Wolfgang Hurst

Guidebook – FreeBSD & ISP Firewall

Ein FreeBSD Firewall zur Entkopplung eines Provider Netzwerkes

Wolfgang Hurst
Version 1.0

Guidebook – FreeBSD & ISP Firewall – Version 1.0

Dieses Dokument ist eine vollständige Neufassung. Es basiert auf unzähligen Webdokumenten, die ich im Laufe der Zeit erstellt habe. Durch ständiges Wechseln der CMS Systeme und Website Designs habe ich mich dazu entschlossen, all meine Dokumentationen von HTML auf PDF zu ziehen. Das lästige Rad-Neu-Erfinden bei jeder Webseiten Umstellung soll mir dadurch erspart bleiben.

Weiterhin will ich mich bei Hans Reitz bedanken. Er hat das Dokument noch einmal gelesen und meinen ganzen Text in eine für Menschen lesbare Form gebracht. Lochkarten sind ja nun auch schon länger nicht mehr aktuell …

Jetzt noch viel Spaß beim Lesen…

Wolfgang Hurst

http://www.whurst.net

Verzeichnisse

Inhaltsverzeichnis

Einleitung

Dieses Dokument beschreibt die Installation einer FreeBSD Maschine die das eigene Netzwerk vom Provider Netzwerk trennt. Das Dokument oder diese Vorgehensweise ist für alle Interessant, die in ihrem Netzwerk mehr als nur eine Maschine haben, seien es Drucker, Telefone, Faxgeräte, Fernseher, Laptops oder anderes. Durch die Installation eines Gateway-Servers zwischen dem Provider und uns können wir letztendlich frei entscheiden und alles machen, was uns einfällt. Die Funktionen des gelieferten Routers vom Provider können uns also vollkommen egal sein.

Die folgenden Seiten beschreiben den Aufbau eines solchen Gateways. Ich selbst habe das Gateway hier am Laufen und bin seitdem super-glücklich. Und als ich den Provider gewechselt hatte, brauchte ich nur den Eintrag für das Default-Gateway auf meinem Gateway-Server ändern. Das war's. Der Rest hier im Netzwerk zeigte sich von der Umstellung vollkommen unbeeindruckt. Mittlerweile habe ich einige Geräte hier, die nach einer IP verlangen. Da alles durch mein Gateway versorgt wird, haben diese Geräte den Wechsel noch nicht einmal mitbekommen. Und genau so sollte es sein.

Verwendete Variablennamen und Platzhalter

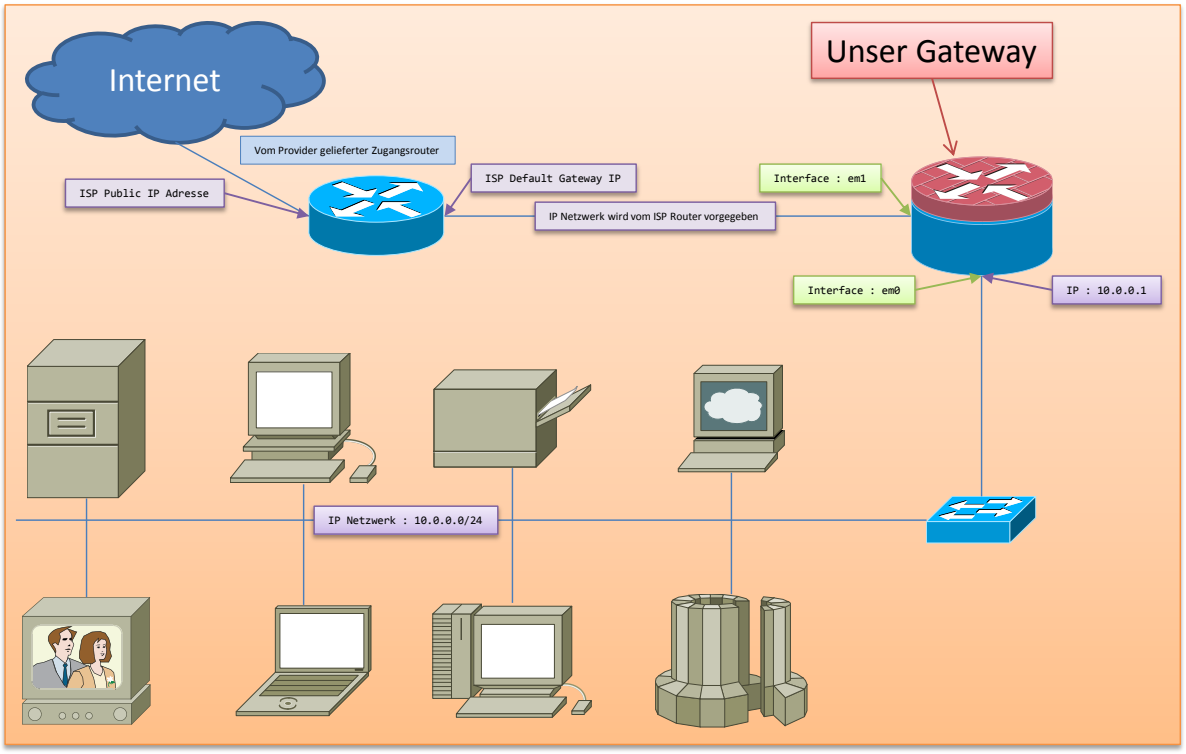
In dem Dokument habe Platzhalter für Hostnamen und dergleichen definiert. Die sind vom Installateur entsprechend mit den richtigen Werten zu füllen. Verwendete Variablenamen habe ich so gekennzeichnet: `IMMER_GROSS_BUCHSTABEN`. Und hier die Liste:

MYDOMAIN	**Das steht für den Domainnamen, den ihr intern im Netzwerk vergeben wollt. Dieser sollte sich nicht mit einer externen, anderen oder offiziellen Domain überschneiden.**
GWNAME	Der Hostname eures Gateways ohne die Domain. Der Hostname ist bei DNS und DDNS sehr wichtig.
MYWORKGROUP	Der Name eurer Arbeitsgruppe im Windows Netzwerk. Wenn ihr den Masterbrowser einrichten wollt, braucht ihr den

Netzwerkplan

Hier erst einmal der Netzwerkplan. Ich habe mich entschieden, das IP Netzwerk 10.0.0.0/24 zu benutzen. Nur das 10'er Netzwerk ist groß genug, um all meine Wünsche zu erfüllen ☺. IPv6 ist in Vorbereitung und kommt wohl in der nächsten Dokument-Version

Abbildung 1: Übersicht des gesamten Netzwerkes



Anforderungen und Wunschliste

Als erstes müssen wir die Anforderungen kennen und definieren: Was wollen wir, was wollen oder brauchen wir nicht. Das mag jeder frei selbst entscheiden. Folgende Dienste benötige ich und will ich haben:

- **Paketfilterung**. Sowohl eingehend (was ja sowieso nicht geht) als auch ausgehend. Ich habe ein paar Geräte/Programme hier, die möchten wohl gerne „nach-hause-telefonieren“. Das möchte ich aber irgendwie nicht und will es später abschalten können
- **Outgoing-NAT**. Das muss das Gateway sowieso können, weil ja sonst niemand mehr ins Internet kommen würde. Der Provider-Router weiß nämlich nichts von meinem 10’er Netzwerk
- **Incoming NAT**. Hin und wieder möchte ich aber auch, dass das große Internet direkt auf meine Dienste zugreifen kann. Ich betreibe zum Beispiel hin und wieder mal einen Gameserver. Also muss das Paket durchgereicht werden
- **NTP**. Mein Gateway soll als NTP-Server dienen. Es soll sich dazu die Zeit aus dem Internet holen und es im internen Netzwerk zur Verfügung stellen
- **DHCP**. Da ich es leid bin, mobilen und sonstigen Geräten mit der Fernbedienung oder durch DIP Schalter eine IP zu vergeben, will ich einen DHCP Server. Dann brauche ich das Ding nur anzuschließen und kann arbeiten
- **DNS / DDNS**. Mein Gateway soll ein Rekursiven-Root DNS-Server zur Verfügung stellen, damit jeder (im internen Netzwerk) ausschließlich nur mein Gateway für DNS Anfragen benutzen kann und soll. Und das eigene Netzwerk vom DHCP sollte er auch zur Verfügung stellen
- **WINS**. Ich hab ein paar Geräte hier, die sich in einer Windows Domain sehr wohl fühlen. Da ich keine permanenten Windows Server am Laufen habe, brauche ich zumindest einen festen WINS Server, an den sich alle klammern können
- **PROXY**. Ein http/https Proxy ist angedacht, aber noch nicht umgesetzt. Den kann man wunderbar zum Filtern von komischen Webseiten nehmen. Die ganzen Ad-Server und ga.js und Zählpixel Kram könnte man so global für alle abschalten. Bin aber noch nicht dazu gekommen
- **VPN**. Ich will gerne einen Server im Internet transparent in mein 10’er Netzwerk mappen. Es sieht dann so aus, als ob der Server lokal angeschlossen wäre, in Wirklichkeit steht er aber irgendwo anders. Das ist auch noch nicht implementiert, kommt eventuell in der nächsten Dokument Version

Installation und Basis Konfiguration des Gateways

Zur Installation von einem FreeBSD möchte ich nicht viel sagen. Der Vorgang sollte vertraut sein. Man nimmt die CD und installiert es. Punkt. Ich hab noch ein 8.2 verwendet. Bei der Installation sollte man jedoch **/var** auf eine eigene Slice ziehen - falls man mal was Debuggen will und überall die Protokoll Switche einschaltet, könnte es nach einer gewissen Zeit dort sehr eng werden.

Ansonsten entspricht die Installation dem Standard. Keine Maus, kein DHCP, kein NFS, kein NIS, kein nix. Außer SSH natürlich. Beide Interfaces sollte man schon einmal konfigurieren. Wir müssen unbedingt mit der Maschine ins Internet - das muss laufen, weil wir Pakete brauchen. Alles andere ziehen wir im Verlauf des Dokumentes glatt.

Nach der Installation bearbeiten wir noch den Syslog Daemon. Ich habe dem Syslog Daemon verboten, am Netzwerk zu lauschen. Damit nehme ich mir zwar die Möglichkeit, einen globalen Log Daemon zu implementieren, aber ich will das auch nicht. Dazu gehe ich dann mal kurz in die **/etc/rc.conf** und füge folgende Zeile hinzu:

```
syslogd_flags="-v -v"
```

Beim nächsten Reboot oder Restart des Syslog Daemon verschwindet der Socket dann, und alles ist gut.

Ihr könnt aber auch frei alles abschalten, was ihr nicht braucht oder wollt. Im weiteren Verlauf des Dokumentes brauchen wir zwar diverse Dienste, ich beschreibe aber immer alle Schritte. Bedeutet: das System ist jetzt richtig „Blanco“.

Was man aber unbedingt noch in die **/etc/rc.conf** einfügen sollte, ist das hier:

```
tcp_extension=YES
icmp_drop_redirect=YES
icmp_log_redirect=YES
```

ICMP Redirects sollte man grundsätzlich wegwerfen und den Sender sofort untersuchen und ggf. abschalten. Sofern man so etwas dreckiges nicht selbst gebaut hat ☺

Konfiguration des Paketfilters

Zum Einsatz kommt IPFW. Dabei verfolge ich aber eine andere Vorgehensweise als alle anderen (wie mir scheint). (Zumindest laut Google). Letztendlich ist es nur wichtig, dass der Filter das tut, was wir wollen. Wie ihr es macht, ist eigentlich egal. Ich baue keine IPFW Rule Sets mit Variablen und Bau-Mir-Die-Regeln-Scripte. Das finde ich unübersichtlich. Aber jeder so wie er will.

Ich baue ein Rule Set, das beim Starten einmal aufgerufen wird. Alle Änderungen, die ich im laufenden Betrieb mache, schreibe ich in das Rule Set und mache dann ein Copy&Paste auf der Kommandozeile. Somit bin ich mir 10000% sicher, dass die Regel auch nach den Booten ohne Probleme durchläuft. Andere Restarten einfach den Filter - wenn man Pech hat, verliert man dabei aber Verbindungen (das mögen meine Maschinen überhaupt nicht).

Als erstes sorge ich dafür, dass mein Rule Set beim Starten auch ausgeführt wird. Dazu schreibe ich folgende Zeilen in die **/etc/rc.conf**:

```
firewall_enable=YES
firewall_type=/etc/fwruleset
firewall_logging=YES
```

Danach erstelle ich die **/etc/fwruleset** Datei mit dem Grundgerüst, das ich brauche:

```
# allow ALL IP on loopback device
add 30101 allow ip from any to any via lo0

# allow ALL IP from internal network
add 30201 allow ip from any to any via em0

# allow ALL from myself to internet
add 30301 allow tcp from me to any via em1 keep-state
add 30302 allow udp from me to any via em1 keep-state
add 30303 allow icmp from me to any via em1 keep-state

# DROP ANYTHING else
add 30999 deny log all from any to any
```

Diese Regeln sind mit Sicherheit verbesserungswürdig. Ich lasse erst einmal alles Interne hemmungslos auf mein Gateway zugreifen. Und dem Gateway selbst gebe ich alle Rechte, um ins Internet zu gehen. Wer mag, darf das gerne für seine Zwecke erweitern.

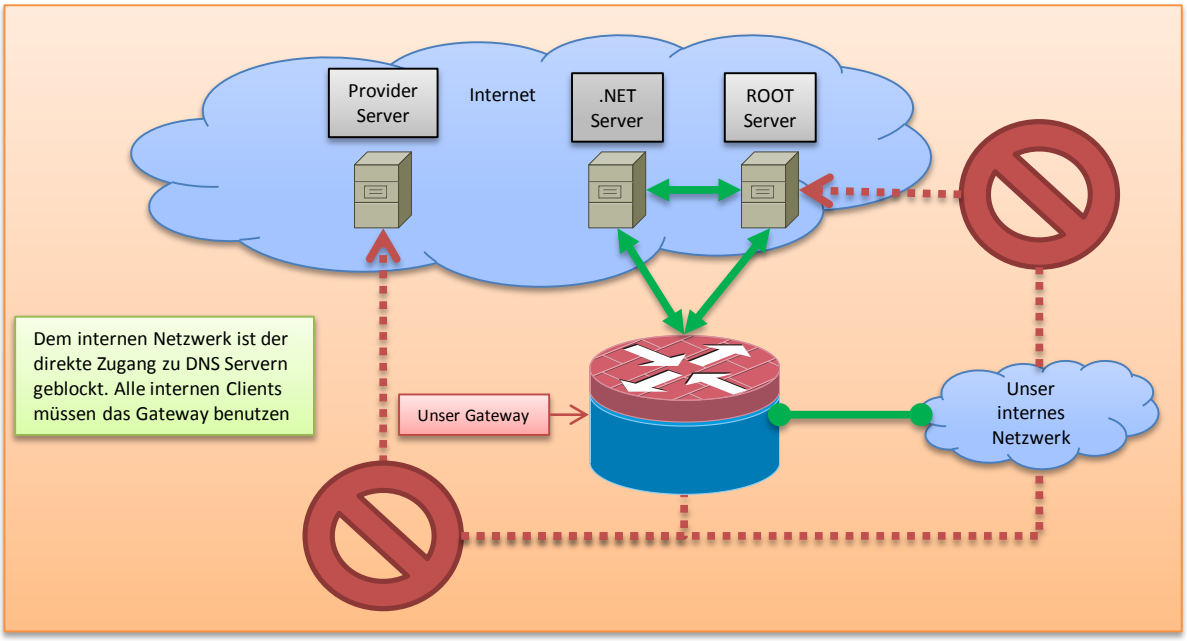
Am besten macht man nun ein Reboot. Statdessen einfach nur die Firewall zu laden, ist bei mir fehlgeschlagen, weil er das Kernelmodul nicht geladen hatte. Ich mach bei so etwas grundsätzlich immer ein Reboot. Will ja nicht beim nächsten Stromausfall oder Netzteildefekt lange rummachen und mich wundern, warum es nicht mehr geht. Je kleiner die Änderungen sind, desto weniger muss man debuggen ☺

Nach dem Reboot sollte die Maschine, wie auch vorher, ins Internet können und schaut mit **ipfw show** nach ob die Regeln alle da sind. Logisch gesehen verhält sich dieser Paketfilter genauso wie kein Paketfilter. Man merkt keinen Unterschied … noch nicht …

Konfiguration von DNS

DNS ist das erste, was wir machen müssen. Damit die Namensauflösung im Netzwerk funktioniert. Und zwar sowohl für das Gateway als auch für unser internes Netzwerk. Der Nameserver wird aber nicht als Forwarder konfiguriert, weil: wir bekommen ja den Provider DNS entweder auf einem Zettel oder per DHCP vom Provider Router zugewiesen. Den wollen wir ja nicht benutzen. Sondern wir wollen die ROOT Server. Die folgende Grafik zeigt die Beziehungen zueinander:

Abbildung 2: Übersicht der DNS Beziehungen



Als erstes aktivieren wir den DNS in der `/etc/rc.conf` mit folgender Zeile:

```
named_enable=YES
```

Danach editieren wir die `/etc/resolv.conf` und tragen dort unseren DNS ein, also nicht den DNS Server den wir vom Provider bekommen haben:

```
search MYDOMAIN
nameserver 127.0.0.1
```

Jetzt editieren wir die von FreeBSD schon fast fertige `/etc/namedb/named.conf` und stellen sicher dass folgende Zeilen drin sind:

```
options {
        …
        listen-on { 10.0.0.1; 127.0.0.1; };
        …
};

zone "." { type hint; file "/etc/namedb/named.root"; };
```

Das **listen-on** sorgt dafür, dass der DNS Server nur auf den localhost und auf der Adresse zum internen Netzwerk hin lauscht. Die **zone** Anweisung ist schon drin, sagt aber aus, dass der DNS Server sich bitte an die ROOT Server wenden soll. Den Rest kann man so lassen. Danach starten wir den DNS Server mit:

```
/etc/rc.d/named start
```

und testen ihn:

```
nslookup www.google.de
Server:         127.0.0.1
Address:        127.0.0.1#53

Non-authoritative answer:
www.google.de   canonical name = www-cctld.l.google.com.
Name:   www-cctld.l.google.com
Address: 173.194.70.94
```

Wichtig dabei ist, dass er die **127.0.0.1** benutzt und irgendeine Adresse rauswirft.

Konfiguration von NTP

Der NTP Server ist ähnlich einfach wie der DNS Server. Deshalb spare ich mir an dieser Stelle eine aufwendige Grafik. Als erstes aktivieren wir den NTP und den NTPDATE in der **/etc/rc.conf** mit folgenden Zeilen:

```
ntpdate_enable=YES
ntpd_enable=YES
```

Die Default NTP Konfiguration fragt die FreeBSD NTP Server. Das kann man ändern in der **/etc/ntp.conf** - ich hab es so gelassen.

Danach startet man einmal den NTP-Date um die aktuelle Zeit zu holen mittels

```
/etc/rc.d/ntpdate start
```

und danach startet man den Daemon dazu:

```
/etc/rc.d/ntpd start
```

Da ich aber eine seltsame Maschine habe, kann es sein, dass mir „die Zeit davonrennt“, falls der NTPD mal keine Verbindung hat. Der NTPD erkennt das leider nicht und kann die Uhrzeit nicht wieder neu setzen, wenn die Differenz zu groß ist. Aus diesem Grunde restarte ich den NTP jede Nacht um 02:00 Uhr neu. Dazu startet mal als ROOT das

```
crontab -e
```

und trägt folgende Zeile (es ist nur EINE ZEILE!) ein:

```
0 2 * * * /etc/rc.d/ntpd stop; sleep 5; /etc/rc.d/ntpdate start;
sleep 2; /etc/rc.d/ntpd start
```

Das war’s auch schon.

Konfiguration von DHCP

Als nächstes kommt der DHCP Server. Wenn ihr keinen haben wollt, könnt ihr das Kapitel überspringen. Ich will aber einen haben, weil ich faul bin und öfter mal eine Installation zum Testen machen will - da ist DHCP richtig gut. Auch hier erspare ich mir eine Grafik, weil das Zeichnen der Funktionsweise eines DHCP Servers dem Aufwand nicht gerecht wird.

Wir brauchen dazu aber ein Paket. Da DNS und Internet rennt, können wir es uns ja holen:

```
pkg_add -r isc-dhcp31-server
```

Danach aktivieren wir es in der **/etc/rc.conf** schon einmal mit folgenden Zeilen:

```
dhcpd_enable=YES
dhcpd_ifaces=em0
```

Die DHCP Konfiguration unter **/usr/local/etc/dhcpd.conf** erstellen wir neu:

```
option domain-name "MYDOMAIN";
option domain-name-servers 10.0.0.1;

authoritative;
default-lease-time 604800;
max-lease-time 2592000;
ddns-update-style none;
log-facility local7;

subnet 10.0.0.0 netmask 255.255.255.0 {
  range 10.0.0.130 10.0.0.199;
  option subnet-mask 255.255.255.0;
  option broadcast-address 10.0.0.255;
  option routers 10.0.0.1;
}

host SPECIAL-HOSTNAME {
        hardware ethernet 01:02:03:04:05:06;
        fixed-address 10.0.0.13;
}
```

Ich habe eine hohe Lease Zeit angegeben, damit alle anderen erst einmal weiter arbeiten können, falls mal mein Gateway ausgefallen ist. Weiterhin habe ich nur den Bereich 10.0.0.130 bis 10.0.0.199 als Pool angegeben. Man kann die ganzen DHCP Clients dann mit 10.0.0.128/25 in der Firewall reglementieren, wenn man möchte. Weiterhin habe ich einen statischen Host mit einer festen IP.

Jetzt startet man den DHCP Server mittels:

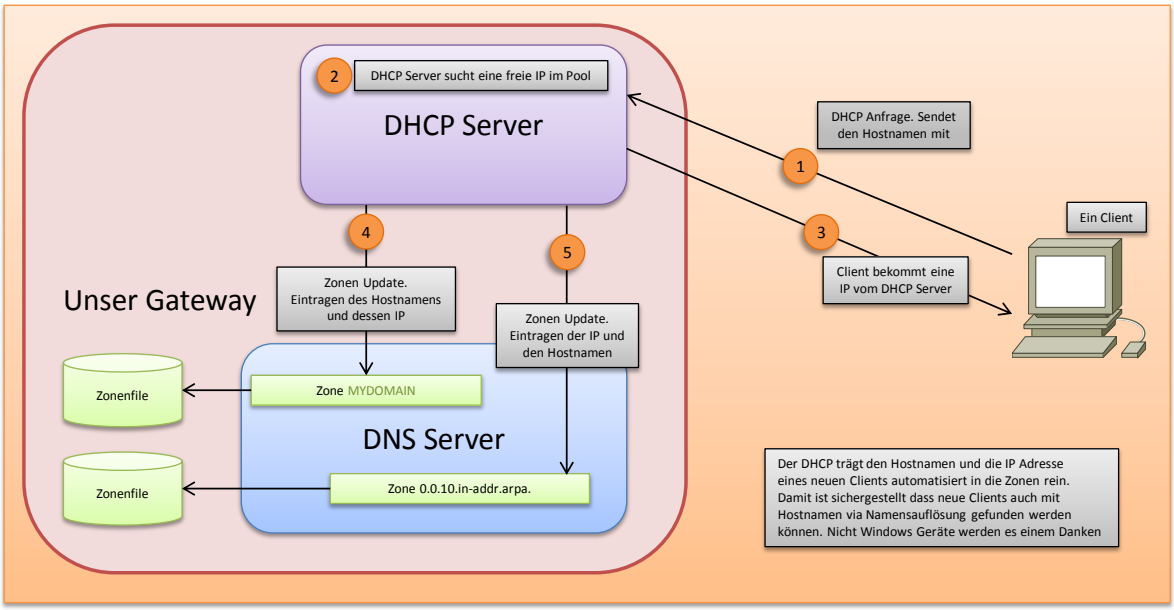
```
/usr/local/etc/rc.d/isc-dhcpd start
```

und testet ihn. Loginformationen liegen in der **/var/log/messages**.

Konfiguration Dynamic DNS mit DHCP

Solltet Ihr keinen DHCP betreiben, dann braucht Ihr das nicht. Das Problem bei DHCP ist ja, dass die Clients immer mal gerne andere IP Adressen bekommt. Wenn man nun einen Test-Server probieren will, ist das halt umständlich, weil man erst schauen muss, welche IP Adresse er hat. Um das Problem zu umgehen, gibt es DDNS. Dabei trägt der DHCP Server bei einer erfolgreichen Anfrage den Hostnamen und die IP in eine Zone des DNS Servers ein. Somit sind die Server oder Clients immer mit dem Hostnamen auffindbar. Die folgende Grafik zeigt die Funktionsweise:

Abbildung 3: Funktionsweise von DHCP und DDNS



Dazu muss der DHCP Server aber erst einmal dafür umkonfiguriert werden. Man muss folgende Zeile in der `/usr/local/etc/dhcpd.conf` anpassen:

```
ddns-update-style interim;
```

Durch den domain-name und domain-name-server findet er den DNS Server. Und dieser muss nun stark erweitert werden. Als erstes passen wir die `/etc/namedb/named.conf` an. Mit:

```
acl dhcpserver {
        10.0.0.1;
        127.0.0.1;
};
```

geben wir dem DNS Server einen Symbolischen ACL Namen. Danach definieren wir unsere Master Zone:

```
zone "MYDOMAIN" {
        type master;
        file "/etc/namedb/dhcp/MYDOMAIN";
        allow-update { dhcpserver; };
};
```

Und eine dazugehörige Reverse Zone:

```
zone "0.0.10.in-addr.arpa" {
        type master;
        file "/etc/namedb/dhcp/0.0.10.in-addr.arpa";
        allow-update { dhcpserver; };
};
```

Danach erstellen wir ein Verzeichnis **/etc/named/dhcp** und erstellen die beiden Zonen-Dateien. Hier die **/etc/namedb/dhcp/MYDOMAIN**:

```
$ORIGIN .
$TTL 3600
MYDOMAIN IN SOA GWNAME.MYDOMAIN. root.GWNAME.MYDOMAIN. (
                                3471230859 ; serial
                                600        ; refresh
                                600        ; retry
                                36000000   ; expire
                                3600       ; minimum
                                )
$TTL 600
                        NS      GWNAME.MYDOMAIN.
$ORIGIN MYDOMAIN.
```

Und wir bauen noch die Reverse Zone in der **/etc/namedb/dhcp/0.0.10.in-addr.arpa**:

```
$ORIGIN .
$TTL 3600
0.0.10.in-addr.arpa IN SOA GWNAME.MYDOMAIN. root.GWNAME.MYDOMAIN. (
                                3471230854 ; serial
                                600        ; refresh
                                600        ; retry
                                36000000   ; expire
                                3600       ; minimum
                                )
$TTL 600
                        NS      GWNAME.MYDOMAIN.
$ORIGIN 0.0.10.in-addr.arpa.
```

Damit der Nameserver diese Dateien jetzt auch noch bearbeiten kann, „schenken“ wir ihm das ganze DHCP Verzeichnis:

```
chown -R bind:bind /etc/namedb/dhcp
```

Danach stoppen wir den DNS und DHCP Server:

```
/etc/rc.d/named stop
/usr/local/etc/rc.d/isc-dhcpd stop
```

Nun starten wir den DNS Server und schauen auf die Logfiles. Die sagen einem ganz genau, ob es klappt oder nicht. Danach starten wir den DHCP Server. Jetzt testet es aus: Hängt einen DHCP Client an die Leitung, und dieser sollte dann in den Zonenfiles erscheinen. Er kann dann mittels Namen gefunden werden. Auch denn wenn er eine andere IP bekommt.

Konfiguration von WINS

Da ich viele SMB-artige Geräte habe, die ich mit SMB anspreche, brauche ich im Netzwerk einen unbestrittenen Master Browser, der sich als Domain Controller tarnt. Damit fällt dann auch diese „Rum-Broadcast-erei“ weg, und ich habe alle Gerät schön brav in meinem Netzwerk Explorer. Wer das nicht will, kann dieses Kapitel überspringen. Ich brauche dafür aber Samba:

```
pkg_add -r samba35
```

Danach aktivieren wir ihn schon einmal in der **/etc/rc.conf**:

```
samba_enable=YES
```

Nun kann man eine vorhandene **smb.conf** nehmen, oder sie von Grund auf neu schreiben. Hier meine **/usr/local/etc/smb.conf**:

```
[global]
   workgroup = MYWORKGROUP
   netbios name = GWNAME
   server string = MYWORKGROUP Master Server
   security = user
   hosts allow = 10.0.0.
   load printers = no
   log file = /var/log/samba/log.%m
   max log size = 50
   bind interfaces only = yes
   interfaces = em0
   local master = yes
   os level = 33
   domain master = yes
   preferred master = yes
   domain logons = no
   wins support = yes
   dns proxy = no
```

Jetzt starten wir den Samba Server mit:

```
/usr/local/etc/rc.d/samba start
```

Damit alle DHCP Clients das auch mit bekommen, benötigen wir noch einen Eintrag in unserer DHCP Konfiguration **/usr/local/etc/dhcpd.conf**:

```
option netbios-name-servers 10.0.0.1;
```

und danach den DHCP Server neu starten mit:

```
/usr/local/etc/rc.d/isc-dhcpd restart
```

Das war’s. Jetzt sollten die DHCP Clients auch alle den Master Browser kennen und sich dort melden.

Konfiguration Outgoing NAT

Nachdem jetzt unser internes Netzwerk einigermaßen funktionieren sollte, wollen wir dafür sorgen, dass die Clients auch den Weg nach draußen schaffen. Zurzeit können sie das nicht. Ich verwende für das Outgoing NAT noch die alte Verfahrensweise. Da gibt es aber leider einen Bug. FreeBSD startet den NATD Daemon erst nach dem Paketfilter, was dazu führt, dass unser Ruleset nicht angenommen wird. Um das zu verhindern laden wir das Kernel Modul in der **/boot/loader.conf** wie folgt:

```
ipfw_load="YES"
ipdivert_load="YES"
```

Danach aktivieren wir den NAT Daemon in der **/etc/rc.conf** wie folgt:

```
natd_enable=YES
natd_interface=em1
natd_flags="-dynamic -m"
gateway_enable=YES
```

Und jetzt machen wir noch ein paar Änderungen in unserer **/etc/fwruleset**. Mit:

```
# translate anything incoming/outgoing
add 30401 divert natd ip from any to any in via em1
```

sorgen wir dafür, dass alle Pakete die es schaffen sollten, am Internet Interface anzukommen, in den NAT Daemon umgeleitet werden. Dieser nimmt das Paket entgegen, und wenn es zu seinen Aufzeichnungen passen sollte, ändert er die IP Adressen um und sendet es weiter. Damit er aber weiß, was er erkennen soll und was weitergeleitet werden soll, ist folgendes nötig:

```
# allow nat services to outside world
add 30501 skipto 40001 icmp from 10.0.0.0/24 to any keep-state
add 30502 skipto 40001 tcp from 10.0.0.0/24 to any keep-state
add 30503 skipto 40001 udp from 10.0.0.0/24 to any keep-state
```

Damit geben wir ihm die Anweisung, dass alles was von 10.0.0.0/24 zu irgendwo hin geht er in die Regeln 40001 springen soll, und dort steht dann:

```
# translate the 10 ip into my own internet ip
add 40001 divert natd ip from any to any out via em1
add 40002 allow ip from any to any
```

Dort geben wir ihm mit, dass diese Pakete mit dem NAT Daemon umgewandelt werden sollen, und er merkt sich das dann auch. In Regel 30401 wird es ja wieder zurück verwandelt, wenn es zurückkommen sollte.

Jetzt muss rebootet werden, damit die Änderungen auch im Kernel Loader anspringen. Danach solltet ihr von allen 10.0.0.0/24'er IP Adressen aus surfen können.

Konfiguration Incoming NAT

Für das Incoming NAT muss man den Kernel neu bauen. Die Forward Option ist im Default Kernel nicht drin. Das Incoming NAT benötigt man aber nur, wenn man einen eigenen Dienst anbieten will, der aus dem Internet aus erreichbar sein soll. Zum Beispiel Remote Desktop oder Gameserver. Um den Kernel zu konfigurieren, befolgt man die normalen Kernel-Bau Anleitungen und fügt in der Kernel-Konfiguration folgende Optionen hinzu:

```
options LIBALIAS
options IPFIREWALL
options IPFIREWALL_NAT
options IPFIREWALL_FORWARD
options IPSTEALTH
options IPDIVERT
```

Danach Kernel bauen und installieren.

Beim Incoming NAT verwende ich nun die neue Art und Weise von IPFW, die mir am besten zusagt. Das folgende Beispiel leitet den Port 25600 (Serious Sam) weiter auf meine kleine schnucklige Spielekiste mit der IP 10.0.0.13:

```
ipfw add 30001 nat 301 udp from any to me dst-port 25600 in via em1
ipfw nat 301 config if em1 redirect_port udp 10.0.0.13:25600 25600
```

In der ersten Zeile sage ich IPFW, dass er bei ankommenden UDP Paketen auf Port 25600 auf dem Internet Interface das Paket mit der NAT Regel 301 bearbeiten soll.

In der zweiten Zeile konfiguriere ich die NAT Regel 301 und sage ihm, dass er das Paket an 10.0.0.13 weiterleiten soll, und zwar auch auf den Port 25600.

Aber Achtung ! Wenn die 10.0.0.13 ein DHCP Client sein sollte, hat man irgendwann ein Problem. Bei einem DHCP Client funktioniert es nur, wenn zum Zeitpunkt der Regeleingabe der Server auch diese IP besitzt. Ändert er seine IP, wirkt diese Regel nicht mehr.

Die Regeln selbst habe ich aber in einem Script und nicht in der `/etc/fwruleset` drin. Ich kann somit die Weiterleitung manuell aktivieren oder durch ein `ipfw delete 30001` wieder deaktivieren. Der ISP Router kann dabei permanent mit der NAT Regel gefüttert werden. Wenn meine Regel 30001 nicht aktiv ist, schlagen die Pakete letztendlich gegen die Deny-Any Regel.