynı zamanda alıcıdan göndericiye doğru iletilen paketler de eşleşir. Örneğin:

```
pass out on fxp0 proto tcp from any to any keep state
```

Bu satırla `fxp0` arabiriminden dışarıya doğru çıkan paketlere izin verildiği gibi bu arabirime gelen cevap paketlerine de izin verilir.

"`modulate state`" tanımı da "`keep state`" gibidir. Tek farkı sadece TCP için geçerlidir. "`modulate state`" ile dışarıya doğru çıkan paketler için rastgele bir Başlangıç Sıra Numarası (ISN) belirlenir. Bu da bazı işletim sistemlerinin zayıf sıra numarası seçiminin önüne geçerek bağlantıları korur. OpenBSD 3.5 ile birlikte "`modulate state`" tanımı TCP dışındaki protokoller için de kullanılmaya başlanmıştır.

Örnek olarak aşağıdaki kural dışarıya doğru çıkan TCP, UDP ve ICMP Paketlerinin durumunu tutup, TCP ISN'lerini de modüle eder:

```
pass out on fxp0 proto { tcp, udp, icmp } from any \
    to any modulate state
```

Durum korumasının başka bir avantajı da TCP paketleri ile ilgili gelen ICMP paketlerini de, durum tablosu girdileri ile eşleştirmesidir. Örneğin belirli bir TCP bağlantısı için gelen "*ICMP source-quench*" paketi de durum tablosundaki ilgili TCP kayıtları ile eşleşip, ateş duvarından geçecektir.

Bir durum girdisinin kapsamı, *if-bound, group-bound* ve *floating* durum anahtar kelimeleriyle kural temelli olarak `state-policy` başlangıç seçeneği tarafından global olarak kontrol edilir. Bu anahtar kelimeler `state-policy` seçeneği ile birlikte kullanıldığında aynı anlama gelmektedirler. Örneğin:

```
pass out on fxp0 proto { tcp, udp, icmp } from any \
    to any modulate state (if-bound)
```

NOT: `nat`, `binat` ve `rdr` kuralları da, ateş duvarı kural kümesinden geçtiği müddetçe durum tablosunda girdi oluştururlar.

7.5.1 UDP için durum koruma

Bazılarına göre UDP durumsuz (stateless) bir protokol olduğu için durum bilgisi de tutulamaz. UDP'nin bağlantılarla ilgili durum bilgisi tutmadığı doğrudur fakat bu PF'in bir UDP oturumu için durum girdisi oluşturmasını engellemez. PF eşleşen paketin gönderilmesinden beri geçen sürenin bilgisini tutmaktadır. Bu süre için bir bitiş süresi belirlenir ve bu süre dolduğunda durum sıfırlanır. Bu bitiş süreleri `pf.conf` dosyasındaki tercihler (options) bölümünde belirtilebilir.

7.6 Durumsal İzleme Seçenekleri

PF tarafından "`keep state`", "`modulate state`" veya "`synproxy state`" ifadelerinden herhangi birisi kullanılarak durum girdileri (`state entry`) oluşturulduğunda, durum davranışını belirleyecek bazı parametreler de belirtilebilir:

`max` *`sayı`* **:** Maksimum durum girdi sayısını belirler. Maksimuma ulaşıldığında durum tablosuna girdi ekleyecek olan paketler iptal edilir. Bu işlem durum tablosundaki girdi sayısı tekrar düşene kadar devam eder.

`source-track` **:** Bu seçenek kaynak IP Adresi başına oluşturulan durumların sayısını tutmayı sağlar. İki farklı biçimi vardır:

- **`source-track rule`** – Bu kural ile oluşturulan maksimum durum sayısı, kuralın `max-src-nodes` ve `max-src-states` gibi parametreleriyle sınırlandırılır. Sadece bu özel kural tarafından oluşturulan durum girdileri, o kuralın limitine kadar işlenir.
- **`source-track global`** – Bu seçeneği kullanan bütün kurallar tarafından oluşturulan durumların sayısı sınırlandırılır. Kuralların her biri değişik `max-src-nodes` ve `max-src-states` seçenekleri belirtebilir, fakat yine de her kuralın oluşturduğu girdi, kendi kuralının limiti içinde değerlendirilir.

Global olarak izlenen toplam kaynak IP Adreslerinin sayısı, `src-nodes` çalışma esnası parametresiyle kontrol edilir.

max-src-nodes *sayı* **:** `source-track` parametresi kullanıldığında, `max-src-nodes` aynı anda durum girdisi oluşturan kaynak IP adresi sayısını sınırlandıracaktır. Bu limitin kapsamı `source-track` parametresine bağlıdır.

Örneğin :

```
pass in on $ext_if proto tcp to $web_server \
    port www flags S/SA keep state \
    (max 200, source-track rule, max-src-nodes 100, max-src-states 3)
```

Bu kural aşağıdaki davranışları tanımlar:

- Bu kuralın oluşturabileceği mutlak maksimum durum sayısı 200 ile sınırlandırılmıştır.
- Kaynak izlemeyi (`source tracking`) etkinleştirir; sadece bu kural tarafından oluşturulan durumları temel alarak durum oluşturmayı sınırlandırır.
- Aynı anda durum oluşturacak maksimum düğüm sayısı 100 ile sınırlandırılmıştır.
- Kaynak IP başına aynı anda oluşturulabilecek maksimum durum sayısı 3 ile sınırlandırılmıştır.

7.7 TCP Bayrakları

Bayrakları temel alarak TCP paketlerini eşleştirme, genellikle yeni bağlantı açmaya çalışan TCP paketlerini süzmek için kullanılır. TCP bayrakları ve anlamları aşağıda listelenmiştir:

- **F :** FIN – (Finish); oturumun sonu
- **S:** SYN – (Synchronize); bir oturuma başlama isteğini işaret eder.
- **R:** RST – (Reset); Bir bağlantıyı sonlandırır.
- **P:** PUSH – (Push); Paket anında gönderilir.
- **A:** ACK – (Acknowlegement); Paket Onaylama
- **U:** URG – (Urgent); Acil Gönderme
- **E:** ECE – (Explicit Congestion Notification Echo); Açıkça tıkanıklık belirtisi
- **W:** CWR – Tıkanıklık penceresi azaltılır.

PF'in TCP bayraklarını inceleyebilmesi için kurallarda "flags" ifadesinin kullanılması gerekmektedir.

Flags check/mask
"mask" kelimesi PF'in hangi bayrakları inceleyeceğini, "check" kelimesi ise bu bayraklardan hangilerinin aktif olduğunu belirtir.

```
pass in on fxp0 proto tcp from any to any port ssh flags S/SA
```

Bu kural ile PF, TCP trafiğinden sadece S ve A (SYN ve ACK) bayraklarına bakar ve bunların içinden S bayrağı aktif olanları geçirir.

OpenBSD'nin eski versiyonlarında "… flags S" şeklinde bir tanımlama yapılabiliyordu fakat artık maske kullanılmadan tanımlama yapılamıyor.

Bayraklar durum girdilerinin oluşturulmasının kontrolünde kolaylık sağlamak amacıyla genellikle "keep state" ifadesiyle birlikte kullanılmaktadır.

```
Örneğin:
pass out on fxp0 proto tcp all flags S/SA keep state
```

Bayrakları kullanarak kural yazarken dikkatli olunmasında fayda vardır ve öneride bulunan insanların söylediklerini de iyi değerlendirmek gerekir. Örneğin bazıları sadece SYN bayraklı paketler için

durum oluşturulmasını önerebilirler. Bu pek iyi bir fikir olmayıp sonucunda ortaya şöyle bir kural çıkacaktır:

```
. . . flags S/FSRPAUEW
```

Aşağıda belirtilen tanımlama oldukça idealdir:

```
. . . flags S/SAFR
```

Bu tanım oldukça güvenli ve pratik olmasına karşın, scrub edilmiş trafik için FIN ve RST'i kontrol etmek gereksizdir. Scrub işlemi PF'in illegal TCP bayrak kombinasyonlarını içeren paketleri (Ör: STN ve FIN veya SYN ve RST) yok saymasını (drop) sağlar. Dolayısıyla gelen trafikte scrub kullanılması şiddetle önerilmektedir.

```
scrub in on fxp0
.
.
.
pass in on fxp0 proto tcp from any to any port ssh flags S/SA \
   keep state
```

7.8 TCP SYN PROXY

Normalde PF, istemci ile sunucu arasında bağlantının başlama safhasındaki 3-yollu el sıkışma paketlerini iletir. Fakat PF istenildiğinde "`synproxy state`" ifadesini kullanarak 3. yollu el sıkışma paketlerini doğrudan sunucuya iletmeden öncelikle kendisi gerçekleştirir ve bunun sonucunda el sıkışma başarılı bir şekilde gerçekleşirse asıl sunucu ile bağlantı kurulur. Bu özellik sayesinde el sıkışma işlemini başarılı bir şekilde gerçekleştiremeyen istemciler PF'den geçemeyecek ve dolayısıyla TCP SYN seli gibi sunucuyu etkileyecek saldırılar da devre dışı kalmış olacaktır.

TCP SYN Proxy özelliği şu şekilde tanılanmaktadır:

```
pass in on $ext_if proto tcp from any to $web_server port www \
   flags S/SA synproxy state
```

Aynı zamanda "`synproxy state`" ifadesi ile yazılan kurallar "`keep state`" ve "`modulate state`" ifadelerini de içine alır.

Eğer PF "`bridge`" modunda çalıştırılıyorsa, SYN PROXY aktif olmayacaktır.

7.9 "SPOOF" Edilmiş (Aldatılmış) Paketleri Engellemek

"Spoof" kelimesini Türkçe'ye aldatma olarak çevirebiliriz. Aldatma gerçek IP'yi gizlemek veya paketleri başka bir istemciden geliyormuş gibi göstermek için kullanılan kaynak IP Adresini değiştirmek suretiyle oluşturulan bir saldırıdır. Aldatma saldırısından sonra kişi bir ağ saldırısı gerçekleştirebilir veya normalde yetki verilmeyen yerlere yetkili bir IP'yi alarak erişebilir.

PF, aldatma saldırılarına karşı "antispoof" ifadesiyle koruma sağlamaktadır.

`antispoof [log] [quick] for` ***`Arabirim`*** **`[`*** ***`af`*** **`]`**

`log` : Eşleşen paketlerin `pflogd(8)` vasıtasıyla kayıt edileceğini belirtir.

quick : Bu parametreyi içeren bir kural ile eşleşme olduğunda kural kümesindeki diğer kurallar işleme konulmadan, o kural aktif olur.

interface : Aldatma saldırısından korunmanın hangi arabirim için aktif hale getirileceğini belirtir. Arabirim liste halinde de verilebilir.

af : Aldatma saldırısından korunmanın hangi adres ailesi (IPv4 için inet, IPv6 için inet6) için geçerli olacağını belirtir.

Örnek Tanımlama:

```
antispoof for fxp0 inet
```

PF kural kümesi yüklendiğinde "antispoof" ifadesi bulunan kurallar iki ayrı kural olarak genişletilir. Örneğin fxp0 arabiriminin 10.0.0.1 IP adresi ve 255.255.255.0 alt ağ maskesi olduğunu farz edelim. Dolayısıyla yukarıdaki antispoof kuralı şu şekilde genişletilecektir:

```
block in on ! fxp0 inet from 10.0.0.0/24 to any
block in inet from 10.0.0.1 to any
```

Bu kurallar iki şeyi sağlar:

- 10.0.0.0/24 ağından gelen ve fxp0 arabiriminden geçmeyen bütün trafiği engeller. (fxp0 arabirimi de 10.0.0.0/24 ağına ait olduğu için, 10.0.0.0/24 ağına ait olup da, fxp0 arabiriminden geçmeyen bir paketin görülmesi imkansızdır).
- fxp0'ın IP adresi olan 10.0.0.1'den gelen bütün trafiği engelle. Bir ağ aygıtı hiçbir zaman kendisine başka bir arabirim üzerinden paket göndermez. Dolayısıyla böyle bir paket büyük ihtimalle zararlı bir pakettir.

NOT: Aldatma saldırısını önlemek için oluşturduğumuz kurallar aynı zamanda loopback arabiriminden yerel adrese gönderilen paketleri de engelleyeceği için ayrı bir kuralla bu durumun önüne geçilebilir:

```
pass quick on lo0 all
antispoof for fxp0 inet
```

IP adresi olmayan bir arabirimde "antispoof" ifadesinin kullanılması sonucu şu kurallar oluşturulacaktır:

```
block drop in on ! fxp0 inet all
block drop in inet all
```

Bunun sonucunda `fxp0` arabirime gelen bütün paketler engelleneceği için, "antispoof" un hangi arabirimde kullanıldığına dikkat edilmesi gerekir.

7.10 Pasif İşletim Sistemi Tespiti

Pasif işletim sistemi tespiti, pasif olarak uzaktaki istemcinin TCP SYN paketlerindeki bazı karakteristiklerin incelenerek işletim sisteminin tespit edilmesi anlamına gelir. Bu seçenek daha sonradan PF kurallarında bir kriter olarak kullanılabilir.

PF, tespit işlemini TCP SYN paketinin karakteristiklerini /etc/pf.os dosyasındaki izler ile karşılaştırarak gerçekleştirir. PF aktif hale geldiğinde geçerli işletim sistemi izleri şu komutla görüntülenebilir:

```
# pfctl -s osfp
```

Bir iz ateş duvarı kurallarında İşletim sisteminin sınıfı (*OS class*), versiyonu, alttip/yama (*subtype/patch*) seviyesine göre belirtilir. Bunların her biri pfctl komutu çıktısında görülebilir. Bir izi süzme kurallarında belirtmek için "`os`" ifadesi kullanılır.

```
pass  in on $ext_if from any os OpenBSD keep state
block in on $ext_if from any os "Windows 2000"
block in on $ext_if from any os "Linux 2.4 ts"
block in on $ext_if from any os unknown
```

İşletim sisteminin izi tespit edilemediği durumlarda "unknown" ifadesi ile karşılaşılır.

NOTLAR:

- Bazı paketler sanki başka bir işletim sisteminden geliyormuş gibi aldatılabilir. Dolayısıyla bu gibi durumlarda sistem tespiti de yanlış olacaktır.
- Bazı revizyonlar ve yamalar işletim sistemi yığıtının davranışını değiştirebilir ve bu da paketlerin iz dosyasıyla eşleşmemesine veya başka bir işletim sisteminin iziyle eşleşmesine neden olabilir.
- Pasif İşletim Sistemi Tespiti sadece TCP SYN paketlerinde geçerlidir. Başka protokollerde veya daha önceden kurulmuş halihazır bağlantılarda çalışmayacaktır.

7.11 IP Seçenekleri

Varsayılan olarak PF, IP seçeneği aktif edilmiş paketleri engeller. Bu da nmap programı gibi işletim sistemi tespitinde kullanılan programların işini zorlaştırmaktadır. Fakat eğer bu tip paketlerin geçmesine gerek duyan IGMP veya multicast gibi bir uygulama çalıştırıyorsanız, allow-opts direktifi kullanılabilir.

```
pass in quick on fxp0 all allow-opts
```

7.12 Süzme Kural Kümesi Örneği

Aşağıda küçük bir ağ ile İnternet arasına yerleştirilmiş bir ateş duvarı için kural kümesi örneği verilmiştir. Bu kümede sadece süzme kuralları ele alınmış olup nat, rdr ve queueing devre dışı bırakılmıştır.

```
ext_if  = "fxp0"
int_if  = "dc0"
lan_net = "192.168.0.0/24"

# Ateş duvarına atanmış bütün IP adreslerini kapsayan tablo
table <firewall> const { self }

# Gelen paketleri scrub yap
scrub in all

# Varsayılan olarak reddet politikasını oluştur
block in  all
block out all

# loopback arabiriminde her iki yöne akan trafiğe izin ver
pass quick on lo0 all

# Dahili arabirim için antispoof korumasını aktif et.
antispoof quick for $int_if inet
```

```
# Yerel ağdan sadece güvenilir bilgisayara, 192.168.0.15
# ssh bağlantısı için izin ver. "block return" ifadesini kullan böylelikle
# engellenmiş bağlantıları kapatmak için anında TCP RST gönderilir.
# "quick" kullan, böylelikle aşağıdaki "pass" kuralları bu kuralın üzerine
# yazamaz.
block return in quick on $int_if proto tcp from ! 192.168.0.15 \
   to $int_if port ssh flags S/SA

# Yerel ağa gelen ve yerel ağdan gelen bütün trafiği geçir.
pass in  on $int_if from $lan_net to any
pass out on $int_if from any to $lan_net

# Harici arabirim üzerinden dışarıya doğru(İnternet) yönelen tcp, udp ve icmp
# paketlerine izin ver.
pass tcp, udp, and icmp out on the external (İnternet) interface.
# udp ve icmp için "keep state" tcp için ise "modulate state" uygula.
pass out on $ext_if proto tcp all modulate state flags S/SA
pass out on $ext_if proto { udp, icmp } all keep state

# Ateş duvarını hedef almadığı sürece (Örneğin yerel ağdaki bir makineyi hedef alan)
# harici arabirim üzerinde içeriye doğru gelen ssh bağlantılarına izin ver.
# Başlangıç paketini kayıt et, böylelikle daha sonra kimin bağlanmaya çalıştığını
# anlatabilelim. Vekil bağlantıları için "tcp syn proxy" kullan.
pass in log on $ext_if proto tcp from any to ! <firewall> \
   port ssh flags S/SA synproxy state
```

8. PF: AĞ ADRES DÖNÜŞÜMÜ (NAT-Network Address Translation)

Ağ Adres Dönüşümü bütün bir ağı (veya ağları) tek bir IP Adresi ile eşleştirmenin bir yoludur. Servis sağlayıcınızın size verdiği adresler, ağınızda İnternet dağıtmayı düşündüğünüz istemci sayısından azsa, mecburen NAT kullanmak zorundasınız. NAT RFC 1632'de tanımlanmıştır.

NAT RFC 1918'de belirtilen rezerve edilmiş IP bloklarını kullanır. En sık kullanılanları şu IP bloklarıdır:

```
10.0.0.0/8        (10.0.0.0 - 10.255.255.255)
172.16.0.0/12     (172.16.0.0 - 172.31.255.255)
192.168.0.0/16    (192.168.0.0 - 192.168.255.255)
```

NAT yapan bir OpenBSD sisteminin, biri İnternet'te, diğeri yerel ağda kullanılmak üzere en az iki adet ağ bağdaştırıcı kartına ihtiyacı vardır. NAT yerel ağdaki bütün paketleri sanki kendinden geliyormuş şekilde dönüştürür ve bu şekilde İnternet'e çıkmalarını sağlar.

8.1 NAT'ın Çalışması

Yerel ağdaki bir istemci İnternet'teki bir makine ile haberleşeceğinde, IP paketlerinin hedef adresini o makineyi gösterecek şekilde ayarlar. Bu paketler hedef ile iletişime geçmek için gerekli bütün adresleme bilgilerini içermektedir. NAT'ın ilgilendiği adresler ise şunlardır:

- Kaynak IP Adresi (Ör: 192.168.1.3)
- Kaynak TCP veya UDP kapısı (Ör: 2132)

Paketler NAT Ağ Geçiti üzerinden geçerken, kaynak adresi ağ geçidinin IP adresi olacak şekilde değiştirilir. Değişiklikler durum tablosuna kayıt edilir, bu şekilde paketlerin geri dönüşümü sağlanır ve

aynı zamanda ateş duvarı tarafından da bloklanmaları engellenmiş olur. Örneğin şu değişiklikler yapılabilir:

- Kaynak IP: Ağ geçidinin dış arabiriminin IP adresiyle değiştirilir. (Ör: 24.5.0.5)
- Kaynak Kapı: Kaynak kapı olarak kullanılmayan, rastgele bir kapı seçilir. (Ö: 53136)

Ne iç taraftaki istemci, ne de hedef makine bu değişikliklerin farkında değildir. NAT sistemi, istemci için basit bir İnternet ağ geçitidir. İnternet'teki bilgisayar için ise paketler doğrudan NAT geçitinden gelmektedir, içerideki başka bir bilgisayardan geldiğinden tamamen habersizdir.

İnternet bilgisayarı gelen bu paketlere cevap vereceği zaman NAT geçitine gönderir. Yani hedef IP'si 24.5.0.5 ve hedef kapısı 53136'dır. NAT geçidi durum tablosunu inceler ve daha önceden bu tip bir bağlantı kurulup kurulmadığını araştırır. Daha sonra IP/kapı eşleştirmesi sonucunda bu tip bir paketin kendisi üzerinden NAT yapıldığını anlar ve eskiden yaptığı değişiklerin tam tersini yaparak paketi içteki bilgisayara (192.168.1.35) ulaştırır. ICMP paketlerinin dönüşümü de benzer şekilde yapılır fakat bunlarda kapı olmadığından, kapı ile ilgili herhangi bir işlem yapılmaz.

8.2 NAT ve Paket Süzme

Dönüştürülmüş paketler de süzme kurallarından geçmek zorundadır. Yine tanımlanmış kurallara göre engelleme ve geçirme işlemleri yapılır. Tek bir istisnası, nat kuralları içerisinde "`pass`" kelimesi tanımlanmışsa, süzülmeden doğrudan geçirilirler.

Ayrıca dönüşüm süzmeden önce olduğu için, süzme motoru dönüştürülmüş paketleri görecektir.

8.3 IP Yönlendirme (IP Forwarding)

NAT genellikle yönlendirici ve ağ geçidi gibi cihazlar üzerinde gerçekleştirildiğinden, paketlerin ağ arabirimleri arasında iletilmeleri için IP yönlendirmenin aktif hale getirilmesine gerek duyulur. OpenBSD'de IP yönlendirme `sysctl` mekanizması ile aktif hale getirilir:

```
# sysctl net.inet.ip.forwarding=1
# sysctl net.inet6.ip6.forwarding=1 (if using IPv6)
```

Bu değişikliği kalıcı yapmak için ise, `/etc/sysctl.conf` dosyasına aşağıdaki satırları eklemek gerekir.

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

Varsayılan yüklemede bu tanımlar yapılmış ama önüne # işareti koyulmuş olabilir. # işaretlerini kaldırıp dosyayı kayıt ettiğimizde, makinenin bir sonraki açılışında bunlar aktif hale gelecektir.

8.4 NAT'ı Ayarlama

pf.conf'taki NAT kurallarının genel biçimi şu şekildedir:

```
nat [pass] on interface [af] from src_addr [port src_port] to \
   dst_addr [port dst_port] -> ext_addr [pool_type] [static-port]
```

nat : bir nat kuralı yazmaya başlanacağını gösterir.

pass : ilgili kuralın süzme kurallarını tamamen baypas geçmesini sağlar.

interface : üzerinde paketlerin dönüştürüleceği arabirimin ismi.

af : IPv4 için inet, IPv6 için inet6 olan adres ailesi. PF genellikle bu parametreyi kaynak ve hedef IP adresine bakarak anlamaktadır.

src_addr : dönüştürülen paketlerin kaynak IP adresidir ve şu şekillerde ifade edilebilir:

- Tek bir IPv4 veya IPv6 adresi
- Bir CIDR ağ bloğu
- Kural kümesi yüklendiğinde tam yetkilendirilmiş domain ismi. Sonuç olarak elde edilen IP Adresleri kurallarda yerine koyulacaktır.
- Bir Ağ arabiriminin ismi. Arabirime atanmış herhangi bir IP Adresi kurallarda yerine koyulacaktır.
- `/netmask` (Ör: /24) ile izlenen bir ağ arabiriminin ismi. Arabirimdeki her bir adres CIDR ağ bloğu oluşturabilmek için ağ maskesi ile birleştirilir.
- Aşağıda belirtilen kelimelerin sonunda bulunduğu ağ arabirim isimleri.
  - **:network** – CIDR ağ bloğunu temsil eder. (Ör: 192.168.0.0/24)
  - **:broadcast** – Ağ *broadcast* adresini temsil eder.
  - **:peer** – noktadan noktaya (*point to point*) bağlantıda bir noktanın IP adresini temsil eder.

Ayrı yeten "**:0**" belirteci bir arabirim isminin arkasına veya yukarıda belirtilen belirteçlerden sonra getirilirse, bu PF'in takma IP Adresi kullanmayacağını gösterir.

src_port :

4. Katman paket başlığındaki kaynak kapısını belirtir. Kapılar şu şekillerde belirtilebilir:
   - 1 ile 65535 arasındaki bir sayı
   - `/etc/services` dosyasından geçerli bir servis ismi
   - listeler kullanılarak belirtilmiş kapı kümesi
   - belirli bir aralık:
     - != (eşit değil)
     - < (den küçük)
     - > (den büyük)
     - <= (den küçük veya eşit)
     - >< (aralık)
     - <> (ters aralık)

   Bu sembollerden son ikisi ikili (binary) operatörlerdir ve iki argüman alırlar.

     - **:** (Özel aralık)

NAT'taki amaç genellikle kapıya bakılmaksızın bütün trafiği dönüştürmek olduğu için, kapı seçeneği genellikle kullanılmaz.

dst_addr : Dönüştürülen paketlerin hedef adresidir. Kaynak adres ile aynı şekillerde tanımlanabilir.

dst_port : 4. Katman paket başlığında bulunan hedef kapı numarasıdır. Kaynak kapı ile aynı şekillerde tanımlanabilir.

ext_addr : NAT geçidindeki paketlerin dönüştürüleceği harici adrestir. Harici adres şu şekillerde tanımlanabilir:

- Tek bir IPv4 veya IPv6 adresi
- Bir CIDR ağ bloğu
- Kural kümesi yüklendiğinde tam yetkilendirilmiş domain ismi.
- Bir Ağ arabiriminin ismi. Arabirime atanmış herhangi bir IP Adresi kurallarda yerine koyulacaktır.
- Parantez ( ) içinde belirtilmiş ağ arabiriminin ismi. Bu tanımlama belirtilen arabirimin IP Adresinin değişmesi durumunda PF'e ilgili arabirimle ilgili kuralı güncelleştirmesi gerektiğini bildirir. Bu tanımlama genellikle DHCP veya çevirmeli bağlantı üzerinden IP alan arabirimler için oldukça faydalıdır. IP adresi her değiştiğinde PF'in kurallarının tekrar yüklenmesine gerek kalmayacaktır.
- Aşağıda belirtilen kelimelerin sonunda bulunduğu ağ arabirim isimleri.
    - **`:network`** – CIDR ağ bloğunu temsil eder. (Ör: 192.168.0.0/24)
    - **`:peer`** – noktadan noktaya (*point to point*) bağlantıda bir noktanın IP adresini temsil eder.

Ayrı yeten "`:0`" belirteci bir arabirim isminin arkasına veya yukarıda belirtilen belirteçlerden sonra getirilirse, bu PF'in takma IP Adresi kullanmayacağını gösterir.

`pool_type` **:** dönüşüm için kullanılacak adres havuzu tipini belirtir.

`static-port:` PF'e TCP ve UDP paketlerindeki kaynak kapı numaralarını dönüştürmemesini söyler.

Basit bir NAT tanımlaması şu şekildedir:

```
nat on tl0 from 192.168.1.0/24 to any -> 24.5.0.5
```

Bu kural 192.168.1.0/24 ağından gelen paketlerin kaynak adresini 24.5.0.5 yaparak tl0 üzerinden NAT yapılacağını belirtir.

Yukarıdaki kural doğru olmasına rağmen pek önerilmemektedir. Çünkü herhangi bir değişiklikte kuralı da güncelleştirmek gerekecektir. Dolayısıyla kuralı şu şekilde yazmak daha iyi olacaktır:

```
nat on tl0 from dc0:network to any -> tl0
```

Bu şekilde bir tanım yapıldığında, IP adresi pf.conf'un yüklenmesiyle belirlenir. Bu da DHCP kullanarak IP Adresi alan sistemlerde sorun çıkaracaktır. Bu sorundan kurtulmak için PF'e dönüşüm adresini otomatik olarak güncelleştirmesi gerektiği tanımlanabilir. Bu işlem için arabirim adını parantez içine almak yeterlidir.

```
nat on tl0 from dc0:network to any -> (tl0)
```

8.5 İki Yönlü Eşleme (1:1)

İki yönlü eşleme `binat` kuralları kullanılarak yapılır. `binat` kuralı bir iç IP adresi ile dış IP adresi arasında bire bir eşleme yapar. Bu tanımlama özellikle yerel ağda gerçek IP adresiyle bir web sunucu yayını yapma gibi durumlarda oldukça faydalıdır. İnternet'ten Web sunucunun dış IP'sine gelecek istekler iç IP'ye çevrilecek ve yine Web sunucu üzerinden DNS istemi gibi dışarıya yönelik istemler de dış IP'ye dönüştürülecektir. TCP ve UDP kapıları NAT kuralları ile birlikte oldukları için `binat` kurallarıyla hiçbir zaman değiştirilmezler.

Örnek:

```
web_serv_int = "192.168.1.100"
web_serv_ext = "24.5.0.6"

binat on tl0 from $web_serv_int to any -> $web_serv_ext
```

8.6 Dönüşüm Kuralları İstisnaları

Dönüşüm kurallarında istisnalar "no" kelimesi kullanılarak oluşturulabilir. Yukarıdaki örneği şu hale getirirsek:

```
no nat on tl0 from 192.168.1.208 to any
nat on tl0 from 192.168.1.0/24 to any -> 24.2.74.79
```

192.168.1.208 IP Adresi dışında, 192.168.1.0/24 ağına ait bütün adresler NAT edilecektir. Burada PF'den farklı olarak dikkat edilmesi gereken husus, **önce gelen kuralın geçerli olmasıdır**. "`no`" ifadesi aynı zamanda `binat` ve `rdr`'de de geçerlidir.

8.7 NAT Durumunun Kontrolü

Aktif NAT kurallarını görüntülemek için pfctl komutu "`-s state`" seçeneği ile kullanılır. Bu seçenek geçerli bütün NAT oturumlarını listeleyecektir:

```
   # pfctl -s state
   fxp0 TCP 192.168.1.35:2132 -> 24.5.0.5:53136 -> 65.42.33.245:22
TIME_WAIT:TIME_WAIT
   fxp0 UDP 192.168.1.35:2491 -> 24.5.0.5:60527 -> 24.2.68.33:53
MULTIPLE:SINGLE
```

İlk satırı ele alacak olursak:

fxp0 Gösterilen durumun bağlı olduğu arabirimdir. Eğer durum "floating" olsaydı yani durum herhangi bir arabirimdeki paketlerle eşleşebilir nitelikte olsadı, ekranda "self" şeklinde bir ifade görecektik.

TCP Bağlantı tarafından kullanılan protokol.

192.168.1.35:2132 Makinenin iç ağdaki IP Adresinin 192.168.1.35 ve kapı numarasının da 2132 olduğunu gösterir. Bu aynı zamanda IP başlığında yer değiştirilen IP adresidir.

24.5.0.5:53136 Paketlerin dönüştürüldüğü, ağ geçidi üzerinde bulunan IP adresi(24.5.0.5) ve kapı numarasıdır (53136).

65.42.33.245:22 içerideki makinenin bağlanmaya çalıştığı hedef IP Adresi (65.42.33.245) ve hedef kapı numarasıdır (22).

TIME_WAIT:TIME_WAIT PF'in, TCP bağlantısının hangi durumda olduğuna inandığını gösterir.

9. PF: YÖNLENDİRME (Kapı Aktarma)

Ofisinizde NAT çalıştırdığınızda İnternet, tüm makinelerce erişilebilir olacaktır. Eğer NAT geçidinin ardındaki bir makineye dışarıdan erişmeniz gerekirse ne yapacaksınız? İşte bu yönlendirmenin söz konusu olduğu noktadır. Yönlendirme gelen trafiğin NAT geçidinin ardındaki makineye ulaşmasını sağlar.

Aşağıdaki örneği inceleyelim:

```
rdr on tl0 proto tcp from any to any port 80 -> 192.168.1.20
```

Bu satır TCP kapı 80(web sunucusu) trafiğinin iç ağdaki 192.168.1.20 IP nolu makineye yönlendirilmesini sağlar. Bu şekilde 192.168.1.20 makinesi, NAT geçidinin ardında olmasına ve alt ağda olmasına rağmen dış dünyadan ulaşılabiliyor olacaktır.

`rdr` satırının "`from any to any`" kısmı çok kullanışlıdır. Eğer hangi adreslerin ve alt ağların web sunucusunun 80 kapısına erişebileceğini biliyorsanız, aşağıdaki şekilde bir sınırlandırma yapabilirsiniz:

```
rdr on tl0 proto tcp from 27.146.49.0/24 to any port 80 -> \
   192.168.1.20
```

Bu komut belirtilen alt ağlar için yönlendirme gerçekleştirir. Aynı zamanda dikkat ederseniz gelen değişik erişimcileri geçidin ardındaki değişik makinelere yönlendirebileceğiniz anlamına da gelmektedir. Bu çok kullanışlı olabilir. Örneğin, ayrı yerlerde kendi bilgisayarlarının başında olan istemcileriniz olabilir. Onların bağlandıkları IP adreslerini biliyorsanız, bu kullanıcıların geçidin ardındaki belirli bir IP ve kapı numarasına erişimini sağlayabilirsiniz.

```
rdr on tl0 proto tcp from 27.146.49.14 to any port 80 -> \
   192.168.1.20
rdr on tl0 proto tcp from 16.114.4.89 to any port 80 -> \
   192.168.1.22
rdr on tl0 proto tcp from 24.2.74.178 to any port 80 -> \
   192.168.1.23
```

9.1 Yönlendirme ve Paket Süzme

NOT: Dönüştürülen paketler hala süzme motorundan geçmek zorundadır ve süzme kurallarına göre geçip geçmeyecekleri belirlenir.

Bu kuralın tek istisnası `pass` anahtar sözcüğünün `rdr` satırında kullanılmasıdır. Bu durumda, yönlendirilen paketler, durumlarını koruyarak süzme motorundan direk olarak geçerler: süzme kuralları bu paketler için çalıştırılmaz. Bu, her yönlendirme kuralına `pass` süzme kuralı eklemekten daha kullanışlı bir yoldur. Bunu , `keep state` anahtar sözcüklü , `pass` süzme kuralı ile ilişkilendirilmiş normal bir `rdr` (`pass` anahtar sözcüğü olmayan) kuralı gibi düşünebilirsiniz. Ama, eğer `synproxy`, `modulate state` gibi özelleşmiş süzme seçenekleri kullanmak istiyorsanız hala yönlendirme kuralıyla ilişkili özel bir `pass` kuralına ihtiyacınız var.

Yönlendirme süzmeden önce gerçekleştirildiğinden, süzme motoru `rdr` kuralı ile belirtilen dönüştürülmüş paketin sahip olduğu hedef IP ve kapısına göre hareket eder. Asıl paketin IP adresi ve kapı numarası kullanılmaz. Buna dikkat edilmesi gerekir. Aşağıdaki durumu inceleyelim:

- 192.0.2.1 – İnternet üzerindeki istemci
- 24.65.1.13 – OpenBSD yönlendirici(*router*) dış adresi
- 192.168.1.5 – web sunucusunun iç IP adresi

Yönlendirme kuralı:

```
rdr on tl0 proto tcp from 192.0.2.1 to 24.65.1.13 port 80 \
   -> 192.168.1.5 port 8000
```

`rdr` kuralı çalıştırılmadan önce paket:

- Kaynak adres: 192.0.2.1
- Kaynak kapı: 4028 (işletim sistemi tarafından rasgele belirlenmiş)
- Hedef adres: 24.65.1.13
- Hedef kapı: 80

`rdr` kuralı işletildikten sonra paket:

- Kaynak adres: 192.0.2.1
- Kaynak kapı: 4028
- Hedef adres: 192.168.1.5
- Hedef kapı: 8000

Süzme motoru, IP paketini dönüştürüldükten sonra görür.

9.2 Güvenlik Boyutu

Yönlendirmenin güvenlik boyutu da oldukça önemlidir. Trafiğin içeriye girmesi için ateş duvarından bir delik açmak, içerdeki makineyi erişime açar. Örneğin içerdeki web sunucusuna trafik yönlendirilirse ve web sunucu arka plan programında (daemon bkz. http://www.webopedia.com/TERM/d/daemon.html) ya da CGI kodlarının çalışmasında bir açık tespit edildiğinde, makine İnternet üzerinden sızmaya açık hale gelir. Böylelikle, sızmaya çalışan kişi için iç ağa açılan ve doğruca ateş duvarından geçebileceği bir kapı oluşmuş olur.

Bu riskler, dışarıdan ulaşılan sistemleri ayrı bir ağa yerleştirmek ile en aza indirgenebilir. Bu ağ genel olarak Silahsızlandırılmış Bölge(*DMZ*) veya Özel Hizmet Ağı(*PSN*) olarak adlandırılır. Bu şekilde *DMZ/PSN* bölgesinden gelen ve bu bölgeye giden trafik iyi bir şekilde süzüldüğü taktirde web sunucu ele geçirilse dahi saldırı sadece bu bölgede sınırlı kalacak.

9.3 Yönlendirme ve Yansıtma

Genellikle yönlendirme kuralları, İnternet'ten gelen paketleri, LAN'da veya iç ağda bulunan özel adrese sahip yerel bir sunucuya doğru aktarmada kullanılır.

server = 192.168.1.40

```
rdr on $ext_if proto tcp from any to $ext_if port 80 -> $server \
   port 80
```

Ama yönlendirme kuralları yerel ağdaki bir istemci tarafından test edildiğinde, bu komut çalışmaz. Bunun sebebi yönlendirme kurallarının yalnızca belirli bir arabirimden geçen paketler için çalışmasıdır. (`$ext_if`, bu örnekte olduğu üzere dış arabirimdir). Ateş duvarının dışındaki bir adrese yerel ağdaki bir istemcinin bağlanması ise paketlerin bu dış arabirimden geçtiği anlamına gelmez. Ateş duvarındaki TCP/IP yığıtı (*stack*) gelen paketlerdeki hedef adresini kendi adresleri ve eş isimli kabul edildiği adreslerle karşılaştırır ve kendi iç arabirimden paketler geçer geçmez direk kendine olan bağlantılarını tespit ederler. Böylesi paketler fiziksel olarak dış arabirime geçirilmez ve yığıt bu geçişi bir benzetime uğratmaz (*simulation*). Bu yüzden, PF bu paketleri asla dış arabirimde görmez ve dış arabirim için tanımlanan yönlendirme kuralı da asla uygulanmaz.

İç arabirim için bir yönlendirme komutu daha girmek de istenen sonucu vermez. Yerel istemci ateş duvarının dış arabirime bağlandığında, TCP el sıkışma safhasının ilk paketleri iç arabirimden ateş duvarına ulaşır. Yönlendirme kuralı uygulanır ve hedef adres dâhili sunucunun adresi ile değiştirilir. Paket yeniden iç arabirime yönlendirilir ve dahili sunucuya ulaşır. Ama kaynak adres dönüştürülmez ve hala yerel istemcinin adresini taşır, bu nedenle sunucu cevaplarını direk olarak istemciye gönderir. Ateş duvarı cevapları asla görmez ve iletişimi uygun şekilde dönüştürme şansı olmaz. İstemci cevapları, asla beklemediği bir kaynaktan alır ve yok sayar. TCP el sıkışma safhası başarısız olur ve bağlantı kurulamaz.

Hala, yerel ağdaki istemcilerin dışarıdan bağlanan bir istemci gibi aynı iç sunucuya bağlanması ve bunun saydam bir şekilde gerçekleşmesi istenen bir şeydir. Bu sorunun pek çok çözümü vardır:

9.3.1 Yatay bölünmüş DNS

DNS sunucuların içerdeki kullanıcıların isteklerine, dışarıdaki istemcilerin isteklerinden farklı olarak cevap vermesini sağlamak mümkündür bu sayede yerel istemciler iç sunucuların adresini isim çözümleme sırasında alırlar. Bundan sonra yerel sunucuya doğrudan bağlanabilirler ve ateş duvarı bu süreçlere katılmaz. Paketler ateş duvarından geçmediğinden yerel trafik de azalır.

9.3.2 Sunucuyu ayrı bir yerel ağa taşımak

Ateş duvarına ek bir ağ arabirim ilave etmek ve yerel sunucuyu istemcinin ağından alarak DMZ'e taşımak yerel kullanıcıların bağlantılarının da harici bağlantılar gibi yönlendirilebilmesine imkan verir. Ayrı ağlar kullanmanın pek çok avantajı vardır. Örneğin; sunucuyu alt ağdaki diğer istemcilerden soyutlayarak güvenliği arttırmış oluruz. Sunucu (bizim örneğimizde olduğu üzere İnternet'den ulaşılabilen bir sunucu) ele geçirilse bile, bunun üzerinden yerel istemcilere doğrudan bağlantı yapılamayacaktır çünkü bütün bağlantılar ateş duvarında geçmek zorundadır.

9.3.3 TCP vekaleti (*Proxying*)

Sıradan bir vekil TCP, ateş duvarı üzerine kurulabilir. Kurulum, vekilin dinlediği kapıya iç arabirimden gelen bağlantıları kabul etmesi ile olabileceği gibi yönlendirilecek kapıyı izlemesi ile de gerçekleştirilebilir. Yerel bir istemci ateş duvarına bağlandığında, vekil bağlantıyı kabul eder, iç sunucuya ikinci bağlantı gerçekleştirir ve bu iki bağlantı arasında veriyi aktarır.

Basit vekiller `inetd(8)` ve `nc(1)` kullanılarak oluşturulabilir. Aşağıdaki `/etc/inetd.conf` kaydı, *loopback* adresine (127.0.0.1) ve 5000 kapısına atanmış, dinleyen bir soket oluşturur. Bağlantılar 192.168.1.10 sunucusundaki 80 kapısına aktarılır.

```
127.0.0.1:5000 stream tcp nowait nobody /usr/bin/nc nc -w \
   20 192.168.1.10 80
```

Aşağıdaki yönlendirme kuralı iç arabirimdeki 80 kapısına gelen istekleri vekile aktarır:

```
rdr on $int_if proto tcp from $int_net to $ext_if port 80 -> \
   127.0.0.1 port 5000
```

9.3.4 RDR ve NAT birleşimi

Dahili arabirimde ek bir NAT kuralıyla, yukarıda belirtilen kaynak adres dönüşüm eksikliği giderilebilir.:

```
rdr on $int_if proto tcp from $int_net to $ext_if port 80 -> \
   $server
no nat on $int_if proto tcp from $int_if to $int_net
nat on $int_if proto tcp from $int_net to $server port 80 -> \
   $int_if
```

Bu komut, istemciden gelen birincil paketin yeniden dahili arabirime yönlendirildiğinde, istemcinin kaynak adresini ateş duvarının dahili adresi ile değiştirmesi ile, paketin yeniden dönüştürülmesini sağlar. Dahili sunucu, yerel istemciye aktarım esnasında hem NAT hem de RDR dönüşümlerini tersine çeviren ateş duvarına tekrar cevap verecektir. Bu yapı her bir yansıyan bağlantı için iki ayrı durum oluşturacağından oldukça karmaşıktır. NAT kuralını dışarıdan başka yönlendirmelerle gelen veya ateş duvarının kendisinden gelen bağlantılardan kaynaklı paketlere uygularken dikkat edilmesi gerekir. Örneğin yukarıdaki `rdr` kuralı TCP/IP yığıtının dahili arabirimine ulaşan paketleri, iç ağdaki bir hedef adrese yönelik olarak görmesine sebep olur.

Genel olarak, daha önce bahsedilen çözümler bunun yerine kullanılabilir.

10 KURAL KÜMELERİ OLUŞTURMAK İÇİN KISA YOLLAR

PF, bir kural kümesini sadeleştirmek için pek çok yol sunar. Makro ve liste tanımları bunlara en güzel örnektir. Buna ek olarak, kural kümesi dili ya da grameri, aynı zamanda kural kümesini daha da basitleştirmek için bazı kısa yollar sunar. Fihrist ile amaçlanan basitleştirmedir. Bir kural kümesi ne kadar basitse o kadar iyi anlaşılabilir ve yönetilebilirdir. Fihrist bu anlayış çerçevesinde biçimlendirilmelidir.

10.1 Makroların Kullanımı

Makrolar kural kümesindeki adresler, kapı numaraları, arayüz isimleri gibi doğrudan yapılan kodlamalar için bir alternatif oluşturdukları için oldukça kullanışlıdırlar. Örneğin bir sunucunun IP adresi değiştiğinde sadece makroyu güncellemeniz yeterli olacaktır. Bu sayede bir hayli enerji harcayarak oluşturduğunuz süzme kurallarıyla uğraşmanıza de gerek kalmayacaktır.

Geleneksel olarak PF kural kümeleri her bir ağ arabirimi için bir makro tanımı yapar. Eğer bir ağ kartının farklı bir sürücü kullanan bir başka kartla değiştirilmesi gerekirse, örneğin *3Com* ile *Intel*'i değiştirecek olursanız, sadece makro dosyasını değiştirmeniz yeterlidir. Süzme kuralları aynen çalışmaya devam edecektir. Makronun sunduğu bir başka fayda da aynı kural kümesini bir çok makineye yükleme sırasında ortaya çıkar. Bazı makineler, üzerlerinde farklı ağ kartları taşıyabilirler, ve makroların kullanımı kurulacak olan kural kümelerinin en az güncellemeyle ağ arabirimlerine

atanmasını sağlar. Özellikle değişikliğe konu olabilecek IP adresleri, kapı numaraları ve arabirim isimleri için makro kullanılması önerilmektedir.

```
# Her bir arayüz için makro tanımla
IntIF = "dc0"
ExtIF = "fxp0"
DmzIF = "fxp1"
```

Bir başka geleneksel yol da, IP adresleri ve ağ bloklarının tanımlanmasında makro kullanmaktır. Bu sayede IP adreslerinin değişikliğe uğraması gibi durumlarda kural kümelerinin korunurluğunu ciddi ölçüde artar.

```
# Ağımızı tanımlayalım
IntNet = "192.168.0.0/24"
ExtAdd = "24.65.13.4"
DmzNet = "10.0.0.0/24"
```

Eğer dahili ağ genişler ya da değişik bir IP bloğuna taşınırsa, makro buna göre güncellenebilir:

```
IntNet = "{ 192.168.0.0/24, 192.168.1.0/24 }"
```

Kural kümesi yeniden yüklenince, her şey eskisi gibi çalışmaya devam eder.

10.2 Liste Kullanımı

Şimdi, RFC 1918'de belirtilen, İnternet'de dolaşmaması gereken ve dolaşmaya kalktığında sorunlara yol açan adreslerin ele alındığı iyi bir kural kümesi örneğini inceleyelim:

```
block in  quick on tl0 inet from 127.0.0.0/8 to any
block in  quick on tl0 inet from 192.168.0.0/16 to any
block in  quick on tl0 inet from 172.16.0.0/12 to any
block in  quick on tl0 inet from 10.0.0.0/8 to any
block out quick on tl0 inet from any to 127.0.0.0/8
block out quick on tl0 inet from any to 192.168.0.0/16
block out quick on tl0 inet from any to 172.16.0.0/12
block out quick on tl0 inet from any to 10.0.0.0/8
```

Şimdi bu kurallar içinde ne gibi basitleştirme yapabiliriz, buna bir göz atalım:

```
block in  quick on tl0 inet from { 127.0.0.0/8, 192.168.0.0/16, \
   172.16.0.0/12, 10.0.0.0/8 } to any
block out quick on tl0 inet from any to { 127.0.0.0/8, \
   192.168.0.0/16, 172.16.0.0/12, 10.0.0.0/8 }
```

Bu şekilde kural kümesi 8 satırdan 2 satıra indirgenmiş oldu. Liste ile birlikte Makro kullanıldığında işler daha da düzelecektir:

```
NoRouteIPs = "{ 127.0.0.0/8, 192.168.0.0/16, 172.16.0.0/12, \
   10.0.0.0/8 }"
ExtIF = "tl0"
block in  quick on $ExtIF from $NoRouteIPs to any
block out quick on $ExtIF from any to $NoRouteIPs
```

Dikkat ederseniz makrolar ve listeler `pf.conf` dosyasını sadeleştirir, ama aslında satırlar `pfctl` tarafından genişletilerek daha çok satıra dönüştürülür. Böylece yukarıdaki kurallar aslında aşağıdaki şekli alır:

```
block in  quick on tl0 inet from 127.0.0.0/8 to any
block in  quick on tl0 inet from 192.168.0.0/16 to any
block in  quick on tl0 inet from 172.16.0.0/12 to any
block in  quick on tl0 inet from 10.0.0.0/8 to any
block out quick on tl0 inet from any to 10.0.0.0/8
block out quick on tl0 inet from any to 172.16.0.0/12
block out quick on tl0 inet from any to 192.168.0.0/16
block out quick on tl0 inet from any to 127.0.0.0/8
```

Görüldüğü üzere, PF açılımı, pf'in asıl kullandığı kuralları sadeleştirmez, sadece `pf.conf` dosyasının yazıcısı ile yöneticisi arasında bir uyumluluk sağlar.

Makrolar, adres ve kapı tanımlamaktan farklı olarak başka bir çok alanda kullanılabilirler. Örneğin makrolar PF kural dosyasının herhangi bir yerinde kullanılabilirler:

```
pre = "pass in quick on ep0 inet proto tcp from "
post = "to any port { 80, 6667 } keep state"

# David'in sınıfı
$pre 21.14.24.80 $post

# Nick'in evi
$pre 24.2.74.79 $post
$pre 24.2.74.178 $post
```

Açılmış hali:

```
pass in quick on ep0 inet proto tcp from 21.14.24.80 to any \
   port = 80 keep state
pass in quick on ep0 inet proto tcp from 21.14.24.80 to any \
   port = 6667 keep state
pass in quick on ep0 inet proto tcp from 24.2.74.79 to any \
   port = 80 keep state
pass in quick on ep0 inet proto tcp from 24.2.74.79 to any \
   port = 6667 keep state
pass in quick on ep0 inet proto tcp from 24.2.74.178 to any \
   port = 80 keep state
pass in quick on ep0 inet proto tcp from 24.2.74.178 to any \
   port = 6667 keep state
```

10.3 PF Grameri

Paket süzgeci, kural kümeleri için oldukça esnek bir yapıya sahiptir. PF, daha önceden açık bir şekilde kurallarda belirtilmesine gerek duymadan birtakım anahtar sözcükleri anlayabilmektedir. Aynı zamanda PF'de anahtar sözcük sıralaması da, söz dizimini ezberlemeyi gerektirmeyecek ölçüde esnek bir şekilde hazırlanmıştır.

10.3.1 Anahtar sözcüklerin elenmesi

Varsayılan olarak reddet ("`default deny`") politikası tanımlamak için aşağıdaki iki kural kullanılır:

```
block in  all
block out all
```

Bu iki kural aşağıdaki şekilde daha da indirgenebilir:

```
block all
```

Hiçbir yön belirtilmemişse PF, kuralın her iki yöndeki paketler için uygulanacağını varsayar.

Basitçe, "`from any to any`" ve "`all`" sözleri kuralda belirtilmeyebilir, örneğin;

```
block in on rl0 all
pass  in quick log on rl0 proto tcp from any to any port 22 keep state
```

şöyle basitleştirilebilir;

```
block in on rl0
pass  in quick log on rl0 proto tcp to port 22 keep state
```

İlk kural `rl0` arabiriminde herhangi bir yerden gelen ve herhangi bir yere giden paketleri engellerken, ikinci kural `rl0`'daki 22 numaralı kapıya yönelik TCP trafiğini geçirir.

10.3.2 Return sadeleştirmesi

Paketleri engellemeye yarayan ve geriye "TCP RST" veya "ICMP Unreachable" yanıtları veren bir kural kümesi aşağıdaki şekilde tanımlanabilir:

```
block in all
block return-rst in proto tcp all
block return-icmp in proto udp all
block out all
block return-rst out proto tcp all
block return-icmp out proto udp all
```

Bu kural kümesini şu şekilde sadeleştirebiliriz:
```
block return
```

PF "`return`" anahtar sözcüğünü görünce uygun cevabı gönderebilecek kadar akıllıdır ya da basitçe hiç cevap vermemeyi tercih edebilir. Bu, reddedilen paketin protokolüne göre belirlenir.

10.3.3 Anahtar sözcük sıralaması

Anahtar sözcük kullanımındaki sıralama pek çok durum için esnektir. Örneğin, aşağıdaki şekilde yazılan bir kural;

```
pass in log quick on rl0 proto tcp to port 22 \
   flags S/SA keep state queue ssh label ssh
```

aynı zamanda şu şekilde de yazılabilir:

```
pass in quick log on rl0 proto tcp to port 22 \
   queue ssh keep state label ssh flags S/SA
```

Yine buna benzer başka tanımlamalar da yapmak mümkündür.

SONSÖZ

Bu dökümanda, en güvenli işletim sistemleri arasında yer alan OpenBSD'nin varsayılan olarak sunduğu PF isimli ateş duvarının temel kullanımı için gerekli bilgiler aktarılmış olmakla birlikte, resmi bir OpenBSD dökümanı değildir. Yazarlar dökümanda anlatılan bilgilerin kullanımından kaynaklanacak zararlardan sorumluluk kabul etmezler. Bu döküman tamamen yada parça parça yazarların izni olmadan çoğaltılamaz, kaynak belirtilmeden alıntı yapılamaz.