(no title)

Ingo Blechschmidt

4. August 2005

Inhaltsverzeichnis

0.1 1337

0.1.1 Infothek-Aufbau

Yeah da Aufbau da Reload of da Infothek visit us at irc.holbein-gymnasium.de port 6667 #raum29 and always remember phear the Heinz!

Thankz to da Frieb for da Picturez

Brought to j00 by da team on 9.9.2004 and following dayz

0.1.2 John the Ripper

Mitschneiden

»**/mnt/warez/files/**«
: Enthält Unterordner mit den Namen der Streams. In diesen Ordnern sind dann die mitgeschnittenen Dateien.

»**/mnt/warez/logs/**«
: Die Log-Datei von streamripper, für jeden Stream. Escape-Sequenzen, die streamripper erzeugt, werden von log-cleaner automatisch weggeschnitten.

»**/mnt/warez/server-started**«
: Existiert diese Datei, so läuft streamripper.

  Wird benötigt von streamcd, um den streamd je nach Tageszeit zu starten oder zu stoppen.

»**/mnt/warez/streamd**«
: Startet alle streamripper-Prozesse, unabhängig von der Tageszeit.

»**/mnt/warez/streamcd**«
: Abhängig von der Tageszeit wird streamd aufgerufen, um streamripper zu starten oder zu stoppen. Wird automatisch alle fünf Minuten von cron aufgerufen.

»**/mnt/warez/log-cleaner**«
: Löscht die Escape-Sequenzen in den streamripper-Logs. Wird jede halbe Stunde von cron aufgerufen.

SSH-Tunnel

»/home/ripper/tunnelnx.sh« baut einen Tunnel von »m19s28.vlinux.de:1235« zu »john:22« auf, und kümmert sich auch darum, dass der Tunnel automatisch wieder gestartet wird, wenn er mal down geht, etc. Wird automatisch von rc gestartet (»/etc/rc2.d/S99tunnler«).

Zugriff geht dann mit: »ssh -p 1235 root@m19s28.vlinux.de«. Aber: Man wird ständig seine »~/.ssh/known_hosts« anpassen müssen, weil der Hostname der gleiche ist (m19), aber der sshd dahinter nicht. Abhilfe schafft folgender Eintrag in der »/etc/hosts«:

```
$ su -
Password:
# cat >> /etc/hosts
83.151.29.94    john.infothek   john
^D
# exit
$ ssh -p 1235 root@john
Password:
```

0.1.3 Mini-HowTo: Wie bekommt mein Netz IPv6

Softwarevoraussetzungen

- Auf dem Router testen, ob folgende Befehle funktionieren. Die genauen IPs sind (noch) nicht entscheidend. Wichtig ist nur, dass die Befehle überhaupt funktionieren. Wenn´s irgendwo ein Problem gibt: »ipv6«-Kernelmodul geladen? Ist IPv6 im Kernel aktiviert?

```
# ip tunnel add tunnelbroker mode sit remote 80.81.x.y
ttl 64
# ip link set tunnelbroker mtu 1472
# ip link set dev tunnelbroker up
# ip -6 addr add 2001:8e0:abcd::42/126 dev tunnelbroker
# ip -6 addr add 2001:8e0:abcd:14d::/64 dev eth0
# ip -6 route add 2000::/3 via 2001:08e0:abcd::535 dev
tunnelbroker
# ip -6 route add 3ffe::/16 via 2001:08e0:abcd::535
dev tunnelbroker
# echo "1" > /proc/sys/net/ipv6/conf/all/forwarding
```

- Auf den Clients:

```
# ping6 -I eth0 ff02::1
```

  Das sollte einfach nur alle Rechner des lokalen Netzes anpingen.

Tunnel von as8758.net

- Auf »http://tunnelbroker.as8758.net/«[1] gehen, registrieren und Tunnel holen.
- Um ein »/64«er-Subnetz bitten.
- Eigene IPv6-Adressen erhalten.

[1] `http://tunnelbroker.as8758.net/`

Konfiguration des Routers

- Hier ein Perl 5-Programm, welches auf »http://tunnelbroker.as8758.net/« geht und die Tunnelparameter entsprechend der aktuellen öffentlichen IPv4 ändert. Der Mechanismus zur Bestimmung der aktuellen IPv4-Adresse muss evtl. geändert werden. Oder das Programm wird in »ip-up« eingebaut, da gibt´s die öffentliche IPv4 als Parameter, welcher dem Programm dann übergeben werden kann.

```
#!/usr/bin/perl

use warnings;
use strict;

INIT { $ENV{PATH} = "/sbin:/usr/sbin:$ENV{PATH}" }
use WWW::Mechanize;

use constant {
  USERNAME => "...",
  PASSWORD => "...",
  EXT_IF   => "ppp0",
};

my $ipv4_public =
  (split /\s+/, 'ip addr show dev @{[EXT_IF]} | grep
inet')[2];
$ipv4_public =~ s/\/.*//;

die "Couldn't get public IPv4 address!\n" unless $ipv4_public;
print STDERR "Public IPv4 address: $ipv4_public\n";

print STDERR "Logging in to as8758...\n";
my $mech = WWW::Mechanize->new;
$mech->get("http://tunnelbroker.as8758.net/login.php");
$mech->form_number(1);
$mech->field(username => USERNAME);
$mech->field(password => PASSWORD);
$mech->click("Login");

print STDERR "Changing tunnel parameters...\n";
$mech->form_number(1);
$mech->field(ipv4tunnelend => $ipv4_public);
$mech->click("ipv4change");
```

- Aufbauen des Tunnels nach Änderung des Tunnelkonfiguration durch das Programm des vorherigen Schritts:

```
#!/bin/sh
# Aufruf: skriptname "öffentliche_ipv4"

LOCAL4="$1"
```

```
REMOTE4=212.25.25.23            # Tunnelserver von
as8758.net
LOCAL6=2001:08e0:abcd::536/126  # Zugeteilter IPv6-Endpunkt,
wir
REMOTE6=2001:08e0:abcd::535     # Zugeteilter IPv6-Endpunkt,
as8758
NET6=2001:8e0:abcd:14d::/64     # Zugeteiltes /64er-Subnetz

# Tunnel hochbringen
ip tunnel add tunnelbroker mode sit remote ${REMOTE4}
ttl 64
ip link set tunnelbroker mtu 1472
ip link set dev tunnelbroker up
ip -6 addr add ${LOCAL6} dev tunnelbroker
ip -6 addr add ${NET6} dev eth0

# Standardrouten setzen
ip -6 route add 2000::/3 via ${REMOTE6} dev tunnelbroker
ip -6 route add 3ffe::/16 via ${REMOTE6} dev tunnelbroker

# Forwarding für andere Clients des Netzes aktivieren
echo "1" > /proc/sys/net/ipv6/conf/all/forwarding

# Firewall
ip6tables -F
ip6tables -A INPUT -i tunnelbroker \
    ! -s ${NET6} -p tcp --dport 6000:6010 -j DROP
```

- Konfiguration des Router Advertisement Daemons (»radvd«) auf dem Router:

```
interface eth0 {
    # Router Advertisements rausschicken
    AdvSendAdvert on;

    MinRtrAdvInterval 3;
    MaxRtrAdvInterval 10;

    # Kein Mobile-IPv6
    AdvHomeAgentFlag off;
    AdvLinkMTU 1280;

    # Definition für zugewiesenes Netz
    prefix 2001:08e0:abcd:14d::/64 {
        # Wir sind zuständig...
        AdvOnLink on;
        AdvAutonomous on;
        # ...und geben den Clients auch unsere IPv6
        AdvRouterAddr on;
    };
};
```

- Konfiguration auf den Clients: Keine notwendig, da IPv6 Stateless Autoconfiguration :)

Alternative zu as8758

Wenn man auf dem Router kein perl 5 installieren kann/will, und/oder sich nicht bei as8758 registrieren will/kann, muss man dennoch nicht auf IPv6 verzichten, die Antwort heißt 6to4[2].

Skript, welches von »ip-up« aufgerufen werden muss:

```
#!/bin/sh

LOCAL4="$1"
LOCAL6="‘printf "2002:%02x%02x:%02x%02x::1" \
    $(echo "$LOCAL4" | tr "." " ")‘"

ip tunnel add tun6to4 mode sit ttl 64 \
    remote any \
    local "$LOCAL4"
ip link set dev tun6to4 up
ip -6 addr add "$LOCAL6"/16 dev tun6to4
ip -6 route add 2000::/3 via ::192.88.99.1 dev tun6to4 metric
1
```

Die radvd-Konfiguration muss leicht angepasst werden:

```
interface eth0 {
    AdvSendAdvert on;

    MinRtrAdvInterval 3;
    MaxRtrAdvInterval 10;

    # Kein Mobile-IPv6
    AdvHomeAgentFlag off;
    AdvLinkMTU 1280;

    prefix 0:0:0:42::/64 {
        AdvOnLink off;
        AdvAutonomous on;
        AdvRouterAddr on;
        Base6to4Interface ppp0;
        AdvPreferredLifetime 20;
        AdvValidLifetime 30;
    };
};
```

Wichtig ist noch, dass der »radvd« bei jeder Neueinwahl ein »SIGHUP« gesendet bekommt.

[2] `http://www.tldp.org/HOWTO/Linux+IPv6-HOWTO/configuring-ipv6to4-tunnels.html`