OpenBSD PacketFilter(PF) ile Firewall Uygulamaları

Yıllarca NetBSD takımında aktif geliştirici olarak bulunmuş Theo de Raadt NetBSD işletim sistemini geliştirilmesi ile ilgili olarak diğer developerlarla birçok noktada farklı düşünüyordu, sonrasında NetBSD takımından ilişkisi kesilmesiyle birlikte kendi düşlediği işletim sistemini oluşturmak amacı ile OpenBSD Projesini kurdu. Kısa sürede aynı fikirleri paylaştığı birçok geliştirici OpenBSD projesine geçiş yaptı. Amaçları güvenli, sağlam, kullanışlı ve tamamen "özgür" bir işletim sitemi ortaya çıkarmak olan bu takımın, başladığı günden bugüne kadarki gelişim evresini izlersek amaçlarının çoğuna ulaşmış olduklarını görebiliriz.

PF(PacketFilter) OpenBSD takımında doğan ve sonrasında diğer *BSD(NetBSD, FreeBSD)lere port edilen ticari yazılımlarla birebir rekabet edebilecek bir Firewall ürünüdür. Yapılandırma biçimi daha once OpenBSD tarafından da kullanılan IPF e çok benzer fakat kodları tamamen sıfırdan yazılmıştır. OpenBSD 3.0 ile birlikte artık çekirdekte IPF'e ait herhangi bir kod yoktur bunun yerine PF gelmiştir.
PF'i yapılandırabilmeniz için OpenBSD ağ yönetimi hakkında bilmeniz gereken bazı noktaları hatırlatıp PF'in kullanımına geçeceğim. Tüm dünyada olduğu gibi Turkiye'de de OpenBSD ile ilgili yerel dilde yazılmış döküman eksikliği vardır. OpenBSD ile ilgili kısa ipuçları ile sisteminizi tanımak isterseniz http://ipucu.enderunix.org/index.php?tip_id=2&lang=tr adresine başvurabilirsiniz.

OpenBSD de Basit Ağ ayarları

İlk olarak sistemimizdeki arabirimlerin hangi adlarla tanındığını bulmalıyız, bu Linux kullanıcıları için biraz garip kaçabilir çünkü Linux'da Ethernet kartının markası ne olursa olsun birinci Ethernet kartı eth0, ikinci Ethernet kartı eth1 şeklinde adlandırılır.

OpenBSD ve diğer *BSD lerde ise durum biraz farklıdır ve arabirim kartının markasına gore isim alır, şöyleki "a" marka bir Ethernet kartınız "xl0" olarak adlandırılırken "b" marka bir Ethernet kartı tamamen diğer Ethernet kartından bağımsız olarak "fxp0" olabilir.

Bunun getireceği çeşitli avantaj ve dezavantajlar vardır , *BSD listelerindeki arama fonksiyonunu kullanarak geçmişte bu konular ile ilgili yapılmış tartışmaları okuyabilirsiniz.

Ağ Arabirimlerini Yapılandırmak

Sistemimiz tarafından tanınmış ağ aygıtlarına ait yapılandı rma bilgilerini görebilmek ve değiştirebilmek için diğer Unix/Linux larda olduğu gibi "ifconfig" komutunu kullanarız. OpenBSD de ifconfig komutunu –a parametresi ile kullandığımızda sistemdeki tüm arabirimlerin özelliklerini görebiliriz. Sadece bir arabirime ait özelliklere bakmak istiyorsak ifconfig komutuna parametre olarak sadece o ağ arabirimin adını vermek yeterlidir, mesela xl0 arabirimine ait yapılandırma seçenekleri için;

```
bash-2.05b# ifconfig xl0
xl0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> mtu 1500
        address: 00:01:02:b7:bf:26
        media: Ethernet autoselect (100baseTX full-duplex)
        status: active
        inet6 fe80::201:2ff:feb7:bf26%xl0 prefixlen 64 scopeid 0x2
        inet 10.0.0.1 netmask 0xff000000 broadcast 10.255.255.255
        inet 10.0.0.100 netmask 0xffffffff broadcast 10.0.0.100
```

Eğer ağ arabiriminizi ilk defa yapılandırıyorsanız bilgilerin kalıcı olması için bunları ***/etc/hostname.arabirim_adi*** şeklinde bir dosyaya aşağıdaki formatta kaydetmeniz gerekmektedir, yine xl0 arabirimimizden örnek verelim; bu arabirime ait yapılandırma dosyası **/etc/hostname.xl0** dosyasıdır, bu dosyanın formatı aşağıdaki gibi olmalıdır
Adres_sinifi ip_adresi ağ_maskesi yayın adresi [diğer seçenekler]

Adres_sinifi **->**ipv4 için inet, ipv6 için inet6 kullanılır
ip_adresi ->atamak istediğimiz IP adresiz
ağ_maskesi **->**Bulunduğumuz ağa uygun bir ağ maskesi
yayın_adresi **->** Bulunduğumuz ağa uygun bir yayın adresi
Diğer_seçeekler →eklenebilecek tüm seçenekler için **man 5 hostname.if** Komutunu girmemiz yeterlidir.

IP adresini DHCP sunucudan aldırmak
IP adresini elle verebileceğiniz gibi ortamda bulunan bir DHCP sunucudan da aldırabiliriz, bunun için tek yapmamız gereken hangi arabirimin dhcp den ayarlarını alacağını belirlemek ve aşağıdaki komutu vermek(mesela benim örneğimde xl0 arabiri için dhcp kullanmak istiyorum)
bash-2.05b# echo dhcp >/etc/hostname.xl0

Yönlendirme Tablosunu okumak

bash-2.05b# route show
Routing tables
Internet:
Destination Gateway Flags
default 212.174.108.161 UG
10.0.0.0 link#2 U
..
127.0.0.0 localhost.0.0.127. UG
localhost.sanalm localhost.0.0.127. UH
212.174.108.160 link#3 U
212.174.108.161 0:b0:c2:88:d9:e6 UH
ya da netstat komutuna –rn parametresi vererek okuyabiliriz
bash-2.05b# netstat -rn
Routing tables
Internet:
Destination Gateway Flags Refs Use Mtu Interface
default 212.174.108.161 UGS 7 12238496 - rl0
10/8 link#2 UC 0 0 - xl0
10.0.0.200 0:0:21:c7:a7:32 UHL 0 11837 - xl0
127/8 127.0.0.1 UGRS 0 0 33224 lo0
127.0.0.1 127.0.0.1 UH 0 5393 33224 lo0
212.174.108.160/28 link#3 UC 0 0 - rl0
212.174.108.161 0:b0:c2:88:d9:e6 UHL 1 144 - rl0
224/4 127.0.0.1 URS 0 0 33224 lo0

Ağ arabirimine alias tanımlamak

Ağ arabirimine alias tanımlamak bir ağ arabirine birden fazla IP adresi eklemek demektir, OpenBSD de Linux lardaki alias tanımlamalarından farklı bir tanımlama şekli mevcuttur,ufak bir hatırlatma ile linuxlarda nasıl IP aliasing yapıldığına bakalım sonra bunun openBSD de nasıl yapıldığına bakalım;
Linux için;
Ilk Ethernet kartına alias tanımlayarak 2.bir IP adresi ekleyelim
[root@yubam ssh]# ifconfig eth0:0 192.168.21.1 netmask 255.255.255.255

openbsd için;

bash-2.05b# **ifconfig xl0 inet alias 10.0.0.100 netmask 255.255.255.255**
xl0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> mtu 1500
address: 00:01:02:b7:bf:26
media: Ethernet autoselect (100baseTX full-duplex)
status: active
inet6 fe80::201:2ff:feb7:bf26%xl0 prefixlen 64 scopeid 0x2
inet 10.0.0.1 netmask 0xff000000 broadcast 10.255.255.255
à **inet 10.0.0.100 netmask 0xffffffff broadcast 10.0.0.100**
à
bash-2.05b# ifconfig xl0 delete
komutu ile de xl0 arabirimine eklenmiş alias I silebiliriz.

Eklenen alias ların kalıcı olmasını sağlamak için ilgili arabirimin konfigürasyon dosyasına yazmamız gerekmektedir, xl0 için bu dosya /etc/hostname.xl0 dır ve yazım şekli aşağıdaki gibi olmalıdır.
inet alias 10.0.0.100 255.255.255.255

dosyanın son hali aşağıdaki gibi olmalıdır,bundan sonra ekleyeceğimiz her aliası bu satırların altına ekleyebiliriz
bash-2.05b# cat /etc/hostname.xl0
inet 10.0.0.1 255.0.0.0 NONE
inet 10.0.0.100 255.255.255.255 NONE

değişikliklerin aktif olabilmesi için sisteminizin yeniden başlatılması gerekir, yeniden başlatılmadan aktif hale geçirmek istiyorsak
bash-2.05b# **ifconfig xl0 inet alias 10.0.0.100 netmask 255.255.255.255**
komutunu vermemiz gerekiyor.
Arabirimlere atanan aliasların ifconfig komutu ile görünebilmesi için –A parametresi ile birlikte kullanırız,
bash-2.05b# ifconfig -a
xl0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> mtu 1500
address: 00:01:02:b7:bf:26
media: Ethernet autoselect (100baseTX full-duplex)
status: active
inet6 fe80::201:2ff:feb7:bf26%xl0 prefixlen 64 scopeid 0x2
inet 10.0.0.1 netmask 0xff000000 broadcast 10.255.255.255
bash-2.05b# ifconfig –A
xl0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> mtu 1500
address: 00:01:02:b7:bf:26
media: Ethernet autoselect (100baseTX full-duplex)
status: active
inet6 fe80::201:2ff:feb7:bf26%xl0 prefixlen 64 scopeid 0x2
inet 10.0.0.1 netmask 0xff000000 broadcast 10.255.255.255
inet 10.0.0.100 netmask 0xffffffff broadcast 10.0.0.100
ya da ifconfig komutuna parametre olarak arabirim ismini veririz.
bash-2.05b# ifconfig xl0 gibi.

OpenBSD'yi Gateway olarak kullanmak

OpenBSD nin bize sağladığı özelliklerden biri de ÇıkışKapısı(gateway) olarak kullanılabilme özelliğidir. Bu özelliğini aktif hale getirip kullanabilmek için aşağıdaki işlemleri gerçekleştirmemiz gerekir.
/etc/sysctl.conf dosyasını herhangi bir editörle açarak
net.inet.ip.forwarding= satırını bulup karşısına 1 yazıyoruz
sonra değişikliklerin aktif hale gelebilmesi için makineyi yeniden başlatıyoruz, makineyi yeniden başlatmadan ip.forwarding özelliğini aktif hale getirebilmek için ise aşağıdaki komutu vermemiz gerekmekte,

bash-2.05b # sysctl -w net.inet.ip.forwarding=1
Note: Bu komutla aktif hale gelen "ip.forwarding" özelliği makinenin bir sonraki yeniden başlama süresine kadardır,yani makinemiz yeniden başlayınca bu komutu tekrar vermemiz gerekmekte.

OpenBSD PF Uygulamaları

OpenBSD 3.0 ve sonrası sürümlerde Paket filtreleme, NAT işlemleri ve bandwith kontrolu için PF(PacketFilter) adı verilen bir yazılı m kullanmaya başlamıştır.Ondan önceki versiyonlarda IPF kullanılmaktaydı ve 3.0 versiyonu ile birlikte önceki firewall uygulamalarının tüm desteği kaldırılmıştır.

PF her nekadar syntax yapısı olarak IPF e oldukça benzer gözüksede IPF taban alınarak değil de **Daniel Hartmeier** tarafından sıfırdan yazılmıştır,

PF'i her açılışta aktif hale getirmek için /etc/rc.conf dosyasına aşağıdaki satırı eklememiz ve sistemi yeniden başlatmamız yeterlidir.
PF=YES

Aynı şekilde PF i sistemi yeniden başlatmadan aktif hale getirebilmek için
Komut satırından

#pfctl –e yazmamız yeterlidir.
-e →enable
aktif olan PF i pasif hale getirmek için ise

#pfctl –d yazmamız gerekiyor.
-d →disable

NOT:Bu komutlarla sadece PF aktif ya da pasif etmiş oluyoruz aktif ettiğimizde otomatik olarak kural tablosunu okumaz!!

Kural tablosu: PF birçok modern Firewall gibi kurallarını syntax yapısı belirli bir dosyadan okuyarak çalışır. Bu sebeple PF I aktif hale getirmemiz bizim PF I kullanabilmemiz manasına gelmez, PF'I kullanabilmek için kural tablosunuda yüklemeliyiz.

Bunu da yine /etc/rc.conf dosyasına
pf_rules=/etc/pf.conf satırını ekleyerek çözebiliriz. (Büyük ihtimalle /etc/rc.conf dosyasında bu satır ekli haldedir)
PF varsayılan olarak kural tablosunu **/etc/pf.conf** dosyasından okur fakat bunu değiştirme imkanımız vardır.PF.CONF dosyası 7 bölümden oluşur ,bunlar;
Makrolar:
Tablolar
Seçenekler
Scrubs
Kuyruga alma
NAT ve REDIRECTION
Filtreleme
Olarak sıralanabilir.

pf.conf dosyasındaki boş satırlar pf tarafından önemsenmez ve # ile başlayan satırlar yorum satırı olarak Kabul edilir.

Paket Filtreleme Yönetim Programı

Tüm paket filtreleme olayları pfctl adı verilen program aracılığı ile gerçekleştirilir.

Bazı Pfctl komutları ve açıklamaları;
#pfctl –f /etc/pf.conf pf.conf dosyasını okuyarak kural tablosunu yükler

#pfctl -nf /etc/pf.conf sadece dosyayı syntax olarak denetle, yükleme

pfctl -Nf /etc/pf.conf Sadece NAT Kurallarını yükle

pfctl -Rf /etc/pf.conf Sadece Filtreleme Kurallarını yükle

pfctl -sn ya da –s nat Anlık NAT kurallarını gösterir

pfctl -F rules paket filtreleme kurallarını kaldırır

pfctl -s rules anlık kural listesini görebilmek için

scrub in all fragment reassemble
pass in inet proto tcp from any to 10.1.1.4 port = www
pass in inet proto tcp from any to 10.1.1.4 port = https
...
pfctl -s –v rules daha detaylı bir görünüm için

block drop in on xl0 inet all
[Evaluations: 1 34884 2 Packets: 21 3 Bytes: 16330 4 States: 0
]
..

bash-2.05b# pfctl -s nat
nat on rl0 inet from 10.0.0.0/8 to any port < 1024 -> 212.174.108.169
nat on rl0 inet from 10.0.0.0/8 to any port < 1443 -> 212.174.108.169
rdr on xl0 inet proto tcp from any to any port = ftp -> 127.0.0.1 port 8021
rdr on rl0 inet proto tcp from 195.174.11.167 to 212.174.108.169 port = 1433 -
10.0.0.4 port 1433

#pfctl –z pfctl tarafından tutulan sayaçları sıfırlar

pf: rule counters cleared

Listeleme
Bazen bir kuralı birden fazla IP adres için yazmamız gerkebilir bu durumda her ip için bir kural tanımlamamız gerekir işte listeler burada devreye girerek bize istediğimiz sayıda ip adresini ya da port numarasını belli kategorilere koyarak kullanma işlevi sağlar,mesela YASAK_IP diye bir liste tanımlayıp içerisine sistemimize erişmesini istemediğimiz ip adreslerini tanımlayarak sonradan bu tanımlamayı kurallarımız içerisinde kullanabiliriz.

Bu listede tanımlayabileceğimiz kıstaslar protocol listesi, port numarsı listesi, ip adresi listesi vb olabilir.

Pfctl bu şekil bir listeli kurala rastladığında onu tek tek kabul ederek işleme sokar. Mesela,

block out on rl0 from { 192.168.0.1, 10.5.32.6 } to any
satırı işleme sokulurken
block out on rl0 from 192.168.0.1 to any
block out on rl0 from 10.5.32.6 to any
şeklinde düşünülür.Yine aynı şekilde bir kural içerisinde birden fazla listeleme kullanılabilir
,
**block out on rl0 proto { tcp udp } from { 192.168.0.1, **
10.5.32.6 } to any port { ssh telnet }
şeklinde.

Tanımlamalar(Macros)
Tanımlamalar programlamadaki değişkenler misalidir, bilgisayar dilindeki bir veriyi kendi anlayacağımız halde ifade edebilmemize kolaylık sağlarlar.

Mesela, firewallumuzun dışarıya bakan ayağı rl0 olsun biz bunu kural listemizin daha anlaşılır olabilmesi için bir makro tanımlayarak dis_ag= "rl0"diyebiliriz, bunun sağlayacağı en büyük yaralardan biri de, gun gelipte Ethernet kartımızı değiştirmemiz gerektiğinde tüm kural setini baştan sonra değiştirmek yerine sadece tanımladığımız makroyu değiştirerek işimizi halletmiş oluruz.

Ic_ag="fxp0"
Dis_ag="rl0"

pass in on $dis_ag from any to any

makroları kurallar içerisinde kullanırken belirteç olarak $ işaretini kullanırız.

Paket filtreleme

Paket filtreleme işlemi networkumuze giren ve çıkan peketler üzerinde detaylı bir denetim kazandırır.Layer 3(IPv4 and IPv6)
) ve layer 4(TCP, UDP, ICMP,
and ICMPv6) de paket filtreleme işlemi yapabiliyor. Çok sık kullanılan filtreleme seçenkleri, hedef ip-port, kaynak-ip port ve protocol seviyelerine göredir.

Paket filtreleme de kurallar baştan sona doğru okunur ve en son uyan kural kazanır felsefesi geçerlidir. Yani ilk kural da bir ip adresine geçiş verdiyesniz sonra ki bir kural da durdurduysanız sonraki kural geçerlidir. Bunun geçersiz olduğu durum **quick** kullanımıdır, kuralın sonudan geç kelimesini gördüğü zaman ondan sonraki kurallara bakmaz ve o kuralı eğer uygunsa uygulmaya alır.

Kural sözdizimi

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

action→ kurala uyan pakete uygulanacak olaydır. Bunun için 2 seçeneğimiz var, pass ya da block yani geçir ya da durdur. **block drop** ve **block return** olmak üzere farklı kulanımları da vardır. **pass** ile geçen paket bir sonraki işleme dahil olur, quick ile geçen ya da durdurulan paketler sonraki işlemlere dahil olmazlar!

Direction →arabirim üzerinden içeri ya da dışarı yönlendirilecek paketler için kullanılır

Log → pflogd tarafından loglanacağını belirtiriz. Tüm paketleri loglamak için log-all seçeneğini kjullanmamız gerekmektedir.

Quick → eğer paket bu kurala uygunsa bu kuralı o paket için son kural olarak belirle. yani bundan sonra işleme sokma demektir.

Interface → paketin içinden geçtiği ağ arabiriminin ismi.

Af → Address family ,IPV4 için inet Ipv6 için inet 6 kullanılır, PF genellikle bunu hedef ya da kaynak ip adresine bakarak tahmin edebilir.

Protocol→
4.Katman Protokolleri
+ tcp
+ udp
+ icmp
+ icmp6
+ /etc/protocols dosyasında listelenmiş herhangi bir protokol

src_addr, dst_addr
IP başlığında kullanılan kaynak ve hedef IP adresleri tanımı
+Tekil IPV4 ya da IPV6 adresi olabilir

+ CIDR bloğu olabilir
+ DNS ile çözümlenebilecek bir tam tanımlanmış konak ismi

src_port, dst_port

4. Katman kaynak/hedef port numarası

aşağıdaki şekillerde belirtilebilir
+ 1 ve 65535 arasında herhangi bir değer
+ /etc/services dosyasında bulunan herhangi bir servis ismi
+ port tanımlamalarıda kullanılabilecek bazı eşitlikler
  - o != (eşit değil)
  - o < (kuçuktur)
  - o > (buyuktur)
  - o <= (küçük eşittir)
  - o >= (büyükeşittir)
  - o >< (Aralık)
  - o <> (ters aralık)

Bu kadar teorik bilgi verdikten sonra gerçek bir uygulama üzerinde bu teorik bilgilerimizi uygulayalım.

OpenBSD üzerinde Firewall Uygulamaları

Bir Firewallun sağladığı güvenlik üzerinde çalıştığı platform kadar olabilir, örnek olarak Checkpoint INC. firmasının çıkardığı Firewall ürünü tüm dünyada geçerliliği kabul edilmiş en iyi en güvenli ürün* olarak bilinmektedir.

Fakat bu ürün Windows ya da unix sistemleri üzerinde çalıştığından sağlayabileceği max güvenlik bu sistemlerin sağladığı güvenlik kadardır diyebiliriz, buradan da Firewall un çalıştığı sistemle bir bütün oluşurduğu ve bir zincirin halkası gibi birbirine bağımlı olduğunu görüyoruz. Zincir en zayıf halkasından kopar sözünü anımsatma gereği duyuyorum.

Firewall tasarımında karar vermemiz gereken ilk hususlardan biri de açık sistem mi kapalı sistem mimarisini mi uygulayacağımızdır. Açık ve kapalı system mimarilerinden kastımız şudur;
Açık sistemde varsayılan olarak tüm portlar tüm adreslere açıktır ve biz bunlar arasından istemediğimiz trafiği yasaklarız.

Kapalı sistemde ise varsayılan olarak tüm portlar tüm adreslere kapalıdır biz ihtiyacımıza göre bunlara gerekli izinleri veririz. Bence kapalı sistemin tercih edilmesi daha uygundur , çünkü açık sistemde tüm ayrıntıları en ince noktasına kadar düşünmeniz gerekmektedir kaçıracağınız ufak bir ayrıntı firewall kurallarınızın işlevseilliğini yitirmesine sebep olacaktır.

Kapalı sistem mimarisini uygulamak için
Normal kurulum sonrasında çalıştırılan PF öntanımlı olarak gelen paketlere "geç" politikası uygular, yani gelen/giden bir paket için herhangi uygun bir kural yoksa paketi geçirir.
/etc/pf.conf dosyasının başına aşağıdaki satırları ekleyerek bu politikayı değiştirebiliriz.

block in all
block out all

bu kural tanımlamaları ile sistemdeki her arabirime giren ve çıkan paketleri yasaklamış olduk.

Yukarıda tanımladığımız kurallar ile sistemimizi tamamen dış ortamdan yalıtmış olduk, şimdi ise dikkatlice düşünerek hangi arabirimlerden hangi paketlerin gidiş gelişine izin vereceğimiz kararlaştırıp kurallarımızı yazmaya başlayalım.

İç ağımızdan Firewall makinesine gelen trafiğe izin verelim , sonra aynı şekilde firewall makinesinden iç ağımıza giden trafiğe de izin verelim.
İlk kuralla 192.168.0.0/24 olarak kullandığımız yerel ağımızdan 192.168.0.1 IP sine sahip olan Firewall makinemize geçişe izin verelim.
pass in on dc0 from 192.168.0.0/24 to 192.168.0.1

pass	->> pakete uygulanacak action
in	->> gelen trafik
on dc0	->>hangi ağ arabirimine işlem yapılacağı
from 192.168.0.0/24	->>hangi aralıktan kabul edileceği
to 192.168.0.1	->>hangi IPye ya da IP aralığına kabul olacağı.

Şimdide OpenBSD makinemizin diğer ağ arabirimne gelen paketler için bir kural yazalım, OpenBSD makinemizin üzerinde bir adet HTTPD sunucu çalıştığını varsayarak bu kuralı yazıyorum.

Önce kuralı Türkçe olarak düşünelim sonra PF ın anlayacağı dilde kuralımızı yazalım.
Herhangi bir IP adresinden ve hedef IP si bizim internete bakan ağ arabirimizin ip si olan, protkol olarak TCP ve hedef portu 80 olan paketler kabul edilsin.

Dışa bakan ağ arabirimizin adı xl0

pass in on xl0 proto tcp from any to fxp0 port www

Son uyan kazanır kuralı!

PF kural işleyiş yapısından bahsederken kuralların en düşük numaralı kuraldan başlayarak en yüksek sayılı kurala doğru ilerlediğini ve son uyan kuralın kazandığından bahsetmiştik, yani

1 nolu kural da da 80.porta gelen istekleri kabul etmiyoruz
2. nolu kuralda 80.porta gelen istekleri kabul ediyoruz
3. nolu kuralda tekrar kabul etmiyoruz burada son kural geçerlidir .

Bu işleyişte harici bir durum söz konusur, bu da quick kelimesinin kullanıldığı kural dizisidir.
Aşağıdaki örneğe bakarak durumu daha iyi anlamaya çalışalım

block in on xl0 proto tcp from any to any port ssh
pass in all

burada büyük bir hata söz konusudur, sistem adminimiz ilk kural da xl0 arabirimi üzerinden tcp 22.porta gelen tüm istekleri yasakladığını düşünüyor sonraki kural da da tüm arabirimlerden gelen tüm paketleri kabul et diyor, yukarıda bahsettiğimiz gibi son uyan kural kazanacaktır ve bir üst kuralın hiçbir yararı olmayacaktır ve dışarıdan herhangi birisi SSH kullanabilecektir.

Şimdide istisna olacak durumu inceleyelim , yine aynı örnek üzerinden devam edelim ve bu sefer in ile on arasında quick kelimesini yerleştirelim

block in quick on xlp0 proto tcp from any to any port ssh

pass in all

şimdiki durumda ssh portumuz dışarıdan gelecek tüm isteklere kapalı olacaktır. Bunu sağlayan **quick** kelimesidir.

Ek:
TCP bayrakları
* F : FIN – bağlantıyı sonlandırmak için
* S : SYN - Synchronize; indicates request to start session
* R : RST - Reset; drop a connection
* P : PUSH - Push; packet is sent immediately
* A : ACK - Acknowledgement
* U : URG - Urgent
* E : ECE - Explicit Congestion Notification Echo
* W : CWR - Congestion Window Reduced

Gerçek Bir Uygulama Örneği

Aşağıda göreceğiniz Firewall kural tanımlamaları gerçek hayatta aktif olarak kullanılan bir OpenBSD Firewall dan alınmıştır.

```
reserved="{ 0.0.0.0/7, 0.0.0.0/8, 2.0.0.0/8, 5.0.0.0/8, 20.20.20.0/24, \
23.0.0.0/8, 27.0.0.0/8, 31.0.0.0/8, 67.0.0.0/8, 68.0.0.0/6, 72.0.0.0/5,
80.0.0.0/4, \
96.0.0.0/3, 127.0.0.0/8, 128.0.0.0/16, 128.66.0.0/16, 169.254.0.0/16,
172.16.0.0/16, \
191.255.0.0/16, 192.0.2.0/24, 197.0.0.0/8, 201.0.0.0/8, 204.152.64.0/23, \
224.0.0.0/3, 240.0.0.0/4 }"
```

reserved adı ile bir macro tanımlayarak ,yerel alan ağları için kullanıma açılmış IP adresi aralıklarını tek bir grupta toparladim. Amacım gerektiğinde her bir ip aralığı için aynı kural tanımlamalarını tekrar etmemek.

Sonra benzer şekilde

```
SAFEPORTS="{ 21, 22, 25, 43, 53, 80, 110, 443, 1433,5802,5902,6001,6002,8181  }"
```

Bir macro tanımlayarak istemcilerin kullanabileceği portları bir gruba aldım.
Bizim iç ağda kullandığımız ip aralığı 10.0.0.0/24

Kurallar

nat on rl0 from 10.0.0.0/24 to any port < 1024 -> 22.14.18.12

Yukarıdaki komutla , rl0 arabirimi için 10.0.0.24 ağından gelip herhangi bir IP adresine giden ve hedef port numarası 1024 den küçük olan paketlerin dışarıya hangi IP ile çıkış yapacaığını belirlemiş oluyoruz. Bu komutla iç ağdaki istemcilerimizin internetteki herhangi bir host un 1024 altı portlarına bağlantısını sağladık

İstemcilermizden birinin bir başka şirkette tutulan SQL sunucuya veri aktarması gerektiği için ozel olarak sql-data portunu da açmamız gerekiyor. Bunun için sadece o porta özel bir nat kuralı ekliyoruz

nat on rl0 from 10.0.0.0/24 to any port 1433 -> 22.14.18.12

Yine benzer şekilde dışarıda bulunan Windows 2000 Sunucumuzu Terminal server aracılığı ile yönetebilmemiz için Terminal Clientin sunucuya bağlanırken kullandığı 3389.portu da açmamız gerekiyor, bunu da aşağıdaki nat kuralı ile oluşturuyoruz.

nat on rl0 from 10.0.0.0/24 to any port 3389 -> 22.14.18.12
İçeride Bulunan Terminal Service makinesine dışarıdan ulaşıp yönetebilmek için , rdr ile nat kuralı yazıyoruz.Burada belirtmem gerek durum iç ağdaki terminal sunucu makinesinin IP adresi 10.0.0.2 ve varsayılan porttan Terminal Service hizmeti veriyor.Bizim internete bakan dış bacağımızın ip adresi ise 22.14.18.12.

rdr on rl0 proto tcp from any to 212.174.108.162 port 3389 -> 10.0.0.2 port 3389

10.0.0.2 IP li makine üzerinde çalışan Damwware adlı uzaktan kontrol programının dışarıdan erişimini sağlamak için de aşağıdaki kuralı yazıyoruz, bu kural kısaca 22.14.18.11/32 adresinden 22.14.18.12 makinesine gelen 6129 port numaralı istekleri 10.0.0.2 IP li makinin 6129 nolu portuna iletmektedir.

#Dameware Remote Control Access
rdr on rl0 proto tcp from 22.14.18.11/32 to 22.14.18.12/32 port 6129 ->
10.0.0.2 port 6129

#--#
#----------------------------- FILTERING RULES------------------#
#---
#
NMAP ve benzeri Port tarama araçlarını etkisiz kılmak için aşağıdaki kural tanımlamalarını giriyoruz.
#--#
NMAP rules - port scan blocking
#--#
block in log quick on rl0 proto tcp from any to any flags FUP
block in log quick on rl0 proto tcp from any to any flags SF/SFRA
block in log quick on rl0 proto tcp from any to any flags /SFRA

Şimdide loopback arayüzünden geçiş yapacak her pakete izin verelim.
#--#
Handle everything from loopback, I am considering trusted
#--#
pass in quick on lo0 all
pass out quick on lo0 all

Yukarıda tanımladığımızı RESERVED grubunda bulunup kendini internetten geliyormuş gibi gösteren tüm paketleri engellememiz lazım,sebebi açık bu ip aralıkları sadece iç ağlarda kullanılmak üzere tasarlanmıştırlar ve internette kullanılamazlar.
block in log quick on rl0 from $reserved to any
block in log quick on rl0 from any to $reserved

#--#
Incoming traffic on rl0 (EXTERNAL)

#--#
SSH
SSH Sunucuyu kullanıma açmak:
Internet üzerinde herhangi bir IP den 22.14.18.12 nin 22.portuna gelen istekleri durum korumasına tabii tutarak Kabul et.

pass in quick on rl0 proto tcp from any to 22.14.18.12 port = 22 flags S/SA
keep state

Huzeyfe ÖNAL huzeyfe[at]cc.kou.edu.tr

Kaynaklar:

http://www.openbsd.org
http://undeadly.org
Absolute OpenBSD: UNIX for the Practical Paranoid (by Michael W. Lucas)
Building Firewalls with OpenBSD and PF, 2nd Edition

TODO:
Nat\rdr işlemleri eklenecek