Fake-Hotspot im Eigenbau

Was funkt denn da?

POVaddict

14. September 2014

Wozu das Ganze?

- Wenige statische Webseiten
- Offensichtlich als Fake erkennbar
- Soll zeigen, dass nicht jedes offene WLAN mit Webseiten tatsächlich Internetzugang bedeutet
- Spass bei der Zugfahrt :-)

Zutaten (Hardware)

- USB-Stick
  mindestens 256 MB
- USB WLAN-Karte
  z.B. TP-Link TL-WN722N, zugehöriger Linux-Treiber muss AP-Modus (Access Point) unterstützen
- x86-basierter Rechner

Zutaten (Software)

- Linux-Kernel $>=$ 3.2
- `hostapd`
  Host-AP daemon, betreibt WLAN-Karte im AP-Modus
- `udhcpd`
  minimaler DHCP-Server
- `netwox`
  Netzwerk-Toolsammlung, u.a. sehr dummer DNS-Server
- `thttpd`
  minimaler Webserver, leider nicht mehr paketiert

Weitere Zutaten (Software, optional)

- `stunnel`
  https mit *sehr* kaputtem Zertifikat
- `charybdis`
  IRC-Server – wir wollen chatten
- `irssi` & `shellinabox`
  Webchat!
- `less mc openssh-client screen tcpdump vim w3m wireless-tools`
  Kann man immer gebrauchen

Wie basteln wir das zusammen?

Debian Live-Build als Basis

- Erstellt ein Live-Image mit individueller Paketauswahl
- Kann zusätzliche Dateien einbinden
- Technische Details nach der Live Demo. . .

Live Demo

Reboot please. . .

ESSID: `GehtNet`

Debian Live-Build

- Baut aus Standardpaketen ein Live-System
  Keine Anpassung an den Paketen nötig
- Installation: `apt-get install live-build`
- Zwei Phasen:
  - `lb config`: Festlegen von Repo-Server, Paketliste, zusätzliche Dateien einbinden
  - `lb build`: Bauen des Live-Systems in einer chroot-Umgebung (benötigt root-Rechte)
- Ergebnis ist ein Hybrid-ISO-Image
  Läuft von CD und USB-Stick

lb config – Optionen (gekürzt)

```
$ mkdir fake-ap
$ cd fake-ap
$ MIRROR=http://ftp.de.debian.org/debian
$ lb config --architectures i386 \
  --linux-flavours 486 \
  --mirror-bootstrap $MIRROR \
  --mirror-chroot $MIRROR \
  --mirror-binary $MIRROR \
  --bootappend-live "boot=live config
    locales=en_US.UTF-8
    keyboard-layouts=de"
```

lb config – Paketauswahl

```
$ echo "user-setup" > \
  config/package-lists/recommends.list.chroot
$ echo "charybdis console-setup hostapd irssi
  netwox shellinabox stunnel udhcpd
  less mc openssh-client screen tcpdump
  vim w3m wireless-tools" > \
  config/package-lists/my.list.chroot
```

lb config – Zusätzliche Dateien (1/2)

```
$ tar xzf ../sysconf.tgz \
  -C ./config/includes.chroot/
```

- `etc/charybdis` — IRC-Konfiguration
- `etc/default/shellinabox` — Kein Autostart
- `etc/skel` — Default-Dateien für den Live-User
- `etc/hostapd.conf` — Konfiguration für unverschlüsselten WLAN-AP
- `etc/shadow` — Passwort-Hash für root
- `etc/stunnel.pem` — Zertifikat für https

lb config – Zusätzliche Dateien (2/2)

- `etc/udhcpd.ap0.conf` — DHCP-Konfiguration
- `lib/firmware` — WLAN-Firmware, Auswahl
- `usr/local/bin/runirc` — Wrapperskript für irssi
- `usr/local/bin/thttpd` — Webserver-Binary
- `usr/local/bin/wifi-fakehotspot` — Fake-Hotspot Hauptskript
- `var/fakeweb` — Fake-Webseitenangebot

lb build – Starten und Teetrinken

```
# lb build
```

- Alles läuft in einer chroot-Umgebung ab
- Platzbedarf ca. 1.5 GB
- Dauert ca. 20 min auf ThinkPad X230 an 3 MBit/s DSL
- Größe des Hybrid-ISO-Image: 160 MB

Konfigurationsdatei `/etc/hostapd.conf`

```
# Interface ap0 benutzen
interface=ap0
# WLAN-Treiber ueber netlink benutzen
driver=nl80211
ssid=GehtNet
channel=9
hw_mode=g
```

Konfigurationsdatei `/etc/udhcpd.ap0.conf`

```
start 172.16.20.10
end 172.16.20.100
interface ap0
opt subnet 255.255.255.0
opt broadcast 172.16.20.255
# Bitte alles ueber uns routen
opt router 172.16.20.1
# DNS-Anfragen bitte an uns stellen
opt dns 172.16.20.1
```

Das Skript `wifi-fakehotspot` (1/4)

```
#!/bin/bash
# AP-faehiges WLAN-Interface in ap0 umbenennen
(
cd /sys/class/net
for i in wlan*
do
    if grep -Eq "(ath[59]|rt73usb|zd1211rw)" \
       "$i/device/uevent" ; then
        ip link set "$i" name ap0
        break
    fi
done
)
```

Das Skript `wifi-fakehotspot` (2/4)

```
# MAC-Adresse zufaellig setzen
uuid="$(cat /proc/sys/kernel/random/uuid)"
mac="00:23"
for i in 1 2 3 4
do
    mac="$mac:${uuid:0:2}"
    uuid="${uuid#??}"
done
ip link set ap0 address $mac
```

Das Skript `wifi-fakehotspot` (3/4)

```
# Unsere IP-Adresse setzen
ip addr flush ap0
ip addr add 172.16.20.1/24 brd + dev ap0
# Traffic zu fremdem Zielnetz zu uns umleiten
IPT=iptables
$IPT -t nat -A PREROUTING -d 255.255.255.255 \
  -j ACCEPT
$IPT -t nat -A PREROUTING ! -d 172.16.20.0/24 \
  -j DNAT --to 172.16.20.1
```

Das Skript `wifi-fakehotspot` (4/4)

```
# Dienste starten
hostapd -B /etc/hostapd.conf
udhcpd /etc/udhcpd.ap0.conf
thttpd -d /var/fakeweb -r
stunnel -d https -r localhost:http -s nobody \
  -p /etc/stunnel.pem -P ''
shellinaboxd -t -b \
  -s /:user:user:HOME:/usr/local/bin/runirc
# netwox bleibt im Vordergrund
netwox 104 -h router.catchme -H 172.16.20.1 \
  -a dns.catchme -A 172.16.20.1
```

Bin ich schon drin?

Einige Geräte prüfen, ob sie „wirklich“ im Netz sind:

Apple iOS:

`http://www.apple.com/library/test/success.html`

Microsoft Windows 7:

`http://www.msftncsi.com/ncsi.txt`

Amazon Kindle:

`http://spectrum.s3.amazonaws.com/kindle-wifi/wifistub.html`

Damit diese Geräte zufrieden sind, bieten wir das auch an :-)

Verlaufen unmöglich

Unsere 404-Seite leitet automatisch auf die Startseite um:

```
<html>
<head>
<meta http-equiv="refresh"
  content="0; URL=http://www.mirdochegal.net/">
<title>Du hast Dich verlaufen!</title>
</head>
<body>
<h3>Du hast Dich verlaufen!</h3>
</body>
</html>
```

Liste AP-fähiger WLAN-Treiber

Keine umfassende Liste, nur selbst getestete Treiber:

- `ath5k` (Atheros, PCI)
- `ath9k` (Atheros, PCI)
- `ath9k_htc` (Atheros, USB)
- `carl9170` (Atheros, USB)
- `rt2500usb` (Ralink, USB)
- `rt73usb` (Ralink, USB)
- `zd1211rw` (Zydas, USB)