



EIGENE FIRMWARE MIT FREETZ!

Free Fritz!

AVMs FRITZ!Boxen fristen oft ein Schattendasein als reiner DSL-Router, obwohl sich aus den kleinen Linux-Rechnern zumindest theoretisch mehr herausholen lässt. Die Scriptsammlung Freetz soll dabei helfen, selbst eine „gemoddete“ Firmware zu erstellen, die zusätzliche Funktionen enthält.

VON **MATTIAS SCHLENKER**

Ein Pinguin erobert die Welt: Fast unbemerkt vom Nutzer tut Linux in vielen DSL- Routern, Medienplayer, ja sogar Fernsehern und DVD-Playern seinen Dienst. Die „virale“ Natur der GPL erfordert die Freigabe der Quellcodes des Systemkerns und einiger weiterer Komponenten und schafft so eine Spielwiese für Entwickler und Bastler. AVMs FRITZ!Box ist solch ein typisches „Embedded System“: bis zu 64 MByte RAM, 16 MByte Flash und ein MIPS-Prozessor, dessen Rechenleistung etwa in den Bereichen von PCs und Workstation von vor zehn Jahren liegt.

Nun gehen die von AVM angebotenen Funktionen einigen FRITZ!Box-Nutzern nicht weit genug: Die Dateiserver-Funktionalität ist eher rudimentär, und die Rechteverwaltung reicht nicht an die einfacher NAS-Systeme heran. Neben der Druckerfreigabe wäre ein Zugriff auf Scanner wünschenswert (das unter Linux übliche SANE-Framework unterstützt Scannen übers Netz), und ganz oben auf der

Wunschliste stehen nächtliche BitTorrent-Downloads, ohne dass ein PC laufen muss, und die Möglichkeit, einen eigenen Webserver zu betreiben, der beispielsweise per DynDNS erreichbar ist.

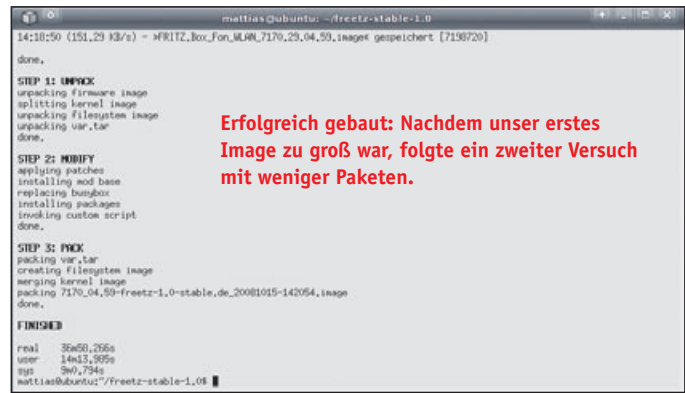
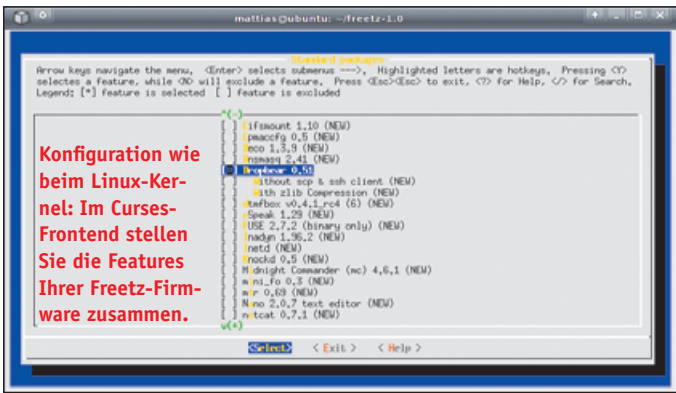
Was macht Freetz?

Die Freetz-Script-Sammlung baut kein komplett eigenes Firmware-Image, sondern modifiziert AVMs vorhandene Firmware. Der



Geflasht wird über das Webinterface der FRITZ!Box, was auch über WLAN funktioniert.

Grund dafür ist, dass die originale Firmware einige Komponenten enthält, die nicht der GPL unterliegen. Dazu gehören ein Teil des Webinterfaces, natürlich das AVM-Branding und einige essenzielle Programme beispielsweise zur Ansteuerung der DSL-DSPs. Freetz kompiliert eine modifizierte Version des Linux-Universalwerkzeuges *Busybox* und Treibermodule für den Linux-Kernel, die zusätzliche Dateisystemunterstützung bieten. Nach Abschluss des „Bauens“ wird ein originales Firmware-Image entpackt, und die unveränderten proprietären Komponenten des originalen Images werden zusammen mit den eigenen Erweiterungen zusammengefasst. Das so erstellte Image kann nun der FRITZ!Box als vermeintliches Update eingespielt werden. Ein gewaltiger Fortschritt gegenüber der Vorgängerversion *ds-mod* ist, dass bei der stabilen Version keine Cross-Compile-Toolchain, also der aus Compiler, Linker und einigen anderen Programmen bestehende Werkzeugersatz für die Übersetzung von Programmen



für fremde Architekturen, mehr erstellt werden muss. Dies eliminiert den einen oder anderen Fehler und senkt so die Einstiegschwelle. Gleichzeitig resultiert daraus ein gewaltiger Geschwindigkeitsgewinn: Auf einer moderat aktuellen Maschine ist ein modifiziertes Image in einer guten halben Stunde erzeugt, folgende Builds mit kleinen Konfigurationsänderungen laufen in wenigen Minuten durch.

Warum keine Firmware-Downloads?

Der Weg über das Freetz-Script mag unnötig umständlich erscheinen, er ist allerdings die einzige Möglichkeit, für die Freetz-Entwickler sich rechtlich abzusichern. Weil die modifizierten Firmware-Images die proprietären Komponenten von AVM enthalten, dürfen keine fertigen Images herausgegeben werden, da dies eine Urheberrechtsverletzung darstellen würde. Eine komplette freie Firmware, wie sie beispielsweise für die Router der WRT-Familie von Linksys (*OpenWRT*) oder viele NAS-Devices erhältlich ist, wird durch die rechtliche Situation unmöglich gemacht. Um den vollen

Funktionsumfang der originalen Firmware zu erreichen, müssten hierfür die proprietären Komponenten nachprogrammiert werden - eine extrem zeitaufwendige Aufgabe, die bei den Freetz-Entwicklern derzeit keine Priorität genießt.

Denn Sie wissen, was Sie tun!

Toolchains für den Bau von Linux-MIPS-Binärdateien fühlen sich auf „unixoiden“ Systemen am wohlsten. Konkret bedeutet das: Unter Windows ist das Bauen einer eigenen Freetz-Firmware praktisch unmöglich. Unter MacOS X ist zumindest einiges an Fingerspitzengefühl mit der beim Betriebssystem mitgelieferten Toolchain erforderlich, bis die Cross-Compile-Toolchain steht. Für Linux existieren dagegen fertige Toolchains mit allen notwendigen Modifikationen, um Binärdateien für die von der FRITZ!Box verwendeten MIPS-Architektur zu erstellen. Falls Sie primär unter Windows oder OS X arbeiten, sollten Sie daher Linux in einer Virtualisierungsumgebung wie *VMware* oder *VirtualBox* installieren. Für diesen Artikel richteten wir eine minimale

Ubuntu-8.04.1-Installation im VMware Player ein. Wer abkürzen möchte, findet auf den Seiten des Freetz-Projektes Links zu fertig vorbereiteten Images für den VMware-Player. Bei Verwendung des VMware-Players sollte dieser die Netzwerkeinstellung *Bridged* verwenden und Ihr PC zudem per Kabel mit der Box verbunden sein, damit ein direkter Upload der Firmware zur Box und im Ernstfall ein *Online-Restore* möglich ist.

Neben einer funktionsfähigen Linux-Installation sind Kenntnisse im Umgang mit Linux von Vorteil, die über reine Linux-Grundlagen hinausgehen. Gelegentlich kann es nämlich vorkommen, dass bei Einzelpaketen Fehler beim Bauen auftreten, weil Compiler und Quellcodes nicht zusammenpassen oder Pakete nicht mehr verfügbar sind. Gelegentlich können Änderungen an der Kernelkonfiguration notwendig werden. Bei den folgenden Anpassungen der Einstellungen sollten Sie wissen, was Sie tun: Wenn Sie schon einmal unter Linux einen Kernel oder eine BusyBox kompiliert haben, kennen Sie den Aufbau des verwendeten Menusystems und können

Kaputt geflasht

➤ Es kam, wie es kommen musste: Übermütig kompiliert und mit zu vielen Änderungen an der Original-Firmware war das FRITZ!Box-Webinterface nicht mehr erreichbar. Der Trick bei der Reparatur einer totgeflashten FRITZ!Box liegt in einem vorgeschalteten, in einem separaten Teil des Flashspeichers abgelegten Bootloader. Dieser ist bei jedem Start der Box auf der IP-Adresse 192.168.178.1 für einige Sekunden per FTP erreichbar. Nach dem Log-in ist es möglich, ein Firmware-Image zu übertragen, das dann direkt auf die Flash-Partition geschrieben wird. Um den „FTP-Urlader“ nutzen zu können, muss die Box per Ethernetkabel mit Ihrem Rechner verbunden sein, WLAN geht nicht.

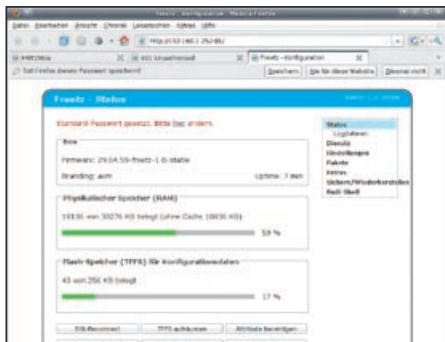
Da das Log-in und die händische Übertragung des Images ein punktgenaues Abpassen des richtigen Zeitpunktes erfordern würde, existieren zwei Werkzeuge, mit denen Sie die Übertragung starten. Freetz bringt bereits ein fertiges Shellscript mit, das sich im Ordner *Tools* befindet. Nach dem Wechsel in diesen Ordner können Sie das entpackte Original-Image mit dem Befehl

```
./push_firmware ../build/original/firmware/var/tmp/kernel.image
192.168.178.1
```

zur Box schicken. Unter Linux erledigt den gleichen Job das *AVM-Recovery-Tool*, welches direkt vom FTP-Server herunter-

geladen werden kann, für die 7170 beispielsweise unter ftp://ftp.avm.de/fritz.box/FRITZ!Box.fon_wlan_7170/x_misc/deutsch.

Klappt das Flashen nicht, sollten Sie verschiedene Ethernet-Ports der FRITZ!Box durchprobieren. Bei einigen Hardware-Revisionen ist der Urlader zunächst nur auf einem einzigen Port aktiv. Da es relativ schnell passieren kann, dass ein neues Image die FRITZ!Box lahmlegt, sollten Sie sowohl die *.config* als auch lauffähige Firmware-Images sichern und Änderungen in kleinen Schritten anbringen, um gegebenenfalls schnell zum letzten als zuverlässig funktionierenden Image zurückkehren zu können.



Hinter dem Link „Freetz“ im Webfrontend der FRITZ!Box verbirgt sich der Nebeneingang zur Freetz-Konfiguration.

die vielen Optionen sowie Fehlermeldungen interpretieren.

Auf gehts

Zunächst gilt es, die Systemvoraussetzungen zu erfüllen. Unter <http://www.freetz.org/wiki/help/howtos/common/install> finden Sie vier Zeilen mit *apt-get*-Befehlen, die Ihr System zuerst auf den aktuellen Stand bringen und dann die nötigen Programme installieren. Sollten einzelne Tools auf Ihrer Ubuntu- oder Debian-Version nicht verfügbar sein, entfernen Sie diese aus der Liste und rufen Sie den Installationsbefehl erneut aus. In der Regel handelt es sich bei den monierten Paketen um weniger wichtige Programme, die nur zum Bauen exotischer Komponenten notwendig sind.

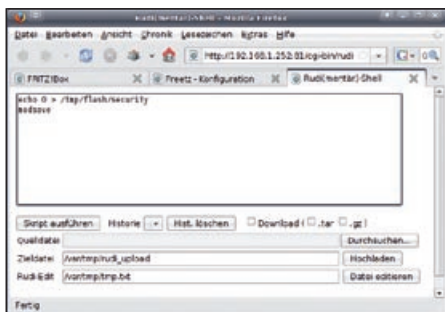
Anschließend folgen die Freetz-Skripte: Bei Redaktionsschluss existierte kein „stabiles“ Freetz-Skriptpaket, stattdessen mussten wir die bereits mit Patches versehene *Freetz-Stable-1.0* aus dem Subversion-Repository auschecken:

```
svn co http://svn.freetz.org/branches/freetz-stable-1.0 freetz-stable-1.0
```

Wechseln Sie anschließend in das entstandene Verzeichnis *freetz-stable-1.0* und rufen Sie dort

```
make menuconfig
```

auf. Nach einigen Sekunden erhalten Sie das vom Linux-Kernel bekannte *ncurses*-Menüsystem, in dem Sie die Konfiguration



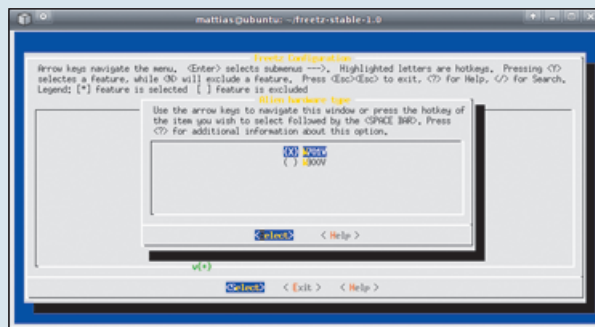
Nach dem Ändern des Sicherheitslevels...

Speedport „verfritzen“

Technisch eng mit der FRITZ!Box verwandt sind die von der T-Com vertriebenen *Speedport*-Router. Allerdings unterscheiden sich die Speedports in vielen kleinen Details von den FRITZ!Boxen und verwenden zudem ein etwas anderes Firmware-Format, um das Einspielen einer „fremden“ Firmware zu verhindern. Die verwendeten Änderungen zielen jedoch auf unbedarfte Nutzer und sind für versierte Anwender leicht zu umgehen. Derzeit existieren zwei Möglichkeiten, einen Speedport zur FRITZ!Box zu machen.

Die erste ist, zunächst eine FRITZ!Box-Firmware zu erstellen und diese dann mit dem Programm *Speed2Fritz* in eine Speedport-Firmware umzuwandeln. Als zweite Möglichkeit erlaubt Freetz die Erstellung von „Alien-Images“, die direkt auf den Speedport geladen werden können.

Beide Herangehensweisen haben Vor- und Nachteile, welche der Eintrag im IP-Phone-Wiki http://wiki.ip-phone-forum.de/skript:freetz_und_speed-to-fritz näher beschreibt.



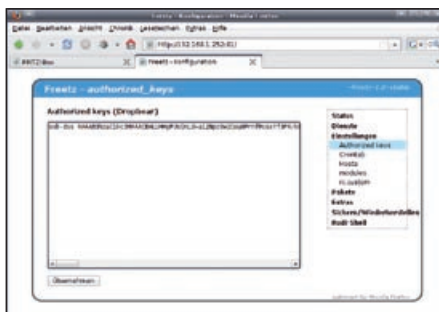
Gut getarnt: Auch im *Speedport* der T-Com steckt eigentlich eine FRITZ!Box – auch für diese kann Freetz Firmware-Images erstellen.

der eigenen Freetz-Firmware anpassen können. Halten Sie die Änderungen beim ersten Bauen moderat und vermeiden Sie die ganz großen Pakete. Ein guter Ausgangspunkt sind entfernte Hilfedateien (spart Platz), aber auch das Hinzufügen von *ctorrent* und dem Wake-On-LAN-Menüpunkt fürs Webinterface sowie dem kompakten SSH-Server *dropbear* fürs Remote-Log-in und Wartungsarbeiten auf der Shell der FRITZ!Box. Nach dem Speichern der Konfiguration startet *make* das Bauen der Firmware. Nach meist 30 bis 40 Minuten sollten der Bauvorgang abgeschlossen sein und Sie finden die Firmware-Datei im aktuellen Arbeitsverzeichnis wieder. Über das Webinterface der FRITZ!Box können Sie diese Firmware nun einspielen. Sollten Sie eine ältere Firmware verwenden, raten wir Ihnen zum zweistufigen Vorgehen: Spielen Sie zunächst die Firmware ein, welche die Basis für Ihre ei-

gene Firmware darstellt. So ist sichergestellt, dass die letzte funktionierende Original-Firmware und Ihre Freetz-Firmware die gleiche Partitionierung verwenden und Sie im Falle eines Falles leicht zur Original-Firmware zurückkehren können.

Das erste Log-in

Nach dem - hoffentlich erfolgreichen - ersten Reboot können Sie sich wie gewohnt an der Box anmelden. Im Webinterface weist lediglich der etwas versteckte Menüpunkt *Freetz* auf die neue Funktionalität hin. Über ihn gelangen Sie zu einer auf Port 81 erreichbaren, kennwortgeschützten Seite, auf der Sie sich mit dem Nutzernamen *admin* und dem Passwort *freetz* zum ersten Mal einloggen. Hier haben Sie die Möglichkeit, die globalen Einstellungen wie die Nutzung einer Auslagerungsdatei oder *-partition* und die Konfigura-



...kann ein SSH-Schlüssel hinterlegt werden,...



...der fortan das passwortlose Log-in und die automatische Ausführung von Befehlen auf der Box per Script vom Desktop aus erlaubt.

tion der hinzugefügten Pakete vorzunehmen. Eine rudimentäre Shell erlaubt die Ausführung einzelner Befehle, die keine Interaktion erfordern, direkt auf der Box.

Das Webinterface ist aus Sicht versierter Nutzer unschlagbar in seiner Direktheit: So erlaubt es die Übergabe von Parametern an gestartete Programme und die direkte Anpassung von Konfigurationsdateien. Um wirklich von den Möglichkeiten der so modifizierten Box profitieren zu können, sind gewisse Linux-Kenntnisse erforderlich. Einen Schritt weiter gehen die Möglichkeiten des Shell-Logins per SSH: Hier können Sie direkt Programme starten, den aktuellen Speicherbedarf ermitteln und Laufwerke mounten oder unmounten.

Einladung zum Spielen

Die Erstellung einer eigenen Freetz-Firmware und deren Installation kann daher eher als eine Art spaltweite Öffnung einer Tür verstanden werden. Dahinter verbergen sich enorme Möglichkeiten, die jedoch ein kundiges Händchen erfordern.

Weblinks

- www.freetz.org/wiki/FAQ
Der ehemalige *ds-mod* wird nun von einer größeren Entwicklergemeinschaft unter einer eigenen Domain gepflegt.
- www.freetz.org/wiki/help/howtos/common/install
Eine ausführliche Installationsanleitung stellt das Freetz-Wiki zur Verfügung.
- http://cdprojekte.mattiaschlenker.de/Public/Artikel/PC-Magazin_Linux_2007_04_-_Boxenstop.pdf
Details zum Aufbau des Linux der FRITZ!Box beschreibt der Artikel *Boxenstop* aus PC Magazin Linux 04/2007 von Mattias Schlenker
- <http://ippf.eu/index.php>
Eine gute Anlaufstelle für das Anpassen von FRITZ!Box und Speedport ist das IP-Phone-Forum

Ein weiterhin wichtiger Einstiegspunkt für weitere Modifikationen ist die bekannte /*var/flash/debug.cfg*, mit der Sie zusätzliche Programme beim Systemstart aktivieren können, auch wenn diese nicht Teil der Firmware sind, sondern auf separaten Datenträgern (beispielsweise einem USB-Stick) liegen. Daneben haben Sie die Möglichkeit, eine *rc.custom* anzulegen, die am Ende des Boot-

vorganges ausgeführt wird. Praktisch sind beim Mounten und Unmounten ausgeführte Scripte, mit denen Sie beispielsweise beim Anstöpseln einer Festplatte einen UPnP-Server oder einen Bittorrent-Client starten können.

Wer fit mit dem Compiler ist, kann sich darüber hinaus mit dem von Freetz mitgelieferten Crosscompiler austoben und zusätzliche Binaries erstellen, die er auf einen EXT3-formatierten USB-Stick kopiert und gegebenenfalls von *rc.custom* ausführen lässt.

Anwender mit tiefen Linux-Kenntnissen können Freetz so auch mit Programmen erweitern, die im offiziellen Freetz-Paket nicht vorhanden sind. Das Risiko so hinzugefügter zusätzlicher Applikationen ist relativ gering - im Zweifel hilft ein Reboot ohne den Datenträger mit den eigenen Anwendungen.

Firmware aus dem Labor

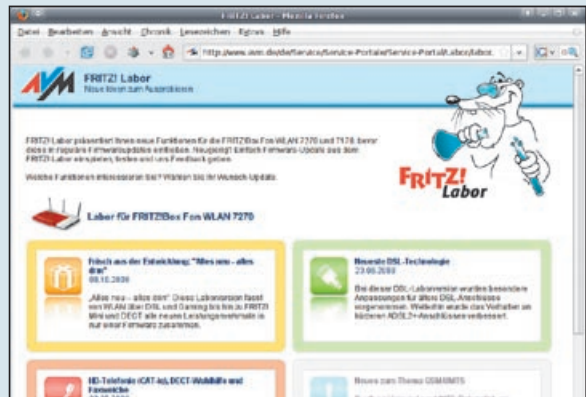
➤ **AVM selbst steht gemoddeten FRITZ!Boxen zwiegespalten gegenüber: Einserseits erkennt der Kommunikationshersteller an, dass die Modifizierbarkeit ein wesentlicher Erfolgsfaktor der Fritz!-Familie ist, andererseits ist der Supportaufwand bei vermeintlich kaputteten Boxen durch nicht autorisierte Modifikationen größer als bei komplett verschlossenen Systemen.**

AVM versucht daher, den Bastelrdrang vor allem weniger versierter Anwender in Bahnen zu lenken, die ein geringeres Risiko bergen. Aktuell ist dies der Fall bei AVMs Labor-Firmware, mit der neue Funktionen ausprobiert werden können, beispielsweise die FRITZ!Box als UMTS-Router via Huawei-USB-Stick oder die Vernetzung per Wireless Distribution System. Je nach Akzeptanz und Stabilität der neuen Features wandern die Funktionen früher oder später in die reguläre Firmware.

Dennoch wäre mehr schön: Wir würden uns wünschen, dass AVM nicht

nur die Entwicklung öffnet, sondern auch die Plattform der FRITZ!Box auf Erweiterungen auslegt.

Denkbar wäre beispielsweise ein Fritz-eigenes Paketformat auf Basis von Squashfs-Containern, die der Nutzer auf einen USB-Stick kopieren kann und deren enthaltene Programme bei Start der Box automatisch integriert werden. Das Risiko einer solchen Erweiterbarkeit wäre bei Beachtung gängiger Sicherheitsregeln moderat, und bei Stabilitätsproblemen würde es genügen, die FRITZ!Box ohne USB-Stick zu starten.



Offizielle Basteleien: Labor-Firmwares von AVM erlauben Ausblicke auf künftige Firmwareversionen.

Grenzen der Hardware

Freetz verlockt: Was könnte man alles per USB an die kleinen Boxen hängen? Wie wäre es mit einem DVB-T-Videorecorder, einer vollwertigen NAS-Lösung, einer USB-Audio-Karte, welche die Box gleich zur Stereoanlage macht, einem Mailserver auf USB-Stick, einem vollwertigen LAMP-Stack für den eigenen DynDNS-Webserver?

Nicht alle dieser Ideen dürften sich befriedigend realisieren lassen: So stellt bei der FRITZ!Box 7170 die Langsamkeit des USB-1.1-Ports oft einen Flaschenhals für hochtrabende Ideen dar, und der Arbeitsspeicher schränkt mit 32 oder 64 MByte die Zahl gleichzeitig laufender Applikationen ein. Die Nutzung von *Swap* mag die Grenze etwas verschieben, wenn jedoch andauerndes Swappen auftritt, ist es mit der Performance vollends vorbei. Eine weitere Grenze stellt der schon angestaubte Kernel 2.6.13 dar: Für ihn gibt es kaum moderne Treiber für USB-DVB-T-Sticks, Bluetooth, USB-Audio und was sonst noch verlockend erscheint.

jkn