

OpenBSD PF ile Güvenlik Duvarı Uygulamaları

1. Güvenlik Duvarı olarak OpenBSD
2. OpenBSD Kurulumu
3. Temel Sistem Ayarları
 - a. Nsh kullanımı
4. Packet Filter'a Giriş
 - a. PF
 - b. Kural yönetim dosyası ve düzeni
5. Paket Filtreleme
6. Nat işlemleri
 - a. Nat, rdr, binat
7. Packet Filter FTP
8. Packet Filter Loglama
9. Packet Filter Yönetim aracı (pfctl)
10. PF ile İleri Düzey işlemler
 - a. Çalışma Zamanı Seçenekleri
 - b. Görünmez Güvenlik Duvarı Kurulumu
 - c. Politika tabanlı Yönlendirme
 - d. Birden fazla internet bağlantısını PF ile yönetme
 - e. Trafik şekillendirme

Güvenlik Duvarı olarak OpenBSD

OpenBSD işletim sistemi temel amacı güvenlik ve kararlılık olan bir UNIX çeşididir. 10 yıllık bir geçmişi olan OpenBSD projesi, güvenlik konusunda bilişim dünyasına birçok konuda önder olmuştur.

OpenBSD, bir güvenlik duvarı için gerekli tüm şartları standartlara uygun, sağlam ve esnek bir yapıda sunar. İşletim sistemi, temelini oluşturan proaktif güvenlik politikası ile bilinen birçok güvenlik zaaflığına karşı korunduğu gibi geliştirdiği alternatif çözüm önerileri ile gelecekte çıkabilecek birçok problemi temelden çözmüş olur.

OpenBSD Kurulumu

Kurulum için öncelikle bir kurulum yöntemi belirlenmesi gerekir. OpenBSD, ftp, http, nfs gibi ağ tabanlı, disket ve cdrom gibi medya tabanlı sistemlerden kurulumu desteklemektedir. Diğer işletim sistemlerinden farklı olarak OpenBSD resmi bir cd medyası dağıtmaz. Eğer OpenBSD'yi cdrom aracılığı ile kurmak isterseniz önünüze iki seçenek çıkar: biri orjinal OpenBSD cd'lerinden edinmek -ki bu yöntem projeye maddi katkı da sağladığı için tercih edilmesi önemlidir - , diğeri de kendi openbsd cdimizi oluşturmak ya da daha önce oluşturan birilerinden edinmek.

Kendi OpenBSD ISO'sunu oluşturmak isteyenler için aşağıda komutlar yeterli olacaktır.

```
$mkdir -p /tmp/openbsd/4.0/i386
$cd /tmp/openbsd/4.0/i386

ftp://ftp.enderunix.org/pub/OpenBSD/4.0/i386/
dizini altındaki paketler indirilir.

$cd ../../
$mkisofs -r -b 4.0/i386/cdrom40.fs -c "boot.catalog" -o OpenBSD40.iso openbsd
```

Kendi OpenBSD sisteminizi oluşturacak kaynaklarınız yoksa <http://files1.enderunix.org/openbsd-iso/4.0/> adresinden tarafımdan oluşturulmuş OpenBSD cd imajlarını indirebilirsiniz.

Eğer daha önce OpenBSD kurulumu yapmadıysanız muhtemelen kurulum size sürpriz gibi gelecektir. OpenBSD kurulumu diğer tüm UNIX ve Linux'lerden farklıdır , bir kere alışınca da oldukça kolay geleceğini söyleyebilirim. Ortalama kurulum süresi, kurulum amacınıza göre 5-10 dakika arasında değişecektir. Kurulum öncesinde nasıl birşey ile

karşı karşıya olduğunuzu görmek isterseniz http://files1.enderunix.org/openbsd40_kurulum.avi adresindeki kurulum dosyasını izleyip fikir sahibi olabilirsiniz.

NOT: Kurulum videosunu izlemek için Vmware codec gerekiyor. Windows için <http://vmware-svca.www.conxion.com/software/VMware-moviedecoder-5.0.0-13124.exe> adresinden edinilebilir. Linux ve BSD için ise mplayer'in son sürüm codec paketini /usr/local/lib/win32/ dizinine koymak yetiyor.(vmnc.dll)

Not: OpenBSD projesi kullanıcılardan gelen destekler ve satılan Cd/t-shirtler aracılığı ile varlığını sürdürmektedir. Daha kaliteli OpenBSD geliştirmeleri için orjinal Cdlerden sipariş edebilir, OpenBSD baskılı t-shirtlerden satın alabilirsiniz. <http://www.openbsd.org/orders.html>

Temel Sistem Ayarları

OpenBSD Ağ ayarları

Ağ arabiriminin özelliklerini görüntüleme

Tüm UNIX sistemlerde olduğu gibi OpenBSD'de de ağ arabirimine ait özellikler **ifconfig** komutu ile görüntülenir.

```
# ifconfig
lo0: flags=8049<UP,LOOPBACK,RUNNING,MULTICAST> mtu 33224
    inet6 ::1 prefixlen 128
    inet6 fe80::1%lo0 prefixlen 64 scopeid 0x6
    inet 127.0.0.1 netmask 0xff000000
rl1: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> mtu 1500
    address: 00:50:bf:4e:dd:1a
    media: Ethernet autoselect (100baseTX full-duplex)
    status: active
    inet6 fe80::250:bfff:fe4e:dd1a%r1 prefixlen 64 scopeid 0x2
    inet 172.16.10.1 netmask 0xfffff00 broadcast 172.16.10.255
pflog0: flags=141<UP,RUNNING,PROMISC> mtu 33224
pfsync0: flags=0<> mtu 2020
enc0: flags=0<> mtu 1536
Enc0 = Enkapsilasyon arabirimi
Pflog0 paket filtreleme arabirimi
```

Ağ arabirimine Ip adresi atama

```
# ifconfig dc0 inet 192.168.0.3 netmask 255.255.255.255
```

ifconfig komutu ile yapılan değişiklikler sistemin ayakta olduğu müddetçe geçerlidir sistemin yeniden başlaması sonrasında bu ayarlar kaybolacaktır. Bu ayarları kalıcı hale getirmek için OpenBSD’de diğer unix’ler gibi ayarları bir dosyada tutar.

OpenBSD’de bir ağ arabirimine ait kalıcılık sağlayan yapılandırma dosyası **/etc/hostname.ARABIRIM_ADI** şeklindedir. Mesela fxp0 arabirimine ait kayıtlar etc/hostname.fxp0 dosyasında tutulur. Bu dosyanın formatı aşağıdaki gibidir.

Adres_sınıfı IP Adresi AĞ maskesi Broadcast [Diğer seçenekler]

Örnek;

```
$ cat /etc/hostname.fxp0  
inet 10.0.0.38 255.255.255.0 NONE
```

```
$cat /etc/hostname.rl0  
inet 10.0.0.38 255.255.255.0 NONE media 100baseTX mediaopt full-duplex
```

Detaylı bilgi **man hostname.if(5)** komutu ile edinilebilir.

Arabirime ek IP tanımlama

```
# ifconfig dc0 inet alias 192.168.0.3 netmask 255.255.255.255
```

Örnekten de görülebileceği gibi OpenBSD’de ağ arabirimine alias tanımlama sadece *alias* ek kelimesi ile olmaktadır. Yine bu alias tanımlarının kalıcı olması istenirse **/etc/hostname.arabirim.adı** dosyasına

Adres_sınıfı alias IP Adresi AĞ maskesi Broadcast [Diğer seçenekler]
Şeklinde girilmelidir.

Not: ifconfig -a komutu ile alias tanımları görüntülenmez, alias tanımlarında görüntülemek için ifconfig -A komutu ya da ifconfig arabirim_Adı komutları kullanılır.

```
#ifconfig -A
```

```
ifconfig r10
```

```
$ ifconfig r10
```

```
r10: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> mtu 1500  
  address: 00:e0:4c:39:4c:a7  
  media: Ethernet autoselect (100baseTX full-duplex)  
  status: active  
  inet6 fe80::2e0:4cff:fe39:4ca7%r10 prefixlen 64 scopeid 0x1  
  inet 192.168.0.89 netmask 0xffffffff broadcast 192.168.0.89  
  inet 192.168.0.88 netmask 0xfffff00 broadcast 192.168.0.255
```

```
# cat /etc/hostname.dc0
```

```
inet 192.168.0.2 255.255.255.0 media 100baseTX  
inet alias 192.168.0.3 255.255.255.255  
inet alias 192.168.0.4 255.255.255.255
```

Takma IP adresi silme işlemi

#ifconfig fxp0 delete komutu ile eklenen alias tanımını kaldırılabilir ya da daha sağlıklı bir çözüm olarak alias eklemek için verilen komuttaki alias parametresi **-alias** olarak değiştirilir.

```
# ifconfig dc0 inet alias 192.168.0.3 netmask 255.255.255.255  
Gibi..
```

DHCP sunucudan IP alma

```
# dhclient fxp0
```

```
# echo dhcp >/etc/hostname.xl0  
#sh /etc/netstart
```

DHCP sunucudan alınacak parameterleri belirlemek için **/etc/dhclient.conf** dosyası kullanılabilir.

Konak ismi(hostname) belirleme

Hostname komutu ile makine ismi belirlenebilir.

```
#hostname puffy.enderunix.org
```

komutu ile kullanılan sistemin tam tanımlı konak adı **test.enderunix.org** olarak düzenlenir. Bu değer kalıcı hale gelebilmesi için **/etc/myname** dosyasına yazılır.

```
$cat /etc/myname
```

test.enderunix.org

Yönlendirme tablosu görüntüleme

Netstat ve route komutları uygun parametrelerle kullanılarak sistemde tanımlı yönlendirme tabloları görüntülenir.

```
#netstat -rn -f inet
Routing tables

Internet:
Destination      Gateway          Flags    Refs    Use  Mtu  Interface
default          194.27.72.1     UGS      1 1292679  -   rl0
127/8            127.0.0.1       UGRS     0    0 33224  lo0
127.0.0.1        127.0.0.1       UH       2    638 33224  lo0
172.16.10/24     link#2           UC       1    0   -   rl1
172.16.10.2      0:d0:b7:b6:d1:c UHLc     1    26   -   rl1
194.27.72/24     link#1           UC       2    0   -   rl0
194.27.72.89     127.0.0.1       UGHS     0    0 33224  lo0
```

Varsayılan ağ geçidi tanımlama

```
#route add default IP_Adresi
```

Bu ayarın kalıcı olabilmesi için /etc/mygate dosyasına yazılması gerekir.

OpenBSD makineyi Router olarak kullanmak(IP FORWARDING)

```
/etc/sysctl.conf dosyasındaki
net.inet.ip.forwarding=1 şeklinde bir kayıt ekleyerek yapılır.
# sysctl net.inet.ip.forwarding=1
net.inet.ip.forwarding: 0 -> 1
```

Dosyalarla yapılan işlemlerdeki yapılandırmaların geçerli olabilmesi için

```
#sh /etc/netstart
```

komutunun çalıştırılması gerekir. Bu işlem ile OpenBSD'ye basit bir yönlendirici(router) vazifesi vermiş olduk. Tam bir router olarak çalışması için zebra, quaggaopenbgpd gibi yazılımlar kullanılabilir.

Gereksiz servisleri Durdurma

OpenBSD’de kurulum sonrası öntanımlı olarak bazı servisler çalışır durumda gelir. Bu servisleri kontrol ederek kullanılmayacak olanlarını kapatmak güvenlik açısından yerinde bir karardır.

Çalışan sunucu servisleri listelemek için netstat komutu kullanılabilir.

```
# netstat -antlegrep 'udplLISTEN'
tcp    0    0 127.0.0.1.587      *.*          LISTEN
tcp    0    0 127.0.0.1.25      *.*          LISTEN
tcp    0    0 *.22              *.*          LISTEN
tcp    0    0 *.37              *.*          LISTEN
tcp    0    0 *.13              *.*          LISTEN
tcp    0    0 *.113             *.*          LISTEN
udp    0    0 127.0.0.1.512     *.*          LISTEN
udp    0    0 *.514             *.*          LISTEN
tcp6   0    0 ::1.587           *.*          LISTEN
tcp6   0    0 ::1.25            *.*          LISTEN
tcp6   0    0 *.22              *.*          LISTEN
tcp6   0    0 *.37              *.*          LISTEN
tcp6   0    0 *.13              *.*          LISTEN
tcp6   0    0 *.113             *.*          LISTEN
udp6   0    0 ::1.512           *.*          LISTEN
```

Kapatılmak istenen servisler /etc/rc.conf ve /etc/inetd.conf’ dan servis önüne # işareti koyarak kapatılabilir.

OpenBSD’yi Cisco benzeri bir kabuk aracılığı ile yönetmek

UNIX konsoluna alışkın değilseniz nsh(Network Shell) [<http://www.nmedia.net/~chris/nsh/>] kullanarak sistem yönetimi ile ilgili çoğu işi kolaylıkla halledebilirsiniz. NSh, cisco CLI benzeri bir arabirim sunar ve ezberlemeniz/bilmeniz gereken yüzlerce komutu hatırlatarak system yönetiminin kolaylaştırır. Daha önce Cisco ya da benzeri bir CLI kullandıysanız komutlar oldukça tanıdık gelecektir.

NSH Kurulumu

```
#cd /tmp/  
#mkdir nsh  
#cd nsh  
#wget ftp://ftp.nmedia.net/pub/nsh/nsh-20050829.tar  
#cd nsh-20050829.tar  
#tar xvf nsh-20050829.tar  
#cd nsh  
#wget http://www.enderunix.org/docs/patches/nsh-openbsd39-honal.patch  
#patch < nsh-openbsd39-honal.patch
```

```
#!/nsh  
openbsd.huzeyfe.net/en  
openbsd.huzeyfe.net(priv)/?  
% Commands may be abbreviated.  
% Commands are:  
  
hostname    Set system hostname  
interface   Modify interface parameters  
bridge      Modify bridge parameters  
show        Show system information  
ip          Set IP networking parameters  
flush       Flush system tables  
enable      Enable privileged mode  
disable     Disable privileged mode  
route       Add a host or network route  
pf          Packet filter rule handler  
quit        Close current connection  
reload      Reboot the system  
halt        Halt the system  
write-config Save the current configuration  
verbose     Set verbose diagnostics  
editing     Set command line editing  
!           Invoke a subshell  
?           Print help information
```

```
openbsd.huzeyfe.net(priv)/pf ?
```

```
% pf edit  
% pf reload  
% pf enable  
% pf disable  
openbsd.huzeyfe.net(priv)/sh ?  
% Commands may be abbreviated.  
% 'show' commands are:  
  
hostname    Router hostname  
interface   Interface config  
route       IP route table or route lookup  
pfstats     PF statistics  
ipstats     IP statistics  
ahstats     AH statistics
```

espstats	ESP statistics
tcpstats	TCP statistics
udpstats	UDP statistics
icmpstats	ICMP statistics
igmpstats	IGMP statistics
ipcompstats	IPCOMP statistics
rtstats	Routing statistics
mbufstats	Memory management statistics
monitor	Monitor routing/arp table changes
ap	Wireless access points
version	Software information
running-config	Operating configuration
startup-config	Startup configuration
?	Options

Packet Filter'a Giriş

PF

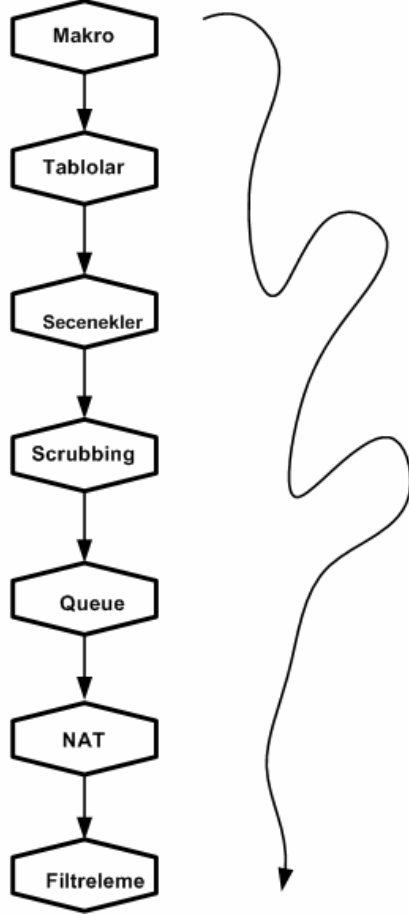
OpenBSD Packet Filter(PF), IPF'deki lisans değişikliği ve OpenBSD takımının yeni lisansı kabullenemez bulması ile ortaya çıkan ve kısa sürede diğer açık kod güvenlik duvarları arasında adından -haklı olarak – sözettiren bir güvenlik duvarı yazılımı. İlk olarak OpenBSD işletim sistemi ile kullanılabilen PF zamanla diğer BSD sistemlerine de uyarlanmıştır. PF'i OpenBSD ile kullanmanın avantajı : OpenBSD base sistemle birlikte gelmesi (OpenBSD sistemi kurduktan sonra PF'i tüm özellikleri ile kullanılabilir durumdadır) ve OpenBSD ile sorunsuz kullanılabilen bazı özelliklerin diğer *BSD sistemlerde tam kararlı çalışmaması olarak sayılabilir.

OpenBSD PF'in güvenlik duvarı olarak sağladığı özellikler piyasada bulunabilecek herhangi bir güvenlik duvarından oldukça farklıdır. Bu yönü ile hem ticari hem de özgür yazılımlar arasında parmak ile gösterilebilecek bir konuma sahiptir.

OpenBSD PF ile diğer özgür güvenlik duvarı yazılımları arasındaki temel farklıklara gözatmak için EnderUNIX ekibi tarafından hazırlanan "Açık kod Güvenlik Duvarları Karşılaştırması" (http://www.enderunix.org/docs/fwcomparetbl_trv01.pdf) tablosu incelenebilir. OpenBSD ile birlikte PF'in günümüz güvenlik duvarı terminolojisi kavramları kullanılarak inceleme haline http://www.enderunix.org/docs/openbsd_firewall.pdf adresinden erişim sağlanabilir.

Kural yönetim dosyası ve düzeni

OpenBSD PF pf.conf – isleyis sirasi



OpenBSD PF, güvenlik duvarı kurallarını pf.conf dosyasından belirli bir sıra ile okuyarak işleme sokar. Bu dozen yandaki gibidir.

Makrolar

Makrolari programlama dillerindeki deęişkenlere benzetebiliriz, kullanılmadan önce mutlaka tanımlanmaları gerekir. Mesela internet tarafına bakan ağ arabirimini \$ext_if="fxp0" olarak tanımlarsak bundan sonra fxp0 kullanacağımız her yerde \$ext_if tanımını kullanabiliriz. Ya da birden fazla portu tek bir makroda tanımlamak istersek liste özelliğini kullanabiliriz

```
yasak_portlar="21 23 80 13 17 135 139 137 445 "
```

gibi.

```
block in on $ext_if proto tcp from any to any port $yasak_portlar
```

kuralı ile tüm portları tek seferde yasaklamış oluyoruz.

Kuralları yazarken yapılacak yanlışlıklar kurallar yüklemeye çalışılırken ekrana basılacaktır. Mesela filtreleme kurallarını NAT kurallarından önce yazıp kuralları yüklemeye çalışılırsa aşağıdaki gibi bir hata verecektir. Pf.conf'taki kuralları yüklemek için pfctl komutu kullanılır.

pfctl -f /etc/pf.conf

```
/etc/pf.conf:18: Rules must be in order: options, normalization, queueing, translation, filtering
```

```
pfctl: Syntax error in config file: pf rules not loaded
```

Hata çok açık bir şekilde Pf kural sıralamasında yanlışlık olduğunu belirtiyor. Yine tanımlamalarda yapılacak hatalar(makrolar) da ekranda açık bir şekilde gözükecektir.

NOT:pf.conf'un bu düzeni kontrol etmesi istenmiyorsa options bölümünde *require-order no* seçeneği belirtilmiş olmalıdır

Tablo Kullanımı

Tablolar, basitçe IP adresi gruplarıdır. Eğer bir kural için belirli sayıdan fazla IP adresi gerekiyorsa bu IP adresleri için tek tek kural yazmak yerine tablo olarak tanımlayarak PF'in daha performanslı çalışmasını sağlayabiliriz. Bu adresleri makro olarak tanımlayıpda kullanabilirdik fakat makrolar ile tablolar arasında performans olarak oldukça fazla fark vardır.

Tabloların diğer bir özelliği de firewall çalışırken tablolara ek eleman eklenebilmesi. Mesela saldırgan isimli bir tablomuz olsun, bu tabloya sisteme eş zamanlı 50den fazla bağlantısı olan Ipleri eklemek isteyelim (bu işi bizim yerimize yapan bir script yazdığımızı farzediyorum) Scriptimiz dakika başı çalışarak fazla bağlantı kurmaya çalışan Ipleri belirleyerek saldırgan tablosuna eklesin. Tüm bu eklemeler anında işleme sokulur ve ekleme sırasında güvenlik duvarını yeniden başlatılmasına gerek yoktur.

NOT: Yukarıda verilen bu örnek, tabloların kullanımına gerek duymadan `max-src-conn-rate` kullanarak da yapılabilir.

NOT: Tablolar sadece IP adresleri için kullanılır. Makrolar ise diğer tür bilgileri de tutabilir.

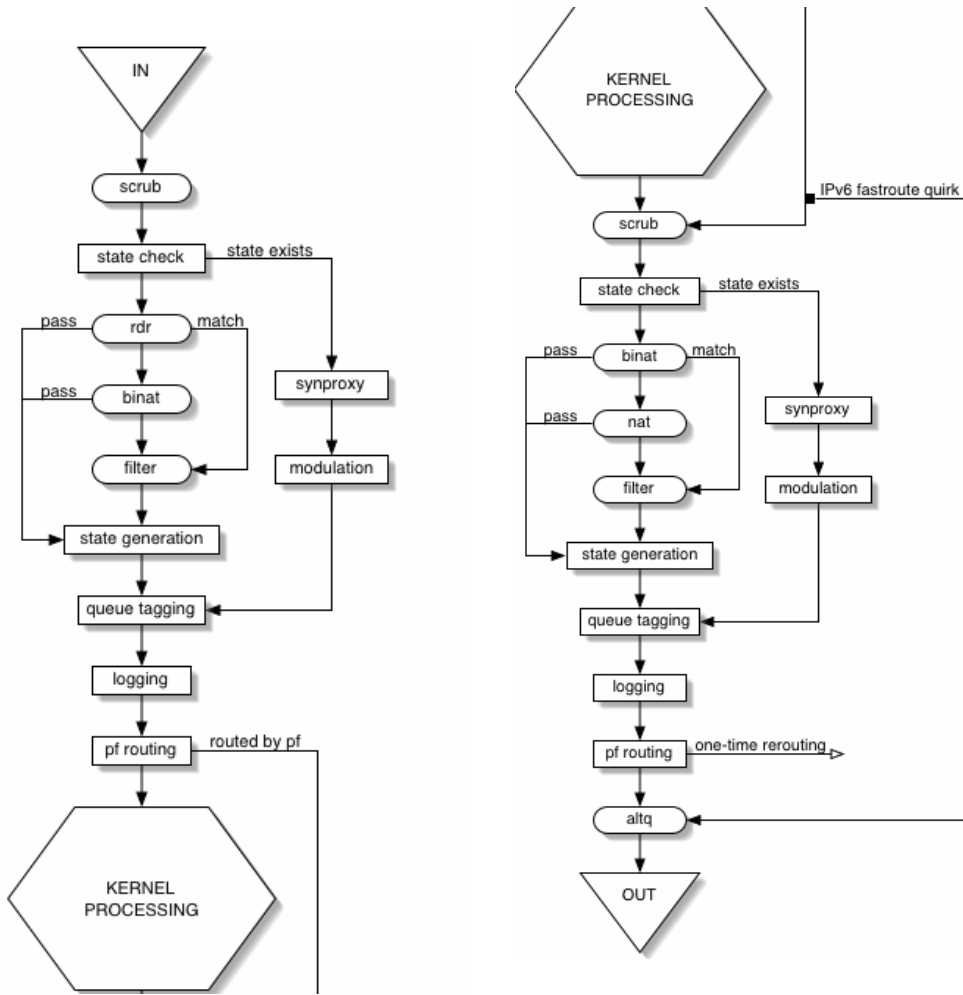
Tablo Tanımlama ve Kullanımı

```
Table <spamciler> persist {1.2.3.0/24, 4.5.6.0/24}
Table <yasaklilar> persist file "/etc/yasaklilar"
```

Tanımladığımız bu tabloları filtreleme kurallarında aşağıdaki gibi kullanabiliriz;

```
Block in on $ext_if proto tcp from <spamciler> to ($ext_if:0) port 25
Block out on $ext_if proto tcp from <yasaklilar> to any
```

PF Çalışma Mantığı



Kaynak: <http://homepage.mac.com/quension/pf/flow.png>

Güvenlik duvarı olarak PF'in işleyişi hakkında ne kadar bilgimiz olursa yapacağımız tasarımlar ve yazacağımız kurallar o kadar kusursuz olacaktır. İşleyiş yapısından kastım paketler nerede PF'e teslim ediliyor, PF gelen/giden paketleri nasıl işliyor ve nereye teslim ediyor gibi soruların cevabı aslında.

Yukarıdaki şekli inceleyecek olursak aslında PF'in genel çalışma mantığı hakkında ön bilgi ediniriz. PF, arabirimlere bağımlı olarak kuralları işleten bir güvenlik duvarıdır, yani kuralları arabirimlere bağımlı yazarsınız (istenirse arabirimden bağımsız da yazılabilir), **şu arabirimden** gelip **şu IP adresinin X. portuna** giden paketleri yasakla şeklinde. Burada izlenen yolun aslında performans olarak çok yararı vardır. Zira PF gelen bir pakete baktığında ilk olarak arabirim karşılaştırması yapar ve eğer gelen paket kuraldaki arabirime uymuyorsa o kuralı direk geçer. Örnekleyecek olursak;

Pass in on \$ext_if proto tcp from any to any port 80 keep state
Pass in on \$dmz_if proto tcp from any to any port 22 keep state

Pass in on \$adsl_if proto tcp from any to any port 110 keep state

.....

Pass in on \$int_if proto tcp from any to any port 80 keep state

Yukardaki gibi onlarca satır farklı kuralımız olduğunu düşünelim, iç ağdan(Firewallun iç ağa bakan bacağının \$int_if olduğu varsayılmıştır)bir paket geldiğinde PF bunu tüm kurallardan geçirmek zorundadır. Paket Güvenlik duvarına geldi, ilk kurala baktı \$ext_if gördü direct geçti, ikinci kurala baktı yine \$int_if göremedi onu da geçti.. ta ki arabirim olarak \$int_if'e rastlayana kadar. Böylece PF kuralı işletmeden es geçmiş oluyor.

PF'in işleyiş yapısı ile ilgili dikkate değer diğer bir husus, paketlerin öncelikli olarak scrub işleminden geçmesi(uygunsuz paketler direkt düşürülüyor) ve sonra durum kontrolü yapılması ve daha sonra nat, filtreleme vs işlemlerine tabi tutulması. Eğer kurallarımızı yazarken durum korumalı yazdıysak (keep state vs) PF gelen paketleri önce bir durum tablosu(state table, pfctl -s state ile bakılabilir)ile karşılaştırır ve duruma uymayan paketleri firewallun normal süreçlerine gönderir.

4.0 sonrası OpenBSD sürümlerinde "keep state" tanımı öntanımlı olarak gelecektir. Yani artık keep state yazmak zorunda kalmadan kurallarımız stateful(durum korumalı) olacak.

Paket Normalleştirme

Paket normalleştirme OpenBSD PF'in sağladığı en güçlü özelliklerden biridir. Diğer birçok güvenlik duvarında port taramaları, işletim sistemi saptalamalarını engellemek, parçalanmış paketleri(fragmented packets) birleştirmek/engellemek için satırlarca kural yazmak gerekirken OpenBSD PF ile bunları tek satırda halletmek mümkün.

Tüm bu korumalar pf.conf'a yazılacak scrub in all satırı ile yapılabilir. Ne kadar kolay değil mi?

Scrub in all

Eğer bu korumayı sadece belirli bir arabirimden gelen paketler için yapmak istersek scrub kuralını sadece o arabirime işletecek şekilde yazmamız gerekir.

Scrub in on \$int_if

Scrub'in sağladıkları özellikler bunla sınırlı değildir. Mesela bir makinenin/işletim sisteminin TTL değerini değiştirebilir, TCP başlığındaki MSS değerleri ile oynayabilir fragmented(parçalanmış) olarak gelen paketleri bütün hale getirip filtreleme kurallarına birleştirilmiş halde sokabilir.

Sıkı bir güvenlik için

scrub on \$ext_if all reassemble tcp

yeterlidir.

Scrub tanımı ile ilgili detay bilgi için **man pf.conf** çıktısına başvurulabilir.

Paket Filtreleme

Paketler güvenlik duvarına ulaştığında PF filtreleme mekanizmasında alınabilecek 3 aksiyon vardır. Bunlar paketi yasaklamak(block), pakete izin vermek(pass) ve paketin geldiği kaynağı doğrulamak(antispoof*).

Bir güvenlik duvarı öğrenilirken dikkat edilesi ilk hususlardan biri kuralların nasıl işlediğidir. Çoğu güvenlik duvarı kuralları işletirken ilk uyan kazanır politikasını uygular. Yani bir paket geldiğinde kurallar baştan aşağı control edilir ve ilk uyan kural paketin kaderini belirler.

Piyasada bulunan birçok güvenlik duvarının aksine PF' de tersi bir durum söz konusudur :*son uyan kazanır politikası*. Bir paket geldiğinde tüm kural ailesi control edilir. Bu yöntemin çeşitli artı ve eksileri vardır. Bu şekilde farklı bir politika uygulamasına rağmen istenirse ilk uyan kazanır tipi bir politika da belirlenebilir. Bunun için pass ya da block ifadesinden sonra quick kelimesi kullanılır. Quick kelimesi ile eğer paket bir kurala uyuyorsa sonraki kurallara bakılmaması gerektiğini söylemiş oluruz.

Filtreleme için kullanılacak sözdizimi;

```
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]
```

Örnek,

```
pass in log on fxp0 inet proto {udp tcp } from any to $DNS_SUNUCU port 53 keep
state
```

Bu örnekte DNS_SUNUCU olarak tanımlanan hosta gelen tcp/udp 53 paketleri kabul ediliyor ve gelen paketlere dönecek cevaplar için de durum tablosu tutularak izin verilmiş oluyor.

Örnek;

```
block in on r10 inet proto tcp from any to any port ssh
```

Yasaklamak istediğimiz paketlerde istersek pakete nasıl davranacağımızı da belirtebiliriz. Yasaklanacak paketin yasaklandığına dair herhangi bir uyarı gönderecek mi yoksa sessizce paketi yasaklayacak mı? (drop / return-rst)

```
block return-rst in log (all) on r10 inet proto tcp from any to any
port ssh
```

Block return-rst kullanarak yasakladığımız paketlerde geriye RST cevabı döndürebiliriz. Böylece paketi gönderen portun kapalı olduğunu ve bir güvenlik duvarı tarafından korunduğunu anlamayacaktır. Return-rst sadece TCP protokolü içindir, UDP protokolü için sadece return kullanılması yeterlidir.

In/Out Kavramı

PF ile kural yazarken en sık yapılan hata paketin yönüdür. Eğer paket güvenlik duvarına doğru geliyorsa in olarak adlandırılır, eğer güvenlik duvarından çıkıyorsa out olarak adlandırılır.

İÇ_ağ ↔ Firewall ↔ İnternet

İç_ağdan bir paket internete giderken öncelikle güvenlik duvarının iç bacağına gelecektir(IN) sonra dış bacağından internete çıkacaktır (OUT). Aynı şekilde dönen paket internette gelirken güvenlik duvarının dış bacağına gelecektir(in) sonra iç ağa geçecektir(out)

Durum Korunmalı filtreleme kuralları yazmak

PF ile yazdığımız kurallarda öntanımlı olarak durum koruması yapılmaz. Yani bir pakete çıkış izni verdiğimizde aynı paket için bir de giriş izni vermemiz lazım. Bunun yerine PF'in durum koruma özelliğini kullanarak giden ya da gelen paketlere durum koruması ekleyerek tek bir kural ile paketin hem gidişine hem de gelişine izin verebiliriz.

```
pass out on $ext_if proto TCP all keep state
```

gibi. Burada anahtar kelime keep state. Bunu filtreleme kuralımızın sonuna eklediğimiz zaman durum korunmalı kural yazmış oluyoruz. Keep state'den başka benzer özellikler sunan (daha gelişmiş seçenekler) modulate state ve synproxy state özellikleri de kullanılabilir.

TCP Bayraklarına Göre Filtreleme

TCP bağlantılarında bağlantının her adımı bayraklar ile control edilir. Yani bir bağlantının başlaması, veri akmasını, bağlantının bitirilmesi adımları tamamen bayraklar kontrolünde yapılır. Bir TCP bağlantısının başlaması için SYN bayrağı ayarlanmış paket kullanılmalıdır. PF TCP bağlantılarında bayraklara göre filtreleme seçenekleri sunar. Akla hemen bunun ne yararı olur gibisinden bir soru gelebilir.. Basitce bir cevap vermek gerekirse TCP bayrakları ile oynayarak düzenlenmiş çeşitli saldırılar kolaylıkla engellenebilir...

```
Pass in on $ext_if proto tcp from any to $ext_if port 80 flags S/SA keep state
```

Bu kural ile gelen paketlerde sadece SYN ve ACK bitlerine bakıp (diğer bayraklar gözardı ediliyor) sadece SYN bayrağı aktif olan paketler kabul ediliyor.

Yorum: Öyle ya bir TCP oturumunun başlaması için sadece SYN paketi set edilmiş paketler kullanılması gerekir. SYN ile birlikte set edilmiş diğer bayraklar ya kötü bir amaçla yapılmıştır ya da çalıştırılan yazılımın bir .eksikliğidir. Her iki durum da TCP oturumu için kabul edilemezdir ve engellenmelidir.

Loglama

Filtreleme yapılırken kullanılacak önemli seçeneklerden biri de log kelimesidir. Belirttiğimiz filtreleme kuralına uyan paketi loglayarak herhangi bir problem durumunda sorunu kolayca bulabiliriz ya da ara ara istatistik bilgileri almak istersek loglamayı aktif edebilirsiniz.

Log seçeneği ile yazılmış örnek filtreleme kuralı::

```
pass in log on fxp0 inet proto {udp tcp } from any to $DNS_SUNUCU port 53 keep state
```

Bu kuralda kullanılan log kelimesi duruma uyan(keep state kullanarak durum korumalı kural yazmış olduk) uyan ilk paket loglanıyor, duruma uyan tüm paketler loglanmak istenirse log (all) kullanılabilir.

```
pass in log (all) on fxp0 inet proto {udp tcp } from any to $DNS_SUNUCU port 53 keep state
```

HATIRLATMA: Firewall kurallarınızda statefull inspection(durum koruma ..) özelliği kullanıyorsanız PF sadece duruma uyan ilk paketi loglayacaktır diğer paketleri es geçecektir, eğer duruma uyan tüm paketleri loglamak isterseniz log yerine log (all) kullanabilirsiniz.

Nat işlemleri

NAT, basitçe bir Ip adresinin internete olduğu gibi değilde farklı bir IP adresi üzerinden çıkması/erişilmesi işlemidir. NAT kullanımının çeşitli sebepleri olabilir , kimi zaman IP yetersizliği kimi zaman güvenlik. Çeşitli firmaların çıkardığı ürünlerde farklı NAT tanımları var kimi hide Nat, static nat, kimi MIP, DIP, VIP kimi de server publishing gibi isimlerle NAT çeşitlerini adlandırıyorlar. Temel olarak 2-3 çeşit nat vardır denilebilir. İç ağ kullanıcılarının internete bir ya da birden fazla IP üzerinden çıkması ve iç ağımızda private adres grubundan IP adresine sahip sunucularımızın internet üzerinden public iplerle erişilebilir olması. Bu iki ana NAT yapısına bağlı kalarak çeşitli nat tanımları üretilebilir.

OpenBSD PF ile 3 çeşit NAT işlemi gerçekleştirilebilir. Bunlar;

- NAT/Masquerading
- RDR
- BINAT

NOT: NAT kullanabilmeniz için işletim sisteminin ip forwarding özelliği aktif olmalıdır. OpenBSD için bu özellik

#sysctl net.inet.ip.forwarding=1

Komutu ile aktif edilebilir.

NAT

Güvenlik duvarı arkasındaki birden fazla makinenin tek (ya da birden fazla) IP üzerinden internete çıkması için kullanılır.

Pf.conf için NAT kural sözdizimi

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

Nat on fxp0 from 192.168.1.0/24 to any -> 212.156.67.222

192.168.1.0/24 ağından gelip herhangi bir yere giden tüm paketleri fxp0 üzerinden çıkarken kaynak IP adreslerini 212.156.67.222 –kaynak portlarını da değiştirerek- yap ve gönder.

Eğer internet bağlantınız DHCP ya da benzeri bir sistemden otomatik IP adresi alıyorsa bu NAT kuralı sağlıklı çalışmayacaktır, NAT kuralının değişen IP adresine göre kendisini yenilemesini istersek nat kuralını aşağıdaki gibi yazmamız gerekir.

```
Nat on fxp0 from 192.168.1.0/24 to any -> (fxp0)
```

Pfctl -s state komutu ile state tablosuna bakarsak hangi IPden nereye giderken NAT yapılmış görebiliriz.

```
all tcp 192.168.1.2:1769 -> 14.7.7.1:54620 -> 212.156.67.222:5900 ESTABLISHED:ESTABLISHED
```

Hariç Tutma

Tüm ağı NAT ile internete çıkarmak için kuralımızı yazdık fakat aradan bazı IP adreslerinin NAT'a uğramamasını istiyoruz, bu durumda no nat ifadesi ile istediğimiz IP adreslerini NAT'dan muaf tutabiliriz. Mesela 192.168.1.3 IP adresini NAT'dan muaf tutmak için aşağıdaki kural kullanılabilir.

```
No nat on fxp0 from 192.168.1.3 to any
```

RDR(Redirection)

Redirection basitçe port yönlendirmedir. Birden fazla sunucunuz varsa ve internete tek bir IP adresi üzerinden hizmet vermek istiyorsanız ya da bu bir zorunluluk durumu ise RDR kullanarak istediğiniz servis portunu istediğiniz IP adresine yönlendirebilirsiniz.

Mesela internete bakan tek bir IP adresimiz var ve iç ağımızda farklı makinelerde www, ftp ve smtp sunucularımız olsun. Bu 3 farklı servisi internete açmak istersek PF'te şu şekilde bir kural tanımlamamız gerekir.

```
Dis_ip="1.2.3.4"  
Web_ip="2.3.4.5"  
ftp_ip="3.4.5.6"  
smtp_ip="4.5.6.7"  
  
rdr on fxp0 proto tcp from any to $dis_ip port 80 -> $web_ip port 80  
  
rdr on fxp0 proto tcp from any to $dis_ip port 21 -> $web_ip port 21  
  
rdr on fxp0 proto tcp from any to $dis_ip port 25 -> $web_ip port 25
```

BINAT – Statik yönlendirme

BINAT çift yönlü çalışan bir NAT şeklidir. Binat kuralı ile güvenlik duvarı arkasındaki bir makine dışarıya tek bir IP adresi ile çıkar ve dışarıdan bu IP adresi ile erişilebilir. NAT ve RDR ikilisi kullanılarak bu tip gereksinimler karşılanabilir olsa da bir gün işinize yarayabilir.

Temel kullanımı;

```
Huzyfe="192.168.1.199"  
Huzi_Dis_ip="1.2.3.4"  
binat on fxp0 from $huzyfe to any -> $huzi_dis_ip
```

Binat için dikkat edilecek husus: tcp/udp portların değişmediğidir.

Yük dengeleme

Yoğun bir servisiniz var ve bu yükü birden fazla sunucuya dağıtmak istiyorsunuz. Bu sunucuların önüne bir adet PF koyarak gelen trafiği bu üç sunucu arasında dengeleyebilirsiniz.

```
web_servers = "{ 10.0.0.10, 10.0.0.11, 10.0.0.13 }"
```

```
rdr on $ext_if proto tcp from any to any port 80 -> $web_servers \  
    round-robin sticky-address
```

Kuralı ile güvenlik duvarına gelen 80. port istekleri arkadaski 3 sunucuya round-robin algoritması kullanılarak dağıtılıyor.

NAT ve Filtreleme ilişkisi

Hatırlanacak olursa PF'in kurallarını belirli bir sırada işleme soktuğunu belirtmiştik. Bahsedilen sıralamada NAT kuralları filtreleme kurallarından önce gelmektedir. Buna göre nat yapılan bir IP adresi aynı zamanda filtreleme kurallarından da izin verilmelidir. Bu iki şekilde yapılabilir.

Birincisi nat kuralı yazarken nat kelimesinden sonra pass kelimesi eklemek, diğeri ise yazılan her nat kuralı için ek bir filtreleme kuralı yazmaktır. Dikkat edilmesi gereken diğeri bir konuda NAT yapılmış paketlere izin verilirken NAT yapılmış iplere izin verilmesi gerektiğidir.

Örnek;

İç ağıımızda bir web sunucumuz olsun biz bu web sunucuya internetten de erişmelerini istiyoruz

Rdr ile ilgili bir kural(web sunucu)

```
Rdr on dis_bacak proto tcp from any to dis_bacak_ip port 80 -> web_sunucu_ip port 80
...
..
Pass in on dis_bacak proto tcp from any to web_sunucu_ip port 80
```

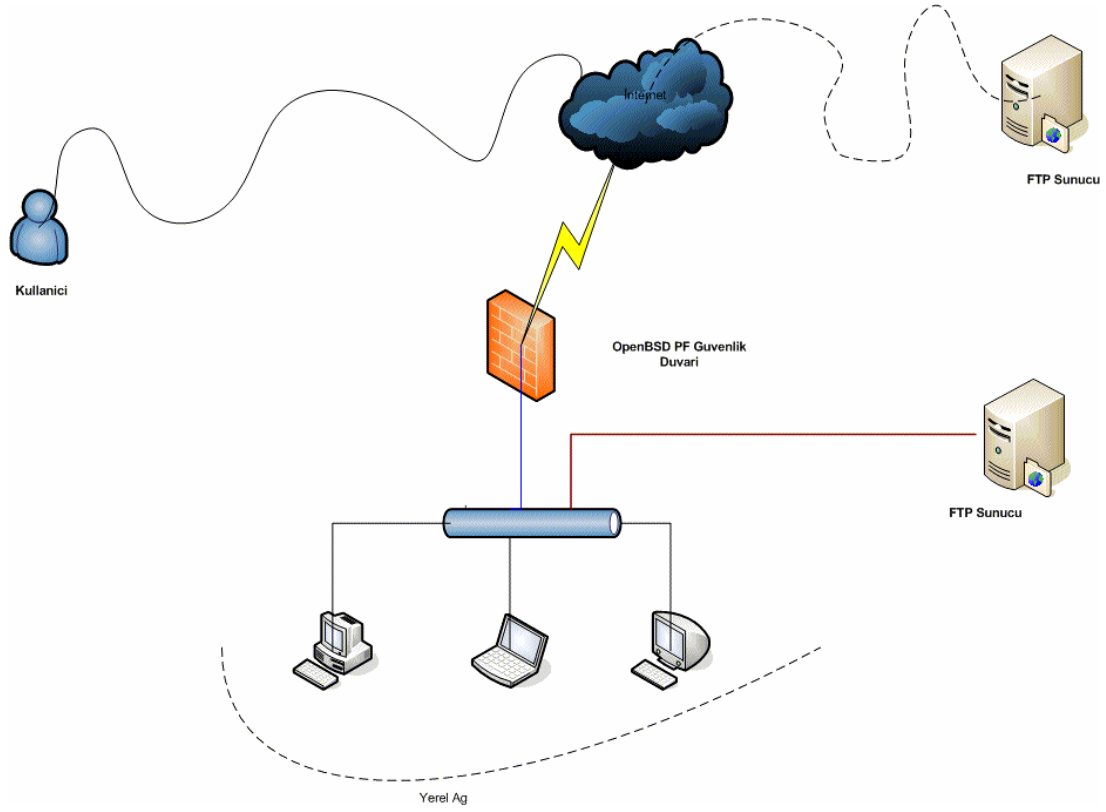
Eğer tek kuralda halletmek istersek rdr kuralına pass ifadesini eklememiz yeterli.

```
--
Rdr pass on dis_bacak proto tcp from any to dis_bacak_ip port 80 -> web_sunucu_ip
port 80
```

Packet Filter ve FTP

FTP TCP kullanan bir protokoldür fakat diğeri TCP kullanan protokollerden farklıdır. Normal protokoller tek port üzerinden çalışırken FTP iki farklı port üzerinden çalışır(IRC'deki veri transferi ve iletişim portu gibi). Bu portlardan biri Command port diğeri DATA port olarak geçer.

FTP iki çeşittir , Pasif ve Aktif FTP. Her ikisinde farklı amaçlı kullanımları mevcuttur. Hangi FTP çeşidinin kullanılacağı ftp istemcisi tarafından belirlenir.



Bir güvenlik duvarı'nın FTP ile ilişkisi iki şekildedir: Güvenlik duvarı iç ağ kullanıcılarının internete çıkması için kullanılıyordur ya da iç ağda/DMZ'de bulunan sunucuların korunması için kullanılıyordur. Aynı anda bu iki görevi birlikte yapabilir.

Aktif FTP

Pasif FTP

İç ağ kullanıcılarının ftp yapması için gerekli PF kuralları.

Pf.conf'un NAT kısmına aşağıdaki kurallar eklenir

```
nat-anchor "ftp-proxy/*"  
rdr-anchor "ftp-proxy/*"  
rdr on $int_if proto tcp from any to any port 21 -> 127.0.0.1 \  
port 8021
```

filtreleme kuralları kısmına

```
anchor "ftp-proxy/*"  
pass out proto tcp from $proxy to any port 21 keep state
```

eklenir.

/etc/rc.conf.local dosyasına ftpproxy_flags="" satırı eklenmelidir.

Bunun haricinde istemcilerin aktif FTP kullanmaları da isteniyorsa ftp-proxy uygulaması -r parametresi ile çalıştırılmalıdır.

Güvenlik duvarı arkasındaki FTP sunucuların dışarıdan erişilebilmesi için gerekli İşlemler

Güvenlik duvarının internete bakan IP adresi 1.2.3.4 olsun, güvenlik duvarı arkasındaki FTP sunucunun IP adresi de 4.5.6.7 olsun, bu durumda FTP hizmetinin sağlığı verilebilmesi için yapılması gerekenler;

/etc/rc.conf ya da /etc/rc.conf.local dosyasına aşağıdaki satırı ekleyin

```
ftpproxy_flags="-R 4.5.6.7 -p 21 -b 1.2.3.4"
```

/etc/pf.conf'a eklenmesi gereken kurallar

```
ext_ip = "1.2.3.4"
ftp_ip = "4.5.6.7"

nat-anchor "ftp-proxy/*"
nat on $ext_if inet from $int_if -> ($ext_if)
rdr-anchor "ftp-proxy/*"

pass in on $ext_if inet proto tcp to $ext_ip port 21 \
  flags S/SA keep state
pass out on $int_if inet proto tcp to $ftp_ip port 21 \
  user proxy flags S/SA keep state
anchor "ftp-proxy/*"
```

Aktif FTP hizmetinin güvenlik duvarı aracılığı ile çalışabilmesi için FTP sunucu'nun 20. portundan ANY'e izin verilmesi gerekir.

NOT: Güvenlik duvarınız hem FTP hizmeti veren bir sunucuyu koruyorsa hem de iç ağ kullanıcılarının internetteki FTP sunucularına erişmesi için kullanılacaksa OpenBSD üzerinde iki farklı ftp-proxy uygulaması çalıştırılmalıdır. Çalışacak ftp-proxy proseslerinin dinleyeceği portlar aynı olacağı için birini farklı porttan çalıştırmak durumundasınız.(-p port_numarasi seklinde)

Packet Filter Loglama

Ateş duvarının paketler üzerinde yaptığı işlemler loglar aracılığı ile izlenebilir. Günümüzde loglama mekanizması olmayan bir güvenlik duvarı düşünülemez. Güvenlik duvarında beklenmedik bir durum ile karşılaşıldığında yapılması gereken ilk işlem logları kontrol ederek gerçekten paketlerin istenildiği gibi işlenip işlenmediği görülebilir. Loglamanın diğeri bir yararı da çeşitli istatistikler çıkarılmasını sağlamasıdır.

OpenBSD PF loglama mekanizması pflogd(8) aracılığı ile çalışır. pflogd, pflog0 adlı sahte bir interface'i dinleyerek bu arabirim aracılığı ile paketleri loglar. Varsayılan log dosyası /var/log/pflog dır, bu dosyanın yeri/etc/rc.conf'ta

```
pflogd_flags="-f /var/log/yeni_log"
```

belirterek değiştirilebilir. Bu dosya normal bir text dosya değildir, loglama işleminin daha performanslı olması için tcpdump uyumlu ikili dosya olarak oluşturulur.

Loglama servisinin alacağı tüm parametreler için manuel sayfası okunmalıdır.

\$man pflogd

Log dosyalarını okuma

Log dosyalarının normal text editörleri aracılığı ile okunamayacağından bahsetmiştik , pf log dosyalarını okuyabilmek için genellikle tcpdump kullanılır. tcpdump aracılığı ile /var/log/pflog dosyasını okutursak PF'in logladığı paketleri görme imkanına sahip oluruz.

```
# tcpdump -tttn -e -r /var/log/pflog.0
```

```
tcpdump: WARNING: snaplen raised from 96 to 116
Jan 01 09:00:03.005228 rule 2/(match) pass out on tun0: 88.234.94.221.44246 >
195.175.39.39.53: 44262+[ldomain]
Jan 01 09:00:03.320052 rule 2/(match) pass out on tun0: 88.234.94.221.27390 >
195.175.39.40.53: 44262+[ldomain]
Jan 01 09:00:13.529378 rule 1/(match) pass out on rl0: 192.168.0.1 > 192.168.0.14:
icmp: 192.168.0.1 udp port 1900 unreachable
Jan 01 09:00:26.746354 rule 1/(match) pass in on rl0: 192.168.0.14.1050 >
207.46.111.79.1863: P 2634792597:2634792602(5) ack 1031604003 win 63852 (DF)
Jan 01 09:00:26.966561 rule 1/(match) pass out on rl0: 207.46.111.79.1863 >
192.168.0.14.1050: P 1:9(8) ack 5 win 65005 (DF)
Jan 01 09:00:27.127067 rule 1/(match) pass in on rl0: 192.168.0.14.1050 >
```

```
207.46.111.79.1863: . ack 9 win 63844 (DF)
```

fakat bu yöntem anlık logları görmemizi sağlamaz, anlık logları inceleyebilmek için tcpdump ile pflog0 arabirimi dinlememiz gerekir.

```
# tcpdump -i pflog0 -tttnn -e
```

```
tcpdump: WARNING: pflog0: no IPv4 address assigned
tcpdump: listening on pflog0, link-type PFLOG
Jan 03 22:48:40.194140 rule 1/(match) pass in on rl0: 192.168.0.14.1320 >
192.168.0.1.22: [tcp] (DF)
Jan 03 22:48:40.206722 rule 1/(match) pass out on rl0: 192.168.0.1.22 >
192.168.0.14.1320: [tcp] (DF) [tos 0x10]
Jan 03 22:48:40.207020 rule 1/(match) pass out on rl0: 192.168.0.1.22 >
192.168.0.14.1320: [tcp] (DF) [tos 0x10]
..
192.168.0.14.1329: [tcp] (DF)
Jan 03 22:48:40.423130 rule 1/(match) pass in on rl0: 192.168.0.14.1329 >
216.231.63.58.80: [tcp] (DF)
Jan 03 22:48:40.568913 rule 1/(match) pass out on rl0: 216.231.63.58.80 >
192.168.0.14.1307: [tcp] (DF)
Jan 03 22:48:40.572027 rule 1/(match) pass in on rl0: 192.168.0.14.1307 >
216.231.63.58.80: [tcp] (DF)
```

Yukarıdaki örneklerde tcpdump'ın sadece gerekli parametreleri kullanılmıştır, tcpdump'ın detaylı kullanımı ile ilgili <http://www.enderunix.org/docs/tcpdump.htm> adresinden faydalanabilirsiniz. OpenBSD ile birlikte gelen tcpdump normal UNIX/Linux sistemlerle birlikte gelen versiyondan farklıdır ve PF loglarını analiz için ek parametreler içerir. Bunlardan bazıları;

rulenum num - kural numarası

action act -pakete uygulanan aksiyon pass ya da block olabilir

inbound - giriş paketleri için.

outbound - çıkış paketleri için.

Örnek;

```
# tcpdump -n -tt -i pflog0 outbound and action block and on em0
```

tcpdump: WARNING: pflog0: no IPv4 address assigned

tcpdump: listening on pflog0

komutu ile em0 arabirimi üzerinden çıkan paketler arasında bloklanmış olanları görebiliriz.

.

HATIRLATMA: OpenBSD ile birlikte gelen tcpdump, PF loglarını analiz için çeşitli ek özelliklere sahiptir.

Packet Filter Yönetim aracı (pfctl)

Kuralları Yükleme

Kural tablosunu Görüntüleme

```
#pfctl -s nat
nat on rl0 inet from 100.100.100.0/24 to any -> 194.27.72.100
nat on rl0 inet from 172.16.10.2 to any -> 194.27.72.88
rdr pass on rl0 inet proto tcp from 217.174.43.20 to 194.27.72.88 port = https -> 127.0.0.1 port 23
rdr pass on rl0 inet proto tcp from 217.174.43.20 to 194.27.72.100 port = https -> 127.0.0.1 port 23
rdr on rl0 inet proto tcp from any to 194.27.72.88 port = www -> 172.16.10.2 port 80
...
rdr pass on rl0 inet proto tcp from any to any port = ftp -> 172.16.10.2 port 21
rdr pass on rl0 inet proto tcp from any to any port = smtp -> 172.16.10.2 port 25
rdr pass on rl0 inet proto tcp from any to any port = pop3 -> 172.16.10.2 port 110
rdr pass on rl0 inet proto tcp from any to any port 55000:60000 -> 172.16.10.2 port 55000:60000
```

benzer şekilde filtreleme kurallarını görmek için pfctl -s rules, queue kurallarını görüntüleyebilmek için pfctl -s queue komutları kullanılabilir.

Görüntülenen kurallar ile ilgili detay bilgi – kural sıra numaraları, kurala uyan paket sayısı vs- almak isterseniz, yazdığınız komut -v, -vv parametrelerini eklemelisiniz.

Mesela filtreleme kurallarını detayları ile görmek istersek aşağıdaki kural işimizi görecektir.

```
#pfctl -s rules -vv
@0 pass in quick on tun0 inet from 200.200.200.2 to any keep state
 [ Evaluations: 14916192 Packets: 0      Bytes: 0      States: 0  ]
 [ Inserted: uid 0 pid 6457 ]
@1 pass out log quick on rl0 inet from 100.100.100.0/24 to any keep state
 [ Evaluations: 14916192 Packets: 0      Bytes: 0      States: 0  ]
 [ Inserted: uid 0 pid 6457 ]
...
...

@11 block drop in on rl0 inet from any to 194.27.72.100
 [ Evaluations: 160865 Packets: 28362 Bytes: 2365539 States: 0  ]
 [ Inserted: uid 0 pid 6457 ]
```

```
#pfctl -s state
all tcp 172.16.10.2:58634 -> 194.27.72.88:56516 -> 66.98.190.35:1982
ESTABLISHED:ESTABLISHED
all tcp 100.100.100.6:3401 -> 194.27.72.100:54738 -> 83.149.123.188:4321
ESTABLISHED:ESTABLISHED
all tcp 100.100.100.6:3575 -> 194.27.72.100:54481 -> 129.21.60.23:80
ESTABLISHED:ESTABLISHED
...
...
all udp 194.27.72.88:53 <- 213.161.151.3:52024 MULTIPLE:MULTIPLE
all tcp 100.100.100.6:2137 -> 194.27.72.100:65122 -> 207.46.111.51:1863
all tcp 100.100.100.6:2822 -> 194.27.72.100:65153 -> 66.249.91.83:443
TIME_WAIT:TIME_WAIT
all tcp 100.100.100.6:2823 -> 194.27.72.100:55567 -> 66.249.91.83:443
TIME_WAIT:TIME_WAIT
```

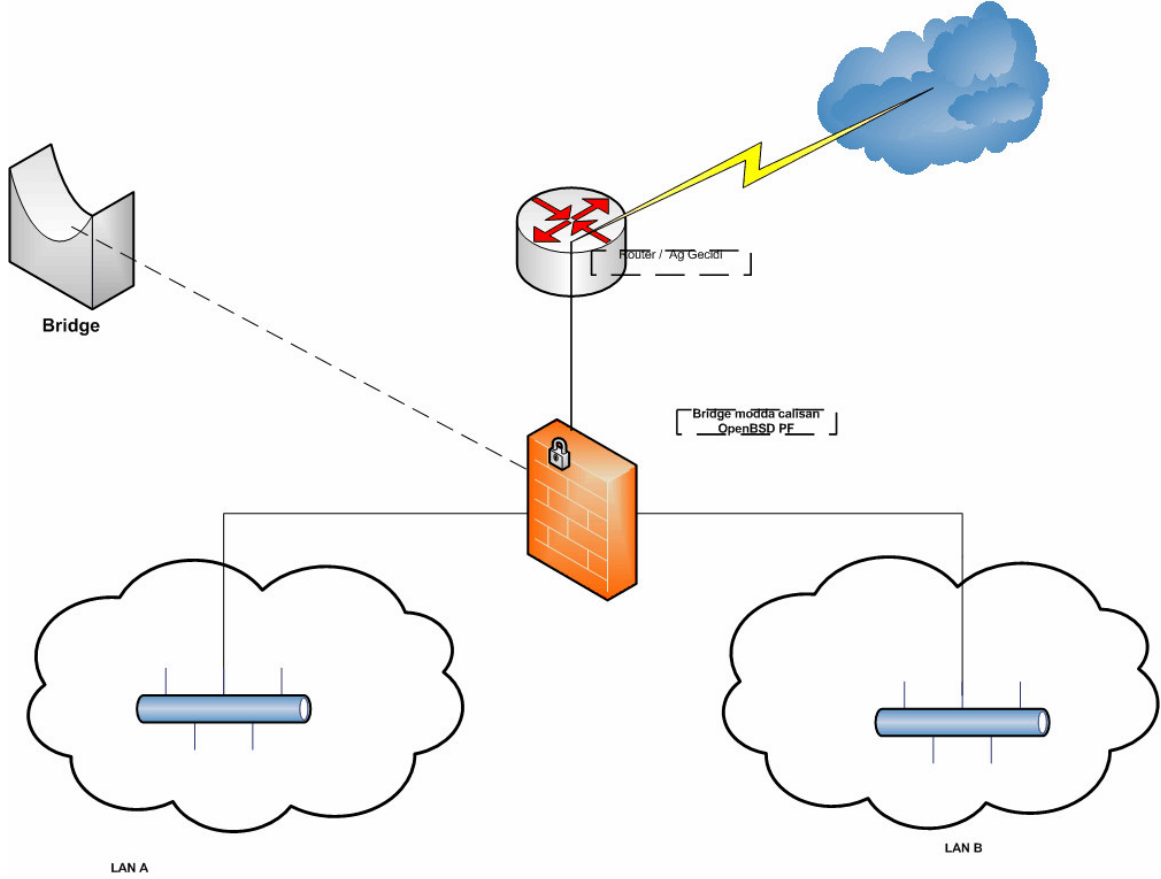
Man pfctl.conf komutu ile gerekli yardımı alabilirsiniz.

PF ile İleri Düzey işlemler

Görünmez Güvenlik Duvarı Kurulumu

Ağlar arası iletişimin yaygınlaşması ve bu yaygınlaşmaya bağlı olarak artan güvenlik riskleri sonucu Firewall'lerden daha farklı beklentiler ortaya çıkmıştır. Bu beklentilerden biri de Firewall'ların ikinci katmanda(görünmez modda) çalışmasıdır

İkinci katmanda çalışan bir Firewall'un getirisi neler olabilir? İlk olarak, katman 2 de çalışan bir firewall üzerinde herhangi bir geçerli IP adresi olmayacağı için firewall'u içeriden ve dışarıdan gelen saldırı risklerinden korumuş oluruz. Temel bir katman 2 Firewall'un yerleşimi aşağıdaki gibidir.



Bridge olarak kullanılan makinenin her iki tarafındaki ağ yapılandırmasının 172.16.10.X sekilde olduğu düşünülür.

Bridge yapılandırımı için brconfig(8) komutu kullanılır. İlk olarak bridge0 sahte arabirimi oluşturulmalıdır.

```
#vi /etc/hostname.rl0
up
#vi /etc/hostname.fxp0
up

# ifconfig bridge0 create
#brconfig bridge0 add fxp0 up
#brconfig bridge0 add rl0 up
#vi /etc/bridgename.bridge0

add fxp0
add rl0
up

#vi /etc/sysctl.conf

net.inet.ip.forwarding=1

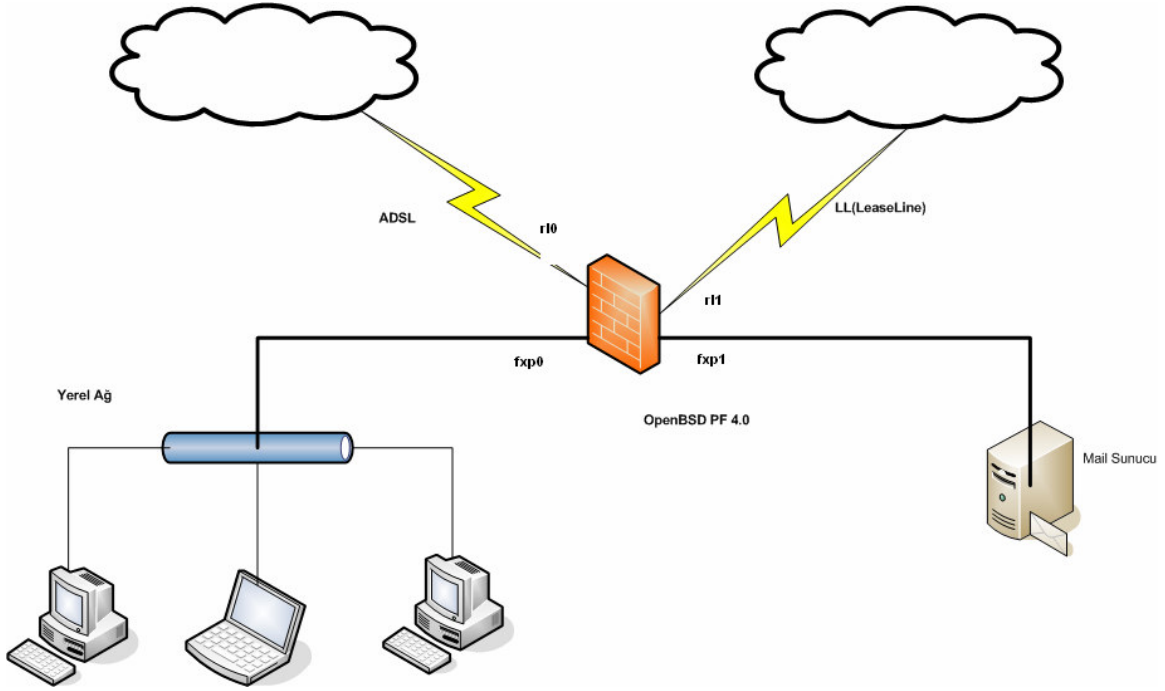
#ifconfig rl0 172.16.10.5
```

NOT: Istenildiği takdirde bridge üyesi olan arabirimlerin bir tanesine ya da bridge'in kendisine IP adresi atanabilir. Firewall'a 3. bir ethernet takma imkanı yoksa ve güvenlik duvarı ile bir şekilde iletişim kurmak isteniyorsa bu yöntem kullanılabilir.

Katman 2 güvenlik duvarını kurduktan sonra normal güvenlik duvarı kurallarını yazabiliriz. PF, katman 2 de çalışırken MAC adreslerine göre filtreleme de yapabilir..

Birden Fazla internet bağlantısını PF ile yönetme

Birden fazla internet bağlantınızı(adsl+ll, adsl+adsl vs)PF ile etkin bir şekilde kullanabilirsiniz. İstenirse iki(ya da daha fazla bağlantı) bağlantıyı tek bir bağlantı gibi düşünüp gelen istekleri bu iki çıkışa yönlendirebilirsiniz ya da belirli kullanıcıları bir hattan diğer kullanıcıları diğer hattan yönlendirebilirsiniz. Hatta bu yönlendirme işlemleri port, hedef vs bazı da yapılabilir. Web istekleri ADSL'den , mail istekleri LeasLine'dan çıksın gibi...



Birden fazla çıkışı olan bir yerel ağ

Güvenlik duvarı Kuralları

Tanımlamalar

```

ext_if_ll="r11"
ext_if_adsl="r10"
int_if="fxp0"
dmz_if="fxp1"
gw_ll="10.16.10.1"
gw_adsl="10.16.20.1"
ic_ag="200.200.0.0/24"
mail="100.100.100.3"
admin="200.200.0.15"
patron="200.200.0.15 200.200.0.222 200.200.0.223 "
sy_imac="200.200.0.5 200.200.0.189 200.200.0.190"
lo_if="lo"

```

RFC'lere uymayan paketleri gozardi et (SYN ile baslamayan vs. gibi)

scrub in all

####Mail makinesi LL uzerinden NAT olacak

nat on \$ext_if_ll from \$mail -> (\$ext_if_ll)

####Ic Ag kullanicilarin cikisini ADSLden ver

nat on \$ext_if_sl from \$ic_ag -> (\$ext_if_adsl)

REDIRECT

Mail Server Port Yonlendirmeleri - 25 => smtp 110 => pop3 143 => IMAP

rdr on \$ext_if_ll proto tcp from any to \$ext_if_ll port 25 -> 100.100.100.3 port 25

rdr on \$ext_if_ll proto tcp from any to \$ext_if_ll port 110 -> 100.100.100.3 port 110

rdr on \$ext_if_ll proto tcp from any to \$ext_if_ll port 143 -> 100.100.100.3 port 143

rdr on \$ext_if_ll proto tcp from any to \$ext_if_ll port 80 -> 100.100.100.3 port 80

rdr on \$ext_if_ll proto tcp from any to \$ext_if_ll port 443 -> 100.100.100.3 port 443

Ic Kullanicilarin www talepleri 8080'e atiliyor, Dansguardian icin

rdr pass log (all) on \$int_if proto tcp from \$ic_ag to any port 80 -> 127.0.0.1 port 8080

#####Ftp Protokolu icin

rdr pass log (all) on \$int_if proto tcp to port ftp -> 127.0.0.1 port 8021

HTTP taleplerini mail sunucuya at, webmail icin

rdr pass log (all) on \$ext_if_ll proto tcp to port www -> 100.100.100.3 port www

#####Filtreleme Kurallari

#Default herseyi blokla

block in log on \$ext_if_ll

block in log on \$ext_if_adsl

```
##### Loopback arabirimine izin veriliyor
Set-skip on lo

##### MSN MESSENGER PORTU'nu yasakla, 80 uzerinden calisanlar dansguardiana
block in log(all) quick on $int_if proto tcp from any to any port 1863

##### Disaridan gelen SSH istekleri kabul ediliyor
pass in log(all) on $ext_if_ll proto tcp to ($ext_if_ll) port 222 keep state
pass in log(all) on $ext_if_adsl proto tcp to ($ext_if_sl) port 222 keep state

##### Disaridan gelen www, Smtpl, Pop, imap istekleri kabul ediliyor
pass in log (all) on $ext_if_ll proto tcp from any to $mail port {25 80 110 143 443}
flags S/SA keep state

##### Admin'dan Mail Server'in SSH portuna izin veriliyor#####
pass in quick log (all) on $int_if proto tcp from $admin to $mail port 22 keep state
pass in quick log (all) on $dmz_if proto tcp from $mail port 22 to $admin keep state

##### Ic kullanicilarin Mail sunucuya POP3 SMTP IMAP erisimi veriliyor
pass in quick log (all) on $int_if proto tcp from $ic_ag to $mail port {25 110 143 443}
keep state
pass in quick log (all) on $dmz_if proto tcp from $mail port {25 110 143 443} to
$ic_ag keep state

#pass out quick log (all) on $dmz_if proto tcp from $mail to $ic_ag port {25 110 143 }
keep state

##### FTP icin ?
pass in log(all) on $ext_if_sl proto tcp to ($ext_if_sl) port > 49151 user proxy keep
state

##### Policy Based routing

#pass in quick log (all) on $lo_if route-to($ext_if_ll $gw_ll) from any to any keep state
```

```
pass in quick log (all) on $int_if route-to($ext_if_sl $gw_sl) from $ic_ag to any keep state
```

```
pass in quick log (all) on $dmz_if route-to($ext_if_ll $gw_ll) from $mail to any keep state
```

```
pass out log (all) on $ext_if_ll from $ext_if_ll to any keep state
```

```
pass out log (all) on $ext_if_sl from $ext_if_sl to any keep state
```

Trafik şekillendirme

Trafik şekillendirme PF'in en güçlü olduğu konulardan biridir. CBQ, HSBC gibi çeşitli algoritmalar kullanılarak paketler üzerinde çeşitli kapasite sınırlandırmaları yapılabilir. Bu konuda EnderUNIX ekibinden Bakır Emre 'nin hazırladığı [FreeBSD Üzerinde Pf ve ALTQ İle Trafik Şekillendirme](http://www.enderunix.org/docs/FreeBSDPfALTQ.pdf) <http://www.enderunix.org/docs/FreeBSDPfALTQ.pdf> belgesi oldukça doyurucu bir kaynak.

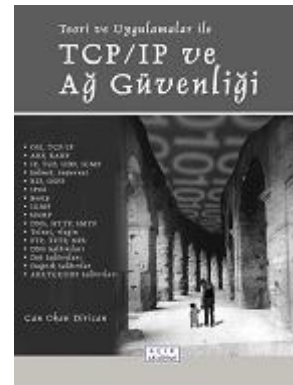
Sonuç

OpenBSD PF , tüm özellikleri ile doğru bir şekilde kullanıldığında ek bir yazılım gerektirmeksizin ağınızın güvenlik duvarı ihtiyacını en iyi şekilde karşılar. Eğer UNIX bilginiz yeterli değilse ve konsoldan çalışmak zor geliyorsa PF'i tüm özellikleri ile kullanamayacaksınız demektir. İnternette bulabileceğiniz çeşitli yönetim arabirimi programları ile PF'i UNIX konsoluna hiç bulaşmadan kullanabilirsiniz .

Bu şekilde bir kullanımın çeşitli yararları olduğu gibi eksik yönleri de vardır. Bir güvenlik duvarı sisteminin nasıl



çalıştığını anlamadan yönetmek silah eğitimi almadan bir atış yapmak gibidir, her an bacağınızı isabet alabilirsiniz. Bu sebeble tavsiyem bir güvenlik duvarını kullanmadan TCP/IP ve ağ güvenliği konularında bilgi edinmeniz. Bu konuda Açık Akademi(<http://www.acikakademi.com>) yayınlarından çıkan “TCP/IP



ve Ağ Güvenliđi” ve “Ađ Güvenliđi İpuçları” kitapları iyi birer başvuru kaynađı olarak kullanılabilir.

OpenBSD PF’i etkin bir şekilde kullanmak ve profesyonel destek almak isterseniz piyasada bu işi yapan çeşitli firmalar/şahıslar var. Bu belgenin yazarı da tamamen OpenBSD PF’e özel olmak üzere çeşitli eğitimler vermektedir. Konu ile ilgilenenler http://www.huzeyfe.net/openbsd_pf_egitimi.pdf adresinden detaylı bilgi edinebilir.

Kaynaklar

[1]<http://ipucu.enderunix.org>

[2]OpenBSd Belgeleme projesi(<http://www.enderunix.org/openbsd>)

[3]PF FAQ (<http://www.openbsd.org/faq/pf/>)