IP MUHASEBESİ

IP Tables ve RDTool'a Uygulamalı Giriş

Can Burak ÇİLİNGİR
canb@canb.net

Bu yazıda Linux dağıtımlarının son versiyonlarında bulabileceğiniz IPTables ile ağımızda dolaşan paketler hakkında temel istatistiksel bilgiler toparlayıp, RRDTool ile grafik halinde nasıl sunulabileceğini anlatmaya çalışacağım. Bu sistemin üzerine Cron'un yeteneklerini de ekleyip sonucu biraz da olsa hareketlendireceğiz. Gerek IPTables'in, gerekse RRDTool'un bu yazıda bahsedilenlerden daha geniş olduğunu unutmayın! Bahsettiğim 3 araç hakkında biraz bilgi verdikten sonra, birleştirip voltranı oluşturmaya başlayacağız. Temel ağ terimleri hakkında fikriniz yoksa, referanslardan ağ temelleri altındaki linkleri ziyaret etmenizi öneririm.

Okumaya başlamadan önce IPTables ve RRDTool'u kurmayı unutmayın. Debian kullanıcısıysanız tek bir komut ile kurabilirsiniz:

```
apt-get install iptables rrdtool
```

Dağıtımınız için önceden derlenmiş paketleri edinmeniz yükleme aşamasını hızlandıracaktır. Eğer bulamazsanız kaynak paketlerini çekerek içerilerindeki "INSTALL" dosyasını okuyarak kurabilirsiniz.

IPTables

IPTables, Linux 2.4 ve 2.6 içerisinde paket yönetimi saglayan netfilter yapısının kullanıcı düzeyindeki yönetim aracıdır. Basitçe çekirdeğin paket filtreleme tablosuna kuralallar ekleyip çıkartmaya yarar. Çoğunluğun, IPTables'ın sadece engelleyici olduğu yönünde yanlış olmasa da eksik düşünceleri var. Engelleyici olmanın yanı sıra, paket iletme, paket içerikleri ile oynama gibi klasik özellikleri sunması ile birlikte patch-o-matic adı altında toplanmış yamalar ile de ilginç yetenekler kazandırılabilmekte. Daha fazla bilgiyi referanslarda patch-o-matic linkinde bulabilirsiniz.

Tablolar

Kuralları işleyecekleri durumlara göre belli tabloların içerisinde yaratırız. Varsayılan sistemde 3 tablo vardır.

FILTER tablosuna sistem tarafından üretilen, sistemden geçen, ya da sisteme adreslenmiş paketleri yönetebileceğimiz kuralları ekleriz. "--table" parametresi ile bir tablo belirtmediğimizde kurallar bu tabloya eklenir.

Açılımı "Network Adress Translation" olan NAT tablosuna paketlerin hedef ve/veya kaynak adreslerini değiştirecek kurallar ekleriz. Bu tabloyu kullanarak port yönlendirme, IP Masquerading gibi uygulamalar yapabiliriz. Yöntemlerin açıklaması için Tablo 1'e bakabilirsiniz.

MANGLE tablosunu paketlerin içeriğini değiştirmek için kullanırız. NAT içerisinde de paketler değiştiriliyordu, fakat yapılan değişikliklerin sonradan geri alınmak üzere kaydı da tutuluyordu. Mangle da yaptığımız değişikliklerin kaydı tutulmaz. Değişikliklerden kasıt, paketi başka araçlarla farkedebilmek için işaretlemek, ağ arabiriminden çıkış sırasında öncelik vermek gibi uygulamalardır.

Hedefler

Kurallar ile paketleri temel olarak 4 hedefe yönlendirebiliriz.

ACCEPT hedefine aktarılan

paket hedefine ulaşmaya hak kazandı demektir. Hedeften kasıt bulunulan yere göre makinadan çıkış hakkı kazanma, yönlendirme hakkı kazanma gibi durumlar olabilir.

DROP hedefine erişen bir paket görmezden gelinir. Yerine göre sistemden çıkışı, sisteme girişi ya da yönlenmesi engellenir.

RETURN hedefine iletilen paket, bulunduğu zincirin varsayılan hedefine aktarılacaktır. Varsayılan hedef ile ilgili bilgiyi Zincirler başlığında bulabilirsiniz. Eğer ki kural başka bir zincirin içerisindeki bir zincirdeyse paket üst zincire döndürülür.

REJECT hedefi de DROP hedefi gibi paketin görmezden gelinmesini sağlar. DROP'tan farklı olarak paketin kaynak adresi bu engellemeden haberdar edilir.

Zincirler

Zincirler, kuralların arka arkaya eklendiği temel kural saklama gruplarıdır. Paketin ne tarafa iletileceği karar verildiğinde o zincirin ilk kuralına bakılacak, eğer kurala takılıyorsa ilgili işleme tabi tutulacaktır.

Her zincir, paket içerisindeki kuralların herhangi birine takılmadığında işleyecek varsayılan davranışa sahiptir. Kuralların hepsinden kazasız belasız geçen paket, zincirin varsayılan davranışına tabi tutulur.

INPUT zincirinden sisteme adreslenmiş paketler geçmeye çalışır.

OUTPUT zincirinden sistem tarafından üretilip dışarı çıkmak isteyen paketler geçer.

FORWARD zıncırından sisteme uğrayan paketler geçer. Bu gibi durumlarda sistem genellikle yönlendirici (router) konumundadır.

Kullanıcı Zincirleri

Kullanıcı zincirleri geniş ölçekli ve karmaşık kurulumlarda kuralların belli ölçütlerde gruplandırılmasını, dolayısı ile tekrarı engelleyen, yönetimi kolaylaştıran kural gruplarıdır. Yazımızın ilerisinde verileri daha rahat elde etmek amacı ile kendi zincirlerimizi tanıtacağız.

Örnek Komutlar

Basitçe teoriden bahsettikten sonra sıra birkaç örneğe geldi. Tahmin edebileceğiniz üzere komutumuzun adı "iptables". Bu bölümde kuralları önce Türkçe ifade edip ardından bunu iptables'a tercüme edeceğim. Her zaman olduğu gibi "man iptables" yazarak tüm parametreleri gözden geçirebilirsiniz.

Filter tablosunun, INPUT zincirinin varsayılan davranışı DROP olsun:

```
iptables --table filter -- policy INPUT DROP
```

192.168.0.123 adresinden gelen her şeyi kabul et:

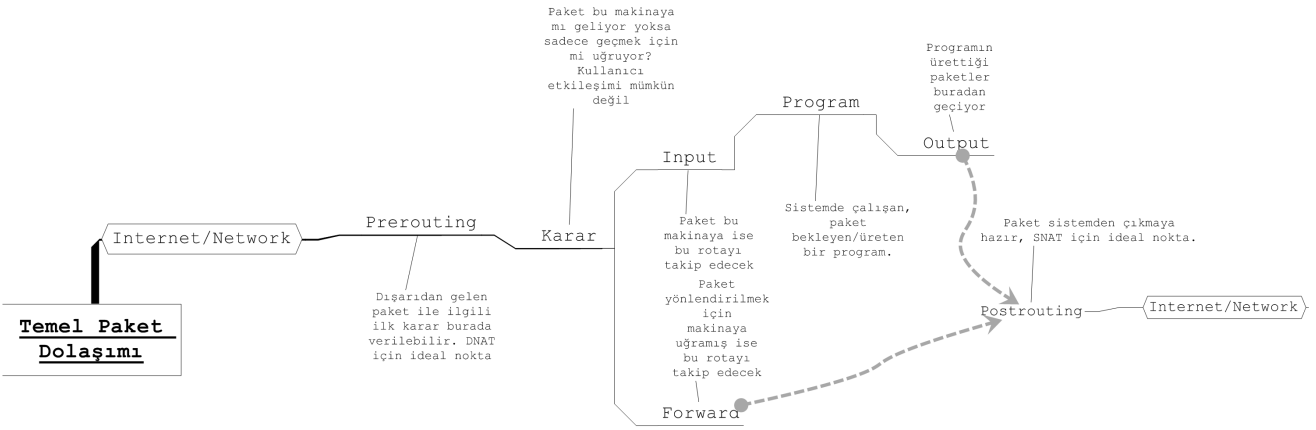
```
iptables --table filter --append INPUT --source
192.168.0.123 --jump ACCEPT
```

3684 numaralı portuma gelen her şeyi 192.168.0.123 adresli makinanın 3451 numaralı portuna ilet:

Tablo 1

Port yönlendirme: Bu yöntemle sistemin x numaralı portuna gelen bağlantı isteklerinin y numaralı bir porta ulaşmasını sağlayabiliriz. y numaralı port aynı makina üzerinde olacağı gibi başka bir makinada da olabilir.

IP Masquerading: Yöntemin temel amacı IPv4 adres alanının yetersizliğine bir çözüm üretmektir. Üst ağ tarafından yönlendiriciye atanmış ip adresleri, eğer ki o ağda bulunup üst ağa çıkması gereken makina sayısından az ise, ağdaki sistemlere üst ağdan farklı ip adresleri atanır ve üst ağa gidecek paketler routera ulaştığında routera atanmış üst ağ adreslerinden biri paketin kaynağı olarak değiştirilir. Bu değişikliği not eden çekirdek, aynı akış içerisinde geri gelen paketlerde işlemi tersine çevirerek iki tarafından da bu değişimden etkilenmeden çalışmalarını sağlar.




```
iptables --table nat --append PREROUTING --
protocol tcp --dport 3684 --jump DNAT --to
192.168.0.123:4662
```

Deneme adında boş bir kullanıcı zinciri yarat:

```
iptables --table filter --new-chain deneme
```

Yeni zincir yarattıktan sonra paketleri başka bir zincirden --jump ile kendi zincirimize geçirebiliriz. Örneğini yakında göreceğiz.

IP Muhasebesi

Sıra sistemden gecen her paketi tek tek sayabilmemiz için çeşitli süzgeçler kurmaya geldi. Bu süzgeçlere takılan paketlerin çeşitli özelliklerine bakıp bir kenera not edeceğiz. Kolaylık olması açısından varsayılan zincirler ile fazla haşır neşir olmadan kendi zincirlerimizi yaratıp istediğimiz "süzgeçleri" o zincirler içine ekleyelim.

Derli toplu bir bakış açıcı edinebilmek için kendimize bir hesap defteri açalım. IPTablesda bu hesap defteri denen şeyi ancak bir kullanıcı zinciri ile elde edebiliriz. Az once örneklerde bahsettiğimiz şekilde 2 yeni zincir yaratalım:

```
iptables --table filter --new-chain net-kullanici
```

Bu zincirde, Internet'ten kullanıcıya gelen paketlerin hesabını tutacağız.

```
iptables --table filter --new-chain kullanici-net
```

Bu zincirde ise kullanıcıdan Internet'e giden paketlerin hesabını tutacağız.

İlgili zincirlerimiz hazır olduğuna göre sıra tutacağımız veriyi belirlemeye geldi. Tercihim port numarası bazında yakalamak. "www" ve "ssh" verisi benim için önemli, o yüzden bu verilerin takılacağı kuralları zincirlerimize ekleyelim (RETURN hedefinin ne işe yaradığını anımsıyor musunuz?):

Önce "www" için:

```
 iptables --table filter --protocol tcp --append
net- kullanici --source-port www --jump RETURN
 iptables --table filter --protocol tcp --append
kullanici-net --destination-port www --jump RETURN
```

Ardından "ssh" için:

```
 iptables --table filter --protocol tcp --append
net- kullanici --source-port ssh --jump RETURN
 iptables --table filter --protocol tcp --append
kullanici-net --destination-port ssh --jump RETURN
```

Farkındaysanız portları numaraları ile yazmadım. Bu hali bana daha insancıl geliyor. /etc/services dosyasına gözatıp www ve ssh'ın neye karşılık geldiğini görebilirsiniz. Komutları yazdık fakat ya ileride daha çok port gözetlemek istiyorsak? Her port için iki satır mı ekleyeceğiz? Tabi ki hayır. Bir döngü bizim için bu işi yapacak! Nasıl mı? Öncelikle şu bash kod parçasını çalıştırıp deneyin (tek satıra yazın):

```
for port in www ssh;do echo $port; done
```

Ekranda sıra ile www ve ssh yazdığını göreceksiniz. Hadi bunu yukarıdaki 2 satıra uygulayalım.

```
 for port in www ssh; do iptables --table filter --
 protocol tcp -append net- kullanici --source-port
$port --jump RETURN;iptables --table filter --
protocol tcp --append kullanici-net --destination-
port $port -jump RETURN; done
```

Artık eklemek istediğiniz portları "in" den sonra yazabilirsiniz.

Zincirimize kuralları ekledik. Peki ya zincirimize paket gelecek mi? Biz söylemeden maalesef gelmeyecek:

```
 iptables --table filter --insert INPUT --jump net-
kullanici
 iptables --table filter --insert OUTPUT --jump
kullanici-net
```

"--insert" ile INPUT ve OUTPUT zincirlerinin en başına paketlerin bizim zincirlerimizden geçmesini sağlayacak emri verdik. Eğer ki makina yönlendirici konumunda ise FORWARD zincirine de aynı şekilde eklemek gerekecektir.

Artık IPTables www ve ssh portlarından giden gelen verinin kaydını tutacak.

IPTables bu verilerin kaydını tutacak. Peki ama biz nasıl göreceğiz? Öncelikle az önce belirlediğimiz portları kullanın. Mesela ssh ile bir yere bağlanın, web sitelerinde gezinin. Ardından "kullanici-net zinciri hakkında bilgi ver, boy bilgilerini de byte cinsinden ver" anlamına gelecek komutu çalıştırın:

```
 iptables --list kullanici-net --verbose -x
```

Benim sistemimde şuna benzer bir çıktı elde ediyorum:

```
 hopkins:~# iptables --list kullanici-net --
verbose -x
 Chain kullanici-net (1 references)
   pkts bytes target prot opt  in  ou destination
     19  3279 RETURN  tcp  -- any any    anywhere
anywhere tcp dpt:ssh
```

Gördüğünüz üzere bu kurala toplam "3279" byte boyunda paket takılmış. İsterseniz bu çıktıları ileride kullanmak üzere bir dosyaya koyalım ve bundan sonra o dosya üzerinde çalışalım:

```
 iptables --list kullanici-net --verbose -x >
kullanici-net
 iptables --list net-kullanici --verbose -x > net-
kullanici
```

Dosyalarımızı olusturduk. Şimdi az önce bahsettiğimiz "3279" u daha sonra kullanılabilir bir halde, yani bir bash değişkeni halinde elde etmeye çalışalım (knSSH "kullanici-net ssh"ın kısaltması):

```
 knSSH=`grep "dpt:ssh" kullanici-net | awk '{print
$2}'`
```

"echo $knSSH" yazdığınızda beklediğimiz sayıyı göreceğiz. Dağınık gelmiş olmalı. Yazının sonunda bu yaptıklarımızı bir betik haline getirip daha rahat kullanabileceğiz.

RRDTool

RRDTool yönlendirici gözlemlemede sıkça kullanılan MRTG adlı uygulamanın grafik çizme altyapısı olarak hayatına başlamış bir uygulamadır. Bu uygulama sayesinde istediğimiz her verinin grafiğini çıkartabiliriz. Biz IPTables sayesinde elde ettiğimiz verinin zamanla ne kadar değiştiğini gözlemek için bir grafik oluşturacağız. RRDToolun tam olarak nasıl çalıştığını anlatmak yerine sadece kullanacağımız komutların açıklamasını vereceğim. Referanslarda bağlantısı bulunan "tutorial"ı okuyarak diğer özellikleri hakkında fikir sahibi olabilirsiniz.

RRDTool kullanmaya başlamadan önce temel olarak nasıl çalıştığını bilmemiz gerekli. RRD "Round Robin Database"in kısaltması olduğundan işin içinde grafik çizen bir uygulamanın yanı sıra bir de veritabanı uygulaması bulunduğunu anlayabiliriz. Bu veritabanının bilinmesi gereken 3 temel özelliği var:

1.Eklenecek her veri bir tarih ile bağlantılı olmalı.

2.Veritabanındaki ayarlanmış yerler bittiğinde en eski verinin üzerine yazacaktır. Buna en güzel ornek bir çember olabilir. Bir noktadan başlarsınız, çemberdeki her yer dolduğunda başladığınız yere gelir ve ilk yazdığınız verinin üzerine yazarak devam edersiniz.

3.RRDTool veritabanları belli bir aralıkta veri bekler. Eğer ki veri gelmez ise o slotu bilinmeyen olarak işaretler.

Bu kısa girişten sonra işe ilk veritabanımızı yaratarak başlayalım. İlk veritabanımızda sisteme giren ssh ve www paketlerinin sayısı tutulsun:

```
rrdtool create sshwww.rrd \
         DS:ssh:ABSOLUTE:600:U:U \
         DS:www:ABSOLUTE:600:U:U \
         RRA:LAST:0:1:480
```

Burada ne denmek istiyor:

rrdtool create sshwww.rrd: sshwww.rrd adında

bir doya yarat

DS:ssh:ABSOLUTE:600:U:U: bu dosyada 600 saniyede bir güncellenecek, ssh adında bir alan yarat.

DS:www:ABSOLUTE:600:U:U: 600 saniyede bir güncellenecek, www adında bir alan yarat.

RRA:LAST:0:1:480: 480 adet kayıt tut, kayıtların her biri veri noktası olarak kullanılabilir. (1 yerine 2 olsa idi mesela 2 veride bir ortalama alınarak veri noktası oluşturulacaktı.) (rrdtool man sayfasını okumanızı hatırlatmama gerek var mı?)

Artık bir veritabanımız var. Sıra geldi içine bilgi eklemeye:

```
rrdtool update sshwww N:$knSSH:$knWWW
```

Gördüğünüz gibi değişkenleri veritabanında yarattığımız sıra ile verdim. knSSH ve knWWW değişkenleri ise yukarıda anlattığım IPTables verisinin değişkene atanması esnasında yaratılan değişkenler.

Veritabanımız da oluştu. Sıra geldi bir grafik çizmeye:

```
rrdtool graph sshwww.png \
    DEF:s=sshwww.rrd:ssh:LAST \
    DEF:w=sshwww.rrd:www:LAST \
    AREA:s#FF0000:s \
    STACK:w#00FF00:w \
    --title="Port kullanimi" > /dev/null
```

rrdtool graph sshwww.png: oluşturulacak dosyanın adı ssswww.png

DEF:s=sshwww.rrd:ssh:LAST: sshwww.rrd dosyasındaki ssh verisini s adı ile kullan

DEF:w=sshwww.rrd:www:LAST: sshwww.rrd dosyasındaki www verisini w adı ile kullan.

AREA:s#FF0000:s: s verisini alan olarak çiz, rengi kızmızı olsun.

STACK:w#00FF00:w: w verisini yeşil olarak bir önceki verinin üzerine çiz.

--title="Port kullanimi" > /dev/null: grafiğin başlığını belirle, çıktıyı görmezden gel (normalde oluşturulan grafiğin boyutu yazar)

Betikler

İşimize yarayacak betikleri oluşturmadan önce elimizde hazır olması gerekenler var. İstediğimiz gibi ayarlanmış bir kural tablosu ve verileri tutacak veritabanları (sshwww.rrd). Yukarıda bahsettiğimiz gibi bunları hazırladıktan sonra her şey betikleri oluşturmak için hazır olacak; fakat ufak bir detayı unutmamak lazım. Bu kurallar sistemi yeniden açtığımızda kaybolacaklar. Ayrıca kurallarla oynarken sil baştan yapmanız da gerekecek.

Her şeyi silip yeniden başlamak için:

```
#varsayilan zincirlerin varsayilan hedefini ayarla
iptables --table filter --policy INPUT
        ACCEPT
iptables --table filter --policy FORWARD
      ACCEPT
iptables --table filter --policy OUTPUT
       ACCEPT
iptables --table nat    --policy PREROUTING
   ACCEPT
iptables --table nat    --policy OUTPUT
       ACCEPT
iptables --table nat    --policy POSTROUTING
  ACCEPT
iptables --table mangle --policy PREROUTING
   ACCEPT
iptables --table mangle --policy OUTPUT
       ACCEPT
iptables --table mangle --policy INPUT
        ACCEPT
iptables --table mangle --policy FORWARD
      ACCEPT
iptables --table mangle --policy POSTROUTING
  ACCEPT
#kullanici zincirlerini sil
iptables -X
#tablolardaki tum kurallari sil
iptables --table filter --flush
iptables --table nat    --flush
iptables --table mangle -flush
```

Aktif kuralları kaydetmek için:

```
iptables-save > /root/kurallarim
```

Kaydedilmiş kuralları yüklemek için

```
iptables-restore < /root/kurallarim
```

Kuralların ayarlı olduğunu ve veritabanlarının hazır olduklarını varsayıp betiğimizin yapması gereken işleri listeleyelim:

1.Verileri oku, daha sonra çabuk elde etmek için dosyaya kaydet. (Komut her çalıştığında zaten üretiyor neden kaydedelim diyebilirsiniz. Pentium 166 makinamda 8 tane gözetleme kuralı olan tablonun verilerini edinmem 15 saniye sürüyor. Her seferinde bunu beklemek istemiyorum.)

2.Verileri degişkenlere ata.

3.Bu değişkenlerdeki verileri veritabanına kaydet.

4.Veritabanındaki verilerden grafik oluştur.

Betiğimiz şöyle olabilir:

```
 #!/bin/bash
 # Bu betik birazdan görecegimiz cron tarafından
çalıstırılacagı için
 # her seyi tam yerleri ile birlikte belirtmemizde
yarar var. Komutların
 # tam yerini bulmak için which komutuna
basvurabilirsiniz.
 # Örnek: "which grep":
 iptables=/sbin/iptables
 grep=/bin/grep
 awk=/usr/bin/awk
 rrdtool=/usr/bin/rrdtool
 kullanicinet=/tmp/kullanicinet
 netkullanici=/tmp/netkullanici
 #Adım 1:
 $iptables --list kullanici-net --verbose -x >
$kullanicinet
 $iptables --list net-kullanici --verbose -x >
$netkullanici
 $iptables -Z
 #Adım 2:
 knSSH=`$grep "dpt:ssh" $kullanicinet | $awk
'{print $2}'`
 nkSSH=`$grep "spt:ssh" $netkullanici | $awk
'{print $2}'`
 knWWW=`$grep "dpt:www" $kullanicinet | $awk
'{print $2}'`
 nkWWW=`$grep "spt:www" $netkullanici | $awk
'{print $2}'`
 #Adım 3:
 $rrdtool update sshwww N:$knSSH:$knWWW
 #Adım 4:
 $rrdtool graph
/web/sunucusunun/sundugu/dizin/sshwww.png \
      DEF:s=sshwww.rrd:ssh:LAST \
      DEF:w=sshwww.rrd:www:LAST \
      AREA:s#FF0000:s \
      STACK:w#00FF00:w \
      --title="Port kullanimi" > /dev/null
```

Betiğimiz artık hazır. Çalıştırılabilir hale getirelim:

```
chmod +x BetigimizinAdi.sh
```

Cron

Bitti mi? Tabi ki hayır. veritabanını yaratırken boşuna 600 saniye demedik. Bu betiğin 600 saniyede bir (10 dakikada bir) çalıştırılması gerekli. Bunun için şu komutu veriyoruz:

```
crontab -e
```

Cron sistemimizde belli bir düzende komut çalıştırmak için bulunan bir araç. Karşımıza düzenlememiz için açılan dosyanın formatı çok karışık değil. 5 süre belirteci ve çalıştırılacak komut. Bir satırı "1 2 3 4 5 komut" şeklinde numaralarsak:

1: saatin şu dakikası, 2: günün şu saati, 3: ayın şu günü, 4: yılın şu ayı, 5: haftanın şu günü anlamına gelir. "*" koyarsak her an o değer doğru sayılır, */n koyarsak n birimde bir doğru olur. Örnek:

* * * * * komut: komut dakikada bir çalıştırılır

5 * * * * komut: her 5 geçe (saatte bir)

5 1 * * * komut: saat 1:05 (günde bir)

15 */5 * * * komut: 5 saatte bir çeyrek geçeleri

5,10,12,20 * * * * komut: dakika her 5, 10,12 veya 20 olduğunda

*/5 * * * * komut: 5 dakikada bir

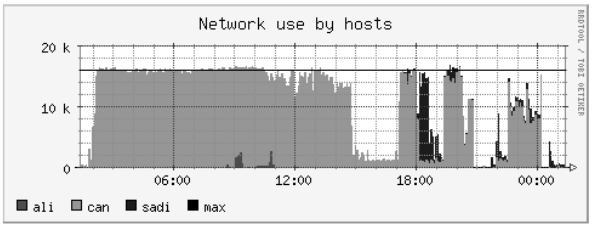
Ekleyeceğimiz satır şu şekilde olacak:

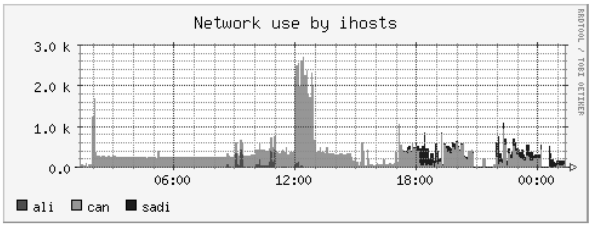
```
 */5 * * * *
/betigimizin/bulundugu/klasor/BetigimizinAdi.sh
```

Son olarak bu grafiği gösterecek bir html dosyası yaratalım:

```
<html>
 <head><title>IP Muhasebesi</title></head>
 <body><img src="sshwww.png" /></body>
</html>
```

Benden bahsettiklerimizden biraz daha kapsamlı 2 adet örnek grafik: (2 png dosyası)





Kaynaklar

RRDTool

http://people.ee.ethz.ch/~oetiker/webtools/rrdtool/
http://people.ee.ethz.ch/~oetiker/webtools/rrdtool/tutorial/

netfilter

http://tr.netfilter.org/documentation/HOWTO/packet- filtering-HOWTO.html
http://www.shorewall.net/NetfilterOverview.html

patch-o-matic

http://tr.netfilter.org/patch-o- matic/pom-pending.html
http://tr.netfilter.org/patch-o- matic/pom-base.html
http://tr.netfilter.org/patch-o- matic/pom-extra.html

NAT

http://tr.netfilter.org/documentation/HOWTO/NAT- HOWTO.html

Ağ Temelleri

http://tr.netfilter.org/documentation/HOWTO/networking- concepts-HOWTO.html