FreeBSD ve PF ile Güvenlik Duvarı

Murat ÜSTÜNTAŞ
murat.ustuntas@linux.org.tr
www.ustuntas.net

PF' nin Kısa Tarihi

- Daren Reed' in geliştirdiği IPFilter, OpenBSD işletim sistemleri üzerinde, Mayıs 2001' e kadar kullanılmıştı.
- Daren Reed' in IPFilter üzerindeki Lisanslama şekli, sonradan farkedilerek, OpenBSD geliştiricilerinin felsefesine uymadığı için, OpenBSD işletim sisteminden IPFilter çıkartılmıştır.
- Daniel Hertmeier, 2002' nin sonlarına doğru PF' yi geliştirerek, OpenBSD işletim sistemlerine entegre etmiştir.
- PF yapılandırılması itibari ile IPFilter' e çok benzediği için, IPFilter kullananlara büyük kolaylık sağlamıştır.
- IPFilter' den farklı olarak PF, çok daha esnek, kolay ve fazla özelliğe sahiptir.
- Şimdi, tüm BSD tabanlı işletim sistemleri tarafından desteklenmektedir.
- Diğer Unix aileleri için entegrasyon çalışmaları yapılmaktadır.

PF' ye Neden İhtiyacımız Var ?

- Ağ ve İnternet ortamlarında, güvenlik vazgeçilemez bir unsurdur.
- Ağ ve/veya İnternet üzerinde, ististismarlara karşı korunacak alanlar mutlaka vardır.
- Ağ ve/veya İnternet üzerindeki istismarlar, dışarıdan değil, kendi içimizden de gelmektedir.
- Güvenlik konusunda kaygılarınız varsa, Güvenlik Duvarına mutlaka ihtiyaç duyarsınız.
- Güvenlik Duvarı çözümleri içerisinde, bir arayış içerisine girersiniz. Lisans ücretleri olan veya olmayan çözümler içerisinde bir karara varmanız gerekmektedir...
- PF bu noktada tüm ihtiyaçlarınızı karşılayacak donanımlara sahiptir.

PF Donanımları

- IPv4 ve IPv6 paket Filitreleme
- Durum Kontrollü Paket Filitreleme (Stateful İnspection)
- Ağ Adres Çevirimi (Nat)
- Port Yönlendirme (Port Forwarding)
- Paket Normalizasyonu (Packet Normalization – Scrubing)
- Bant Genişliği Yönetimi (Bandwidth Management – ALTQ)
- Adres Havuzları (Address Pooling) / Yük Dengeleme (Load Balancing)
- Çoklu Erişim Yönlendirmesi (MultiHomed)
- Dinamik Kural Tanımlamaları
- Paket Kayıt Sistemi (Pflog)
- ... ve her geçen gün eklenen yeni özelikler.

-- http://www.openbsd.org/faq/pf/index.html

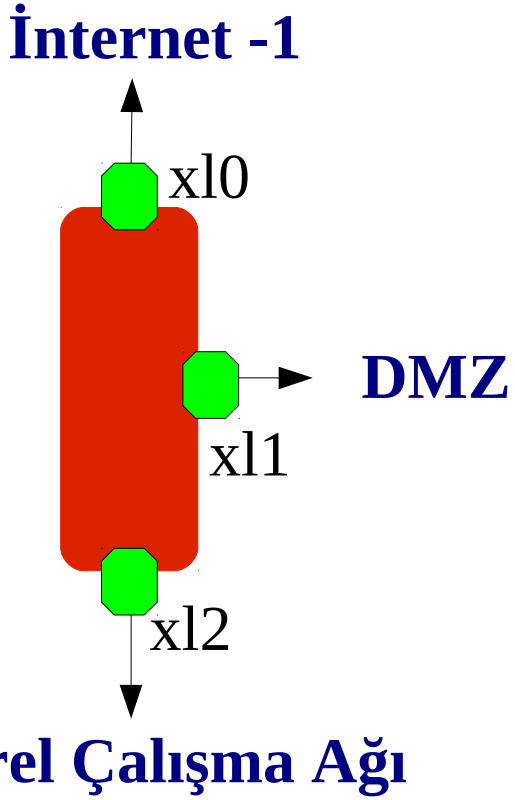
BSD tabanlı bir işletim sistemine ihtiyacınız olacak. Burada biz FreeBSD işletim sistemi üzerinde işlemlermizi anlatacağız. (OpenBSD içerisinde tümleşik olarak PF desteği bulunmaktadır.)

FreeBSD işletim sistemini, minimalist bir yaklaşımla kurmakta fayda var. Sadece “Güvenlik Duvarı” olarak tasarlanan sistemler için bu yaklaşım, diğer servislerden kaynaklanacak güvenlik açıklarının oluşmasını en aza indirecektir.

FreeBSD kurulumunu yaptığınız ve en azından ağ ayarları üzerinde biraz bilgi sahibi olduğunuzu düşünerek bundan sonraki aşamaları anlatacağız.

Kurduğumuz FreeBSD işletim sistemi üzerinde 3 adet ağ kartı bulunmaktadır. Bu ağ kartlarının tamamı 3Com (xl) olarak seçilmiştir.

Güvenlik Duvarı Ağ Bilgileri




```
defaultrouter = “aaa.bbb.ccc.ddd”
ifconfig_xl0 = “inet aaa.bbb.ccc.eee netmask 255.255.255.224”
ifconfig_xl1 = “inet 10.0.0.1 netmask 255.255.255.0”
ifconfig_xl3 = “inet 192.168.0.1 netmask 255.255.255.0”
```

FreeBSD PF Desteği

FreeBSD çekirdeği içerisine, PF desteği ve Bant Genişliği Yönetim desteği aşağıdaki kernel özellikleri eklenerek sağlanır.
(FreeBSD 6.x ve sonrası için çekirdeği tekrar derlemenize gerek yoktur.)

```
# PF Kernel Desteği
device          pf
device          pflog
# Bant Genişliği Yönetim Desteği
options         ALTQ
options         ALTQ_CBQ
options         ALTQ_PRIQ
```

Daha sonra FreeBSD rc.conf yapılandırma dosyası içerisine aşağıdaki tanımlar eklenmelidir. Bu sayede sistem açılışı sırasında, PF otomatik olarak aktif hale geçecektir.

```
pf_enable="YES"     # Sistem Açıldığında PF devreye girer.
pflog_enable="YES" # Sistem Açıldığında PFLOG devreye girer.
```

PF Yapılandırma Dosyası

FreeBSD' nin açılışı sırasında, PF otomatik olarak başlayacaktır. Bu sırada öntanımlı olan /etc/pf.conf dosyası, PF programı tarafından ele alınarak, istenilen kurallar Güvenlik Duvarı' na uyarlanacaktır.

pf.conf yapısı itibariyle 6 ana bölümde incelenebilir.

- **Makrolar:** Tüm pf.conf dosyası içerisinde geçerli olacak tanımlamalar bu bölüm içerisinde yapılır.
- **Tablolar:** Dinamik kurallarda geçerli olacak, ağ veya ip adreslerini içerisinde barındırır.
- **Kirli (Scrubing) Paket Kuralları:** Paket normalizasyonu sırasında kullanılacak olan bölümdür.
- **Bant Genişliği Yönetim Kuralları:** ALTQ ( ) desteği ile ağ üzerinde yer alan paketlerin sıralanması ve bir şekle sokulması bölümünü ele alan kısmıdır.
- **Paket Yönlendirme Kuralları:** Ağ üzerinde bir IP' ye veya Port' a gelen paketlerin, ağ üzerinde farklı bir IP veya Port' a yönlendirilmesi.
- **Paket Filitreleme Kuralları:** Ağ üzerinde gelen giden paketlerin, istenilen kriterlere göre filitreleneceği bölümdür.

Makrolar:

Çok karmaşık ve takibi zor olan ağlarda, pf.conf dosyasının yazılımını kolaylaştırmak için kullanılan bir yöntemdir. Bu bölüm içerisinde verilmiş değerler tüm pf.conf içerisinde geçerlidir.

```
ext_if = “xl0”
web_sunucusu = “10.0.0.5”
block in on $ext_if all
pass  in on $ext_if proto tcp from any to $web_sunucusu
```

Tablolar:

Tablolara kısaca toparlanmış olan IP adresleri olarak bakabiliriz. Birden fazla IP veya Ağ için uygulanacak kurallarda kullanıcılara çok büyük avantajlar getirir. (**const:** İçerisine IP eklenemez, **persist:** İçerisine IP eklenebilir.)

```
table <DMZ> const { 10.0.0.0/24 }
table <YerelAg> persist { 192.168.0.0/24, 192.168.1.0/25 }
block in on $ext_if from any to { <DMZ> <YerelAg } port 80
```

PF Kural Dizilişi

PF Kurallal dizilişi, insanın rahat okuyup anlayabileceği bir sıralama ile yazılmaktadır. Bu diziliş, PF içerisinde bazı durumlarda küçük farklılıklar göstersede genel diziliş her zaman aşağıda verildiği gibi karşımıza çıkmaktadır.

```
hareket [yön] [kayıt|log] [acele|quick] \
   [on hangi kart üzerine] [af adres ailesi] \
   [proto protokol] [from kaynak adresi] \
   [port kaynak port]] [to varış adresi] \
   [port varış portu][flags tcp_bayrakları] [durum]
```

hareket : block | pass

Bu kurala uyan paketin, nihayetinde hangi hareket ile biteceği anlamına gelmektedir. Kural dizilişi başında geçen block ifadesi bu kurala uyan paketlerin artık hareketlerinin bundan sonra izin verilmeyeceği anlamına gelmektedir. Eğer kural dizilişi başında pass ifadesi yer alıyorsa, bu paketin artık yoluna devam edeceği anlamına gelmektedir.

```
block all
pass  all
```

Kural dizilişleri içerisinde, paket filitreleme bölümü içerisinde, en alttaki kural daima en baskın kural olarak karşımıza çıkar. Bir önceki örnekte, tüm paketlerin geçişi yasaklanırken, altında gelen kural tüm paketlerin geçisine izin vermektedir. Bunun sonucu olarak, tüm paketler engellenmeden yollarına devam ederler.

yön : in | out

Bu kural dizilişi içerisinde hareket eden paketin, ağ kartından çıkarken mi bu kurala uyacağı veya ağ kartına girerken mi bu kurala uyacağını belirtir.

kayıt : log

Bu kural dizilişi içerisinde, bu kurala uyan paket veya paketlerin kayıtlarının tutulmasını sağlar. Bu kayıt işlemleri pf ile entegre olarak ve sanal çalışan pflog ağ kartı üzerinden sağlanır. Bu kartın asıl ismi pflog0 olarak belirlenmiştir. Bu kayıt dosyası, standart bir ağ kartı gibi davrandığı için tcpdump komutu kullanılarak, kurallar çerçevesinde kayıtları tutulan tüm paketlerin izlerini takip etmek mümkündür.

```
# tcpdump -nvei pflog0
```

acele : quick

Kural dizilişleri içerisinde bu kurala uyan paketler bir alt kurala geçmeden, harek bölümü her ne ise ona uyarlar. Bu yönerge bulunan kural her zaman önceliklidir. Paket bir alt kurala gönderilmez.

on ag kartı: on $ext_if

Gelen giden paketlerin kontrolü hangi kart üzerinde sağlanacağını belirtir. Güvenlik Duvarı üzerinde yer alan her bir kart için bu yönerge kullanılmak zorundadır.

```
block in on $ext_if all
pass  in on $int_if all
```

af adres ailesi : af inet | af inet6

Güvenlik duvarı üzerinden geçen paketlerin bir ailesi mutlaka olmak zorundadır. Bu aile **inet** (IPv4) veya inet6 (IPv6) olabilir. Bunun dışında herhangi bir adres ailesi bulunmamaktadır. PF bu karara geçen paketlerin ya kaynak adreslerine yada varacakları adres bilgilerine göre karar verir.

```
block in quick on $ext_if af inet6 all
```

proto protokol : proto { tcp udp icmp }

Güvenlik duvarı üzerinden geçen paketlerin, hangi protokol içerisinde yer aldıklarını belirleyen yönergedir.

proto

- TCP
- UDP
- ICMP
- ICMP6
- /etc/protocols dosyası içerisinde yer alan geçerli bir protokol ismi.
- 0-255 arasında yer alan protokol numarası
- Bir liste içerisinde yer alan protokol kümesi.

```
block in       on $ext_if af inet6 all
block in quick on $ext_if proto icmp all
pass  in quick on $ext_if af inet proto { 6 17 }
pass  in quick on $ext_if proto gre
```

`from kaynak_adresi to varış_adresi`

Güvenlik duvarı üzerinden geçen paketlerin IP başlıkları içerisinde belirlenen adreslerinin ayrıştırılmasında kullanılan yönergesidir.

kaynak_adresi/varış_adresi:

- Tek IPv4 veya IPv6 adresi
- CIDR ağ bloğu.
- Ağ Maskeli ağ bloğu.
- FQDN (DNS isim çözümlemesinden dönen tüm IP adresleri eklenir.)
- Bir ağ kartının ismi. (Bu ağ kartı üzerinde tanımlanana tüm IP adresleri için kullanılır.)
- Parantez içerisinde yer alan ağ kartı ismi. (Dinamik olarak değişen ağ kartına bağlı IP adresleri, PF tarafından tanınarak kural içerisine otomatik olarak yansıtılır. DHCP yoluyla IP alan ağ kartları için kullanışlı bir parametredir.)
- Tüm adresler : **any**
- **from any to any** kısaltması **all**
- Bir liste içerisinde kullanılan tüm IP adresleri

kaynak_portu / varış_portu

Güvenlik duvarı üzerinden geçen paketlerin istenilen IP adresi içerisinde hangi porta gittiğini veya hangi porttan geldiğini tanımlamakta kullanılan yönergeyi ifade etmektedir.

port :

- 1 ila 65535 arasında yer alan bir port numarası olabilir.
- /etc/services dosyası içerisinde yer alan geçerli servis isimlerinden bir tanesini içerebilir.
- Bir liste içerisinde tanımlanan port aralıkları veya tek tek tanımlanan portların ifadesinde kullanılır.
- Aralık:
  - != Eşit Değil
  - < Küçüktür
  - > Büyüktür
  - <= Küçük ve eşittir
  - >= Büyük ve eşittir
  - >< Aralık (port 1234 >< 2345 [1234 ve 2345 dahil değildir.])
  - <> Ters Aralık (port 1234 <> 2345 [1 den 1234 de ve 2345 den 65535 e kadar olan aralığı ifade eder.)

kaynak_portu / varış_portu (devam)

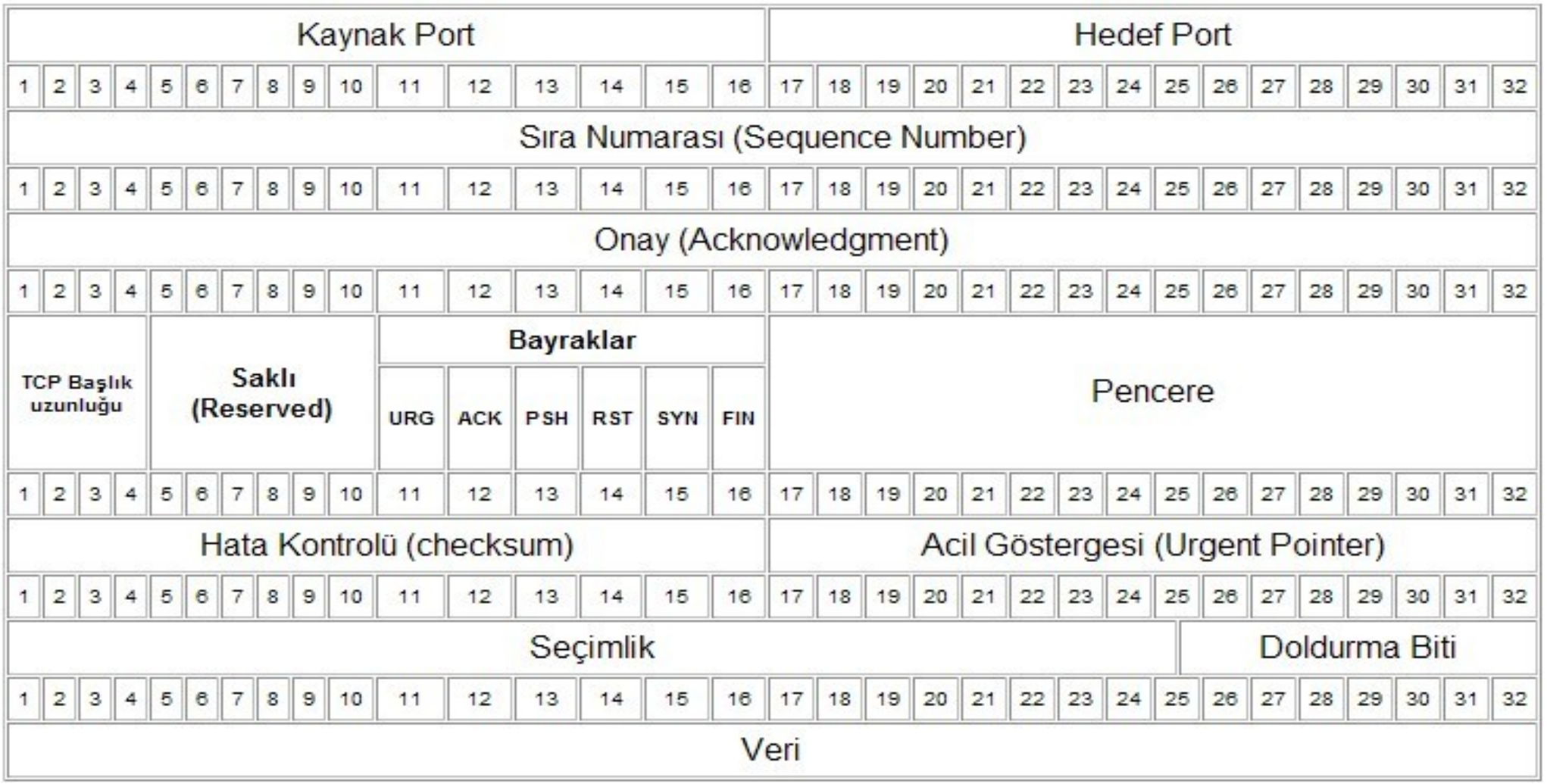
Güvenlik duvarı üzerinden geçen paketlerin istenilen IP adresi içerisinde hangi porta gittiğini veya hangi porttan geldiğini tanımlamakta kullanılan yönergeyi ifade etmektedir.

port :

- : İçerisine alan aralık (port 1234:2345 [1234 ve 2345 portları da dahil olmak üzere bu port aralığında yer alan tüm portlar])

Tcp Bayrakları

TCP katmanı içerinde yer alan bayrak durumlarına düzenleme yapacak yönergedir. Burada **proto tcp** yönergesinin aktif olması ve ondan sonra ise **flags** yönergesi ile istenilen tcp bayraklarının belirlenmesi gerekmektedir.



TCP Bayrakları (devam)

Bayrak bitleri çeşitli denetim fonksiyonlarını sağlar. Bu bitler bağlantı kurma, bağlantı koparma ve denetim bilgisi için kullanılır.

- **URG** Bayrağı : Urgent (Acil) bayrağının 1 olması Acil Göstergesi (Urgent Pointer) bölümünün kullanımda olduğunu gösterir.
- **ACK** Bayrağı: Acknowledgment (Onay) bayrağının 1 olması Onay alanının geçerli olduğunu gösterir.
- **PSH** Bayrağı: Push bayrağı, gönderen TCP 'ye gönderilecek veriyi hemen göndermesi için emir verir ve bu emir alt katmanlara iletilir. Benzer biçimde, alt katmandan gelen verinin hemen üst katmana aktarılması için TCP 'yi zorlar. Normalde TCP bu işlemi kendi önceliğine göre yapar ancak PuSH bayrağı bu önceliği değiştirebilir.
- **RST** Bayrağı: Reset bayrağı sorunlu veya kopmak üzere olan bağlantıları başlangıç durumuna getirmekte kullanılır.
- **SYN**chronize Bayrağı: Synchronize Bayrağı, gönderilen ilk paket ise gönderen ve alan tarafından kurulur. Bunun kurulması gönderen ve alanın sanal bağlantı isteğinde bulundukları anlamını taşır. Bağlantıyı gerçekleştirmek için üç aşamalı işlem gerçekleştirilir.
- **FIN**ish Bayrağı: Bu bayrak gönderenin daha fazla verisi olmadığını belirtir ve bağlantı koparılabilir.

TCP Bayrakları (devam)

PF içerisinde flags yönergesi kontrol/maske şeklinde yorum içerisine alınır. Maske içerisindeki bayraklar kontrol edildikten sonra sade kontrol kısmındaki bayrak aktif ise PF bu duruma uyar.

```
block in log quick on $DIS proto tcp from any to any flags /S
block in log quick on $DIS proto tcp from any to any flags /SFRA
block in log quick on $DIS proto tcp from any to any flags /SFRAU
block in log quick on $DIS proto tcp from any to any flags A/A
block in log quick on $DIS proto tcp from any to any flags F/SFRA
block in log quick on $DIS proto tcp from any to any flags U/SFRAU
block in log quick on $DIS proto tcp from any to any flags SF/SF
block in log quick on $DIS proto tcp from any to any flags SF/SFRA
block in log quick on $DIS proto tcp from any to any flags SR/SR
block in log quick on $DIS proto tcp from any to any flags FUP/FUP
block in log quick on $DIS proto tcp from any to any flags FUP/SFRAUPEW
block in log quick on $DIS proto tcp from any to any flags SFRAU/SFRAU
block in log quick on $DIS proto tcp from any to any flags SFRAUP/SFRAUP
```

Yukarıda yer alan kurallar dizisi, port tarama yapma usuluyle Güvenlik Duvarını tarayan, kötü niyetli kişilerin engellenmesinde kullanılır. Ayrıca log yönergesi ile bu kişilerin hangi IP ler üzerinden geldiklerini de tespit etmek mümkün olacaktır.

Durum : state

PF kural dizilişi içerisinde, kurala uyan paketlerin durum bilgilerinin tutulmasında kullanılan yönergedir. Durum Denetimi (Stateful Inspection) 'ni göz önünde bulundurarak, paketlerin kontrolünü yapan bir güvenlik duvarı herzaman gerekmektedir. İstemcilerin, Güvenlik Duvarı üzerinden yaptıkları erişim bilgileri bir tablo içerisinde tutulur. Bu eşime geri dönen cevaplar, Güvenlik Duvarı içerisinde yer alan **Durum(state)** tablosundan kontrol edilir. Eğer istek durum tablosu içerisinde varsa, yani içeriden gelen bir isteğin devamı ise,paketin içeriye girmesine izin verilir. Aksi taktirde, paket düşürülür.

- **keep state:** Sadece durum kontrolü yapılır. Bu durum kontrolü TCP, UDP ve ICMP protokolleri için geçerlidir.
- **modulate state:** TCP protokolü içerisinde akmaya başlayan bir veride, ISN(initial sequence number) bölümünün güçlendirilmesi için kullanılır. Genellikle Microsoft işletim sistemleri için geçerli olan TCP/IP katmanındaki boşlukların suistimaline karşı alınmış bir önlemdir. Bu durum sadece TCP protokolü için geçerlidir.

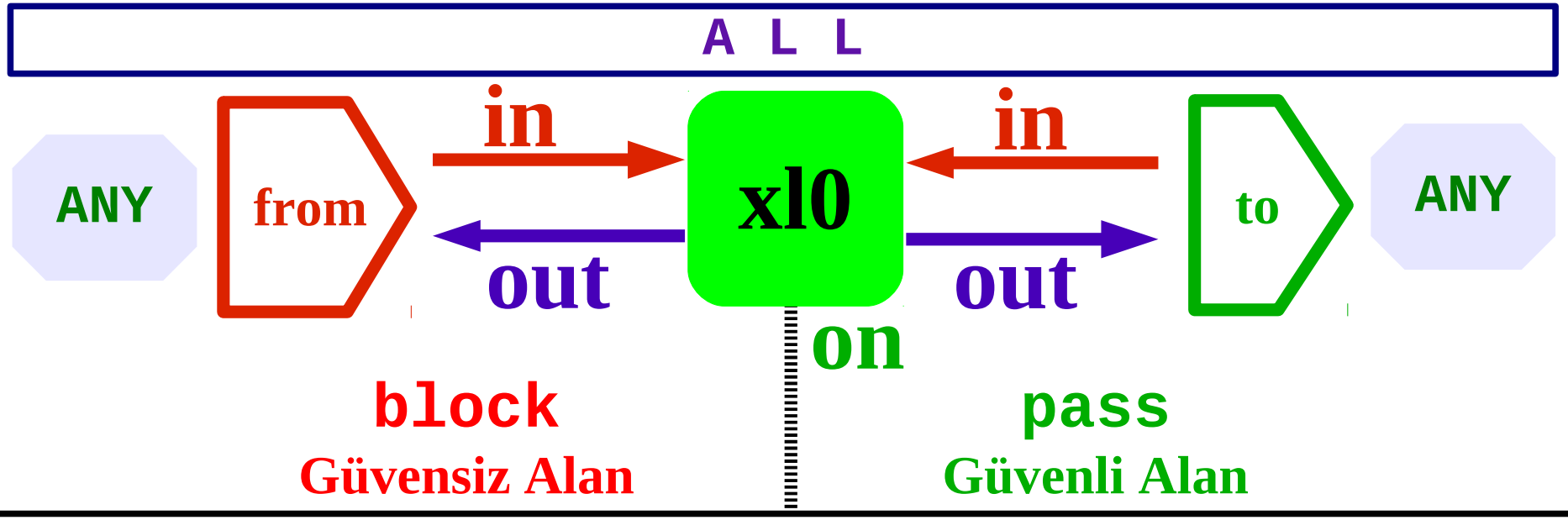
Durum : state (devam)

- **synproxy state:** Özellikle sunucu sistemlerine karşı yapılan SYN saldırılarında kullanılan bir durumdur. (Spoofed TCP SYN flooding attacks**) yalancı ağlardan gelerek, TCP SYN baskın saldırısını kullanan, kötü niyetli kişilerin saldırılarından korunmak için oluşturulmuş bir tampon alan, Güvenlik Duvarı tarafından oluşturulur. TCP balantıları bu tampon alandan geçerken, bütün SYN paketleri direk olarak alıcısına teslim edilmeden, 3 aşamalı el sıkışma prosedürüne tam olarak uyup uymadığı kontrol edilir. Bu prosedüre uymadan gelen, SYN paket baskınları (ard-arda gönderilen tüm SYN paketleri), Güvenlik Duvarı tarafından alıcısına teslim edilmez ve düşürülür. Bu sayede SYN baskın saldırılarının önüne geçilmiş olur. Synproxy state yönergesi aynı zamanda hem keep state yönergesini hemde moduleate state yönergelerinide kendi içerisinde barındırır.

** http://www.networkcomputing.com/unixworld/security/004/004.txt.html

PF Kelimleri

PF kural dizileri içerisinde çok sık kullanılan tanımlar da bulunmaktadır.



- **block**: İfade edilen paketlerin engellenmesini sağlar.
- **pass** : İfade edilen paketlerin yoluna devam etmesini sağlar.
- **on**: Üzerinde işlem görülecek olan, ağ kartının tanımlanmasında kullanılır.
- **in**: Üzerinde çalışılan karta doğru gelen, tüm paketlerin ifadesinde kullanılır.
- **out**: Üzerinde çalışılan kartan giden, tüm paketlerin ifadesinde kullanılır.
- **all**: Yerine göre, tüm Ağ Kartlarını, IP' leri, Port' ları ifade eder.
- **any**: Kullanıldığı yerde, tüm IP' leri veya Port' ları ifade eder.
- **from**: Paketlerin nereden geldiğini işaret eder.
- **to**: Paketlerin nereye gittiğini işaret eder.

“PF Kural Dizilişleri” içerisinde öğrenmiş olduğumuz kelimeleri, artık Güvenlik Duvarı için cümleler oluşturmaya başlayabiliriz.

Paket Geçişini Durdurmak:

Güvenlik Duvarlarında genel yaklaşım olarak, ön tanımlı olarak tüm paketlerin kısıtlanması, ardından ağ üzerinde kullanılacak olan servislerin açılması esastır. Bu sayede, standart ağ servislerine izin verilir, diğer tüm servisler engellenir.

Güvenlik Duvarı üzerinden geçen paketlerin geçişini engellemek için block hareketini kullanırız.

```
block in  all  # Güvenlik Duvarına gelen tüm paketler
block out all  # Güvenlik Duvarından çıkan tüm paketler
```

Bu sayede geçen (sisteme giren ve çıkan) tüm paketleri engellemiş oluruz. Kural dizilişleri içerisinde unutulmaması gereken nokta, en son yazılan kural – eğer içerisinde quick kelimesi yoksa – en baskın kuraldır.

Güvenli Ağ Üzerinden Çeçiş Yapmak:

Tüm geçişleri kapatmanın ardınan, güvendiğimiz ağdan dışarı çıkışlara izin vermemiz gerekir.

```
ext_if="xl0"
int_if="xl1"
int_net="192.168.1.0/24"

pass out quick on $ext_if from $int_net \
                                    to any keep state
block in        on $ext_if all
block out       on $ext_if all
```

Durumun korunması için gerekli yönerge -keep state- yazılmamış olsaydı, $int_net 'in dışarı çıkışı için izin verilen paketlerin cevapları içeri alınmayacağından, bağlantı sorunlu olacaktır.

Güvenli Servisler Çeçiş İzni Verme I:

Ağmızda hizmet veren sunuculara, bu servislerini verebilmeleri için geçiş haklarının tanımlanması gerekmektedir.

```
ext_if="xl0"
dmz_if="xl2"
dmz_net="10.0.0.0/24"
eposta_snc="10.0.0.2"
web_snc="10.0.0.3"
webports="{ 80 443 8080 }"
epostaports="{ 25 110 143 }"

pass in  quick on $ext_if from any to $web_snc \
                 port $webports keep state
pass in  quick on $ext_if from any to $eposta_snc \
                 port $epostaports keep state
block in         on $ext_if all
block out        on $ext_if all
```

Güvenli Servisler Çeçiş İzni Verme II (TCP Bayrakları):

Ağmızda hizmet veren sunuculara legal TCP bağlantısı yaparak gelen istemciler için izin verilmesi gerekir. Tek başına servisleri korumak bazen işe yaramayabilir.

```
win_snc="10.0.0.4" # Windows XX / IIS Sunucusu
web_snc="10.0.0.5" # UNIX / Apache Sunucusu
win_ports="{ 21 80 443 }"
web_ports="{ 80 443 }"
pass in quick on $ext_if proto tcp from any to \
   $win_snc port $win_ports flags S/SA  modulate state
pass in quick on $ext_if proto tcp from any to \
   $web_snc port $web_ports flags S/SA keep state

block in  all
block out all
```

Güvenli Servisler Çeçiş İzni Verme II (TCP Bayrakları): (devam)

TCP bayrakları, ağ üzerinde servis veren sunuculara yeni bağlantılar oluşturulmaya başlandığında, paketlerin kontrolünde kullanılır.
PF flags check/mask şeklinde paketlerin içerisine bakar. “mask”, PF nin hangi bayrakları inceleyeceğine bakar. “check” ise bu bayraklardan hangilerinin aktif olduğuna bakar.

flags:

- **F: FIN : Bağlantı sonu**
- **S: SYN : Bir bağlantıya başlama isteği**
- **R: RST : Bir bağlantıyı sonlandırır**
- **P: PUSH : Paket anında gönderilir**
- **A: ACK : Paket onaylanır**
- **U: URG : Acil gönderme**

flags S/SA: PF S ve A paketlerine bakar ve sadece S paketi aktif ise kurala uyar.

Ağ Adres Çevirimi:

İnternet ortamında iletişim, gerçek IP adresleri üzerinden yürütülür. Gerçek Ipv4 adreslerindeki kısıntıdan dolayı, bir veya daha fazla ağ, tek bir gerçek IP adresine eşlenmesi gerekebilir. Bu durumda NAT kullanılır. RFC-1918 standardına göre, reserve edilmiş ağlardan, gerçek IP dünyasına geçmek için NAT işlemi mutlaka olmalıdır.

```
[no]nat [pass] [on|üzerine] ağ_kartı \
   [from kaynak_adresi] [port kaynak_portu] \
   [to varış_adresi] [port varış_portu] \
         -> [dış_nat_ipsi] [statik_port]
```

- **nat** : Bu yönergeden sonra bir nat kuralı yazılacağını gösterir.
- **pass** : Paket Filitreleme kurallarına uymadan bu paketin direk geçmesine izin verilir.
- **on ag_kartı** : Üzerinde NAT işleminin gerçekleştirildiği, ağ kartını ifade eder.

Ağ Adres Çevirimi: (devam)

- **->** : Nat işlemine geçiş yapılıyor.
- **dış_nat_ipsi:** Natlama işlemi sonucunda dönüşüm sağlanan IP adresi.
- **statik_port** : Tek bir IP ye birden fazla ip adresi için NAT yapılacağında, alt ağ grupları için portlar statik olarak yapılandırılabilir.

```
ext_if="xl0"
ic_net="192.168.0.0/24"
dis_ip="xxx.xxx.xxx.xxx"
nat on $ext_if from $ic_net to any -> $dis_ip
```

Eğer bir ağ içerisinde yer alan tek bir IP adresini veya IP adres listesinin natlanması istenmiyorsa, nat kuralının başına no yönergesi eklenmelidir.

```
no nat on $ext_if from 192.168.1.10 to any
nat on $ext_if from 192.168.1.0/24 to any -> $dis_ip
```

Ağ Adres Çevirimi: (devam)

Eğer sadece sanal bir IP adresinin sadece gerçek bir IP adresine natlanma işlemi için **binat** (1:1) yönergesi kullanılır. Bu sayede DMZ tarafında bulunan sunucuların herbirisi için, farklı IP adresleri kullanılabilir.

```
web_server_ic="10.0.0.3"
web_server_dis="199.200.201.203"
mail_server_ic="10.0.0.4"
mail_server_dis="199.200.201.204"
dis_ip="199.200.201.205"
binat on $ext_if from $web_server_ic \
                    to any -> $web_server_dis
binat on $ext_if from $mail_server_ic \
                    to any -> $mail_server_dis
no nat on $ext_if from 192.168.1.10 to any
nat on $ext_if from 192.168.1.0/24 to any -> $dis_ip
```

Ağ Adres Çevirimi: (devam)

Bir ağ kartına gelen paketin, diğer ağ kartlarına yansıtılmaları için, işletim sistemi tarafında yapılması gereken ufak bir ayarlama yer almaktadır. IP yönlendirme parametrelerinin işletim sistemi tarafında aktif olmaları gerekmektedir.

Aşağıda yer alan iki komut bu işlemi gerçekleştirmektedir.

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

Bu komutların kalıcı olabilmesi ve işletim sisteminin tekrar açılması sırasında, otomatik olarak yapılabilmesi için /etc/sysctl.conf dosyası içerisine bu parametrelerin kaydedilmeleri gerekmektedir.

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

IP veya Port Yönlendirmesi:

Güvenlik Duvarına doğru gelen paketlerin, yönlerinin değiştirilerek, farklı port veya IP adreslerine yönlendirilmelerinin gerektiği durumlar olabilir. Bu durumlarda, Güvenlik Duvarı içerisinde rdr (Redirection) yönergesi kullanılmalıdır. Bu yönlendirmenin genel amaçlarından bir tanesi, yerel ağ içerisinde bulunan sunuculara, dış dünyadan gelen paketlerin yönlendirilmesidir. Yönlendirme işlemi, paketin ilk eriştiği ağ kartı üzerinden yapılmalıdır.

```
ext_if="xl0"
web_ic="10.0.0.3"
ssh_ic="10.0.0.4"
ext_ad="99.100.101.102"
rdr on $ext_if proto tcp from  any to \
                        $ext_ad port 80 -> $web_ic
rdr on $ext_if proto { tcp udp } from any to \
                        $ext_ad port 22 -> $ssh_ic
```

IP veya Port Yönlendirmesi: (devam)

Güvenlik Duvarına gelen isteklerin, yerel alanda bulunan sunucuların farklı portlarına aktarmak ta mümkündür. Bu işleme durumuna göre yansıtma veya yeniden yön verme (reflection veya redirection) denir.

```
ext_if="xl0"
int_if="xl1"
webserver="10.0.0.4"
icnet="192.168.0.0/24"
prxserver="10.0.0.5"
rdr on $ext_if from $icnet to 200.201.202.203 port 80 \
                              -> $webserver port 8000
rdr on $ext_if from any to any port 80 \
                              -> $webserver port 8080
rdr on $int_if from $icnet to ! 200.201.202.203 \
                        port 80 -> $prxserver port 3128
```

NAT Adres Havuzları ve Yük Dengeleme

NAT Adres Havuzu

Birden fazla IP adresi üzerinden NAT işlemi gerçekleştirilecekse, bu durumda IP adresleri bir liste içerisine alınarak NAT işlemi tamamlanır.
Bu işlem round-robin şeklinde gerçekleştirilir.

```
yerelnet="192.168.0.0/16"
nat on $ext_if from $yerelnet to any \
        -> { 200.201.202.203 200.201.202.204 }
```

Ya da bir alt ağ grubu için de bu durum söz konusudur.

```
yerelnet="192.168.0.0/16"
nat on $ext_if from $yerelnet to any ->
                                200.201.202.128/27
```

Yük Dengeleme

Yerel ağda bulunan sunuculara gelen isteklerin bölünmesinde kullanılan etkin bir yöntemdir. Bu sayede, tek bir servis için birden fazla kullanılan sunucuların yükleri dengelenmiş olur.

```
web_sunuclari="{ 192.168.0.1 192.168.0.2 192.168.0.3 }"
rdr on $ext_if proto tcp from any to any \
      port 80 -> $web_sunucular i \
            round-robin sticky-address
```

Sunucular üzerindeki durumların tam olarak korunabilmesi için burada stick-address yönergesinin kullanılması gerekmektedir. Bağlantılarda oluşan süre aşım hatalarında bu yönerge ile tutarlılık sağlanmış olur.

Kirli (Scrubing) Paketlerin Düzenlenmesi:

İnternet üzerinde, gelen giden paketler, herzaman istenildiği gibi ideal olmaya bilir. Bunun birden fazla sebebi olabilir. Bunlardan birisi yanlış yapılandırılmış, yönlendirici (router) ayarlarından kaynaklanıyor olabilir. Dahası, kötü niyetli kişilerin, TCP/IP yapısını suistimal etme işlemlerinde, genellikle kirli (defragmented) paketleri kullanılar. Özellikle, Microsoft Windows İşletim sistemlerindeki TCP/IP yapısını suistimal etmek için, kullanılan yöntemler arasında bu kirli paketler kullanılır.
Bu suistimallerin aşılmasında ve daha sağlıklı paket yapıları için PF nin getirdiği yeniliklerden birtanesi Paket Normalizasyon işlemidir.

```
scrub (in|out) on $ext_if all (inet|inet6)
scrub in  on $dmz_if from $dis_net to $dmz_net
scrub out out $ext_if from $local_net to any
```

PF Bant Genişliği Yönetimi

Bant Genişliği Yönetimi:

Kurumların, kendi ihtiyaçları doğrultusunda temin ettikleri İnternet altyapılarının, daha etkin ve verimli kullanılabilmesi için Bant Genişliği yönetimi kaçınılmaz bir sonuç oluşturmaktadır. Özellikle, son zamanlarda artan İnternet altyapı hızları, sunuculardan veya p2p ortamlardan indirilen programların artmasını sağlamıştır.

Bant Genişliği Yönetimi için, İnternet altyapı hızınıza uygun bir kuyruk tanımlaması ile işe başlanır. Bu kuyruk, istenilen şekilde alt erişim kuyruklarına bölünerek, kural tanımlamalarında kullanılmak için hazır hale getirilir. (CBQ)

Buna ek olarak eğer istenirse, Ağ üzerinde gelen giden tüm paketlerin, paket özelliklerine göre önceliklendirme işlemleri gerçekleştirilebilir. (PRIQ)

Bant Genişliği Yönetimi kural dizilişi **altq** anahtar kelimesi ile başlar.

```
altq on $ext_int (priq|cbq) bandwidth 1(Gb|Mb|Kb) \
                        queue { ssh, mail, boss }
```

PF Bant Genişliği Yönetimi

Paket Önceliklendirme (PRIQ):

Ağ üzerinde gelen giden tüm paketlerin önceliklerini, PF' nin bu özelliğini kullanarak rahatlıkla yapılandırabiliriz. Diyelim ki, yurt dışı bağlantıları olan bir firma, eposta ve web ağırlılı çalışmaktadır. Dolayısıyla, ağ üzerinde gelen ve giden bu paketlerin öncelik sırası diğer paketlere göre önde olacaktır.

```
altq on $ext_if priq bandwidth 1Mb queue { web, mail, dns,
other }
      queue dns   priority 14 priq(red)
      queue mail  priority 12 priq(red)
      queue web   priority 11 priq(red)
      queue other priority 9  priq(default)
```

Yukarıdaki kural tanımında, PRIQ tabanlı bir bant genişliği yönetim kuyruğu oluşturulmaktadır. En öncelikli palet en yüksek önceliklendirme (priority) numarasına sahiptir. Burada en öncelikli servis, DNS hizmetleridir.

Not: priq(red) : Randon Early Detection : Kuyruğa ilk giren rasgele paket mantığı üzerine oluşturulmuş bir yaklaşımdır. **PRIQ** hizmetinde genelde **red** kulanılır. Kuyruk boyutunun dışında kalan ilk paket düşürülür.

Paket Önceliklendirme (PRIQ): (devam)

```
pass out quick on $ext_if inet proto udp \
        from any to any port 53 keep state queue dns
pass out quick on $ext_if inet proto tcp \
        from any to any port 53 keep state queue dns
```

UDP ve TCP protokolleri üzerinde, 53 port istekleri DNS kuyruğuna sokularak paket önceliklendirmesi etkin kılınıyor. Burada paketler, $ext_if (Güvenlik Duvarı Dış Erişim Noktası) üzerinden dışarıya çıkmalarına izin verilir. Bu çıkış sırasında, kuyruk içerisindeki paketlerden, DNS hizmeti için görevlendirilmiş paketler, önceliklendirilir.

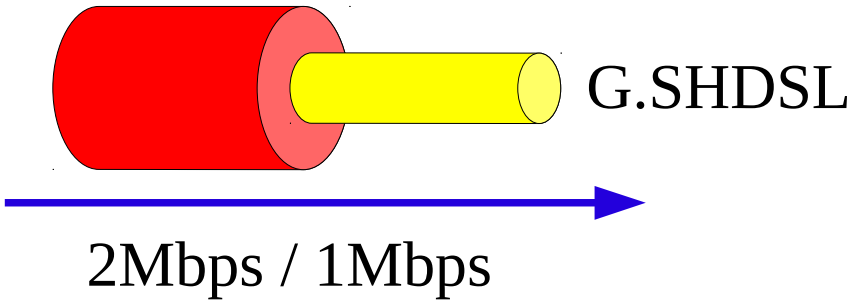
```
pass out quick on $ext_if inet proto tcp \
   from any to any port { 25 110 } keep state queue mail
pass in quick on $ext_if inet proto tcp \
        from any to $mail_server \
            port { 25 110 } keep state queue mail
```

Paket Önceliklendirme (CBQ):

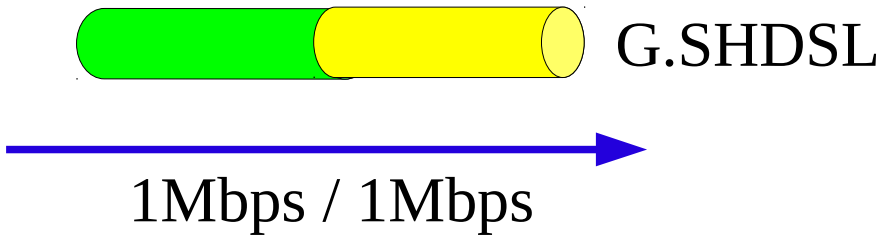
Bazı durumlarda, ağ üzerindeki ihtiyaçlarınızı, paket önceliklendirme yöntemi ile aşamayabilirsiniz. İhtiyacınız olan durum, her bir kuyruğun kendisine has bir bant genişliğine ihtiyaç duymasıdır. Bu durumda, ister IP temelli, ister Protokol temelli ya da isterseniz özel servis temmeli olarak kuyruk oluşturma ile, bu durumun üstesinden gelebilirsiniz.

Öncelikli olarak, oluşturacağımız kuyruğun, sizin internet erişim bant genişliğiniz ile aynı olmasını sağlamaktır. Örnek vermek gerekirse, internet erişimi G.SHDSL üzerinden 1Mbps hız ile sağlayan bir kurum için bu kuyruk boyutuda 1Mbps olmalıdır.

Yanlış Kuyruk Tanımlaması



Doğru Kuyruk Tanımlaması



Paket Önceliklendirme (CBQ): (devam)

PRIQ kuyruk oluşturulması gibi, CBQ kuyruk oluşturulması da aynı dizilim ile yapılır.

```
altq on $ext_if cbq bandwidth 1Mb \
              queue{dmznet, ozelnet, localnet}

queue dmznet bandwidth 40% priority 6 cbq(red)
queue ozelnet bandwidth 35% priority 6 cbq(red)
queue localnet bandwidth 15% priority 5 cbq(default)
```

CBQ kuyruğu burada 1Mbps' lik bir bant geniştiğine ayarlanıyor. Ana kuyruk içerisinden, çeşitli orantısal boyutlarla üç alt kuyruk oluşturuluyor. Burada **dmznet** kuyruğu, DMZ tarafındaki sunucular için oluşturulmuş ve genel bant genişliğinin 40% kısmını teşkil etmektedir; **ozelnet** kuyruğu için bu oran 35% ve **localnet** kuyruğu için ise 15% olarak tasarlanmıştır. Burada dikkat edilecek bir diğer nokta ise, kuyruklar içerisindeki paketlerin önceliklendirme sayılarıdır. **dmznet** ile **ozelnet** için bu sayı 6 iken **localnet** için 5 ile sınırlandırılmıştır.

Paket Önceliklendirme (CBQ): (devam)

```
pass out quick on $ext_if inet proto {tcp, udp} \
   from ($ext_if) port 1024 >< 32255 to any queue dmznet
pass out quick on $ext_if inet proto {tcp, udp} \
   from ($ext_if) port 32256 >< 65535 to any queue ozelnet
```

PF Kural Yönetimi (pfctl)

Güvenlik Duvarı Yönetimi (pfctl):

Yapılandırdığımız Güvenlik Duvarı üzerinde, kuralların tekrar Güvenlik Duvarına okutulması, kuralların durumlarının kontrolü, Güvenlik Duvarının kapatılıp açılması gibi işlemlemlerin yapılması gerekebilir. Bu şartlar altında Güvenlik Duvarı yönetim programı olan **`pfctl`** kullanılır.

Güvenlik Duvarının Aktif veya Pasif Yapılması:

`# pfctl -e` (PF aktif hale getirilir.)
`# pfctl -d` (PF pasif hale getirilir.)

Güvenlik Duvarından Gerekli Bilgilerin Alınması:

`# pfctl -sr` (Geçerli olan kuralların bilgilerini gösterir.)
`# pfctl -sn` (Geçerli NAT kurallarının bilgilerini gösterir.)
`# pfctl -ss` (Geçerli olan durum(state) tablosunu gösterir.)
`# pfctl -si` (PF üzerindeki tüm istatistiki bilgileri gösterir.)
`# pfctl -sa` (PF ile ilgili herşeyi gösterir.)

Güvenlik Duvarı Yönetimi (pfctl): (devam)

Güvenlik Duvarı içerisine tüm kuralları tekrar yükler.

```
# pfctl -f /etc/pf.conf
```

Güvenlik Duvarı içerisine sadece NAT kurallarını yükler.

```
# pfctl -Nf /etc/pf.conf
```

Güvenlik Duvarı içerisine sace Filitre kurallarını yükler.

```
# pfctl -Rf /etc/pf.conf
```

Güvenlik Duvarı kurallarının yazımını kontrol eder, Bir şey yüklemez.

```
# pfctl -nf /etc/pf.conf
```

Güvenlik Duvarı Yönetimi (pfctl): (devam)

Standart Girdi(STDIN) olarak Güvenlik Duvarına kural ekler.

```
# echo "block in all" | pfctl -f -
```

Güvenlik Duvarı içerisine herşeyi sıfırlar(flush) ve tüm kuralları tekrar yükler.

```
# pfctl -Fa -f /etc/pf.conf
```

Güvenlik Duvarı Makro değerlerini değiştirir.

```
# pfctl -D makro=deger
# pfctl -D ext_if=sk0
```

Güvenlik Duvarı kuyruk bilgilerini gösterir.

```
# pfctl -s queue
# pfctl -v queue (çok daha fazla bilgi verir. -vv )
```

Güvenlik Duvarı Yönetimi (pfctl): (devam)

Güvenlik Duvarı bilgileri için daha fazla araç kaynağı.

- *pfstat . pf(4) istatistillerini toplar ve çizer.*
  http://benzedrine.cx/pfstat.html (pfstat)
- *pftop . top(1) veya ntop programına benzer, pf(4)' e ait temel istatistik bilgilerini gösterir.*
  http://www.eee.metu.edu.fr/~canacar/pftop/ (pftop)
- *fwanalog. pf(4) kayıt dosyalarını çözümler ve Analog formata çevirir.*
  http://www.tud.at/programm/fwanalog (fwanalog)

Güvenlik Duvarı Kayıt Dosyası Analizi:

İstediğimiz kuralların kayıtlarının incelenmesi ve sonuçlarının irdelenmesi için tcpdump komutu kullanılarak, pflog0 aygıtı üzerinden geçen kayıtlar incelenebilir.

Güvenlik Duvarı kayıtlarını gerçek zamanlı olarak gösterir.

```
# tcpdump -n -e -ttt -i pflog0
```

Güvenlik Duvarı kayıtlarının dosyadan incelenmesi.

```
# tcpdump -r /var/log/pflog
```

Güvenlik Duvarı kayıtlarının belirli port, IP veya ağ grubuna ait olanları inceler.

```
# tcpdump -r /var/log/pflog port 80
# tcpdump -r /var/log/pflog host 200.201.202.203
# tcpdump -r /var/log/pflog net 200.201.202.0/24
```

Teşekkür Ederim..

Kaynaklar:

- http://www.openbsd.org/faq/pf/index.html
- http://pf4freebsd.love2party.net/
- http://www.benzedrine.cx/pf.html
- http://www.rofug.ro/projects/freebsd-altq/
- http://www.onlamp.com/pub/ct/58
- http://kerneltrap.org/node/view/627

Lisans

Bu belge, GNU Özgür Belgeleme Lisansı altında dağıtılmaktadır.

Bu belgeyi, Free Software Foundation tarafından yayınlanmış bulunan GNU Özgür Belgeleme Lisansının 1.2 ya da daha sonraki sürümünün koşullarına bağlı kalarak kopyalayabilir, dağıtabilir ve/veya değiştirebilirsiniz.
Lisans'ın bir kopyasını http://www.gnu.org/copyleft/gfdl.html adresinde bulabilirsiniz.
Bu belgedeki bilgilerin kullanımından doğacak sorumluluklar ve olası zararlardan belge yazarı sorumlu tutulamaz. Bu belgedeki bilgileri uygulama sorumluluğu uygulayana aittir.
Tüm telif hakları aksi özellikle belirtilmediği sürece sahibine aittir.
Belge içinde geçen herhangi bir terim, bir ticari isim ya da kuruma itibar kazandırma olarak algılanmamalıdır. Bir ürün ya da markanın kullanılmış olması ona onay verildiği anlamında görülmemelidir.