

# Implementation der Dienste Privoxy und TOR auf einem NAS Embedded-Device via Cross-Compiling



**Christian Franzen**  
**Matrikelnummer 137519**  
**FOM Neuss**

**Bachelor of IT-Engineering**  
**Gutachter und Betreuer: Prof. Dr. Finke**

**Dormagen, den 23.02.2009**

## Vorwort

Die Themenwahl ergab sich aus dem Zusammentreffen zweier Ereignisse. Das erste Ereignis ist der momentan stark anwachsende Wunsch nach Überwachung von Seiten vieler großer Industrienationen, der nach Meinung des Autors bedrohliche Ausmaße angenommen hat. Eine Möglichkeit sich diesem Generalverdacht unbescholtener Bürger zu entziehen besteht darin, möglichst wenig Informationen über sich preiszugeben. Im Internet helfen bei diesem Vorhaben Verschleierungsdienste wie Privoxy und TOR.

Das andere Ereignis trat durch den Kauf des Western-Digital Netcenters Ende 2006 ein. Diese Netzwerkfestplatte sollte die Verteilung von Mediendaten im lokalen Heimnetzwerk vereinfachen, als zentrale Datenablage dienen und durch die ständige Verfügbarkeit verhindern, dass stets alle Rechner eingeschaltet bleiben. Das kleine Gerät mit Windowsfreigabe entpuppte sich als verhältnismäßig leistungsfähiger Linuxserver, den man mit Hilfe vieler Arbeitsstunden und dem zusammengetragenen Wissen einiger Internetforen erweitern konnte. Besonders faszinierte den Autor hier die Ressourcenausnutzung des integrierten Linux-Systems. Weiterhin erstaunte die Leistungsfähigkeit des integrierten MIPSel Prozessors trotz geringer Taktung. Die Beschäftigung mit dem integrierten Linux Anfang 2007 führte zu stetigen Lerneffekten im Linux-Umfeld, sodass später auch der Desktoprechner auf Linux migriert werden konnte. Nachdem einige von einer Internet-Community zur Verfügung gestellten fertig-Pakete auf dem Netcenter installiert wurden und sich monatelang bewährten, sollte das Selbstkompilieren von freien Anwendungen als nächstes Ziel erreicht werden. Die mittlerweile zweijährige Erfahrung mit Linux sollte hierbei unterstützen.

Dieses größere Vorhaben lies sich ideal mit der für das Studium fälligen Abschlussarbeit verbinden.

## **Inhaltsverzeichnis**

<b>1</b>	<b>Einleitung</b>	<b>8</b>
1.1	Titel der Arbeit . . . . .	8
1.2	Zur Durchführung der Arbeit . . . . .	8
1.3	Anonymisierung . . . . .	8
1.4	Zielsetzung . . . . .	10
1.5	Textgestalterische Konventionen . . . . .	12
<b>2</b>	<b>Verwendete Hard- und Software</b>	<b>13</b>
2.1	Hardware . . . . .	13
2.1.1	Netzwerkumgebung . . . . .	13
2.1.2	Western-Digital Netcenter . . . . .	13
2.2	Software . . . . .	15
2.2.1	Kernel . . . . .	15
2.2.2	Paketmanager IPKG . . . . .	16
2.2.3	Administrative Software: Bash, SSH-Server und FTP-Server . . . . .	17
2.2.4	Startup Parameter und Bug-Workarounds . . . . .	17
2.2.5	DHCP und DNS . . . . .	21
2.2.6	Lighttpd . . . . .	23
2.2.7	Httpd-Performanceoptimierung . . . . .	25
2.3	Einführung in Proxy Auto Configuration . . . . .	25
2.4	Einführung in das Web Proxy Autodiscovery Protocol . . . . .	27
2.5	Alternative Lösungsmöglichkeit: Zwangsproxy . . . . .	29
<b>3</b>	<b>TOR und Privoxy, Cross-compiling</b>	<b>32</b>
3.1	Funktionsweise und Schwächen von TOR . . . . .	32
3.2	Funktionsweise von Privoxy . . . . .	37
3.3	Natives vs. Cross-Compiling . . . . .	40
3.4	Toolchain und Compiling . . . . .	41
3.5	Erste erfolgreiche Compiling-Versuche: prime und busybox . . . . .	51
3.6	Bibliotheken . . . . .	59
3.7	Cross-Compiling und Konfiguration von TOR . . . . .	65
3.8	Cross-Compiling und Konfiguration von Privoxy . . . . .	69
3.9	Cross-Compiling: Fehlerbehebung . . . . .	75
3.10	TOR-Button . . . . .	78

---

<b>4</b>	<b>Geschwindigkeitsmessung</b>	<b>80</b>
<b>5</b>	<b>Fazit</b>	<b>85</b>
5.1	Erreichte Ziele . . . . .	85
5.2	Blockierte Google-Suche . . . . .	86
5.3	Ausblick . . . . .	87
<b>A</b>	<b>Softwarekonfiguration des Netcenters</b>	<b>89</b>
A.1	xinetd . . . . .	89
A.2	Firmware entpacken . . . . .	93
A.3	Firmware verändern . . . . .	97
A.4	Firmware packen und aufspielen . . . . .	98
A.5	FTP-Server Vsftpd . . . . .	101
A.6	SSH Zugriff - Dropbear . . . . .	102
A.7	Bashrc . . . . .	104
A.8	IP-Updateskript . . . . .	107
A.9	SSLwrap . . . . .	111
A.10	Bit Torrent Protocol Daemon . . . . .	114
A.11	Weitere Anwendungen . . . . .	118
A.12	Startup-Skript rc.start . . . . .	119
<b>B</b>	<b>Kompilation von Dnsmasq, Dropbear und Bash</b>	<b>121</b>
B.1	Dnsmasq kompilieren . . . . .	121
B.2	Dropbear kompilieren . . . . .	123
B.3	Bash kompilieren . . . . .	124

Index138

## Abbildungsverzeichnis

1	Die Netzwerk-Infrastruktur . . . . .	14
2	Eingabe der URL für die PAC-Datei beim Firefox. . . . .	27
3	Abfrage der PAC via DHCP . . . . .	28
4	Abfrage der PAC via DNS . . . . .	29
5	Einstellung des Firefox für die Nutzung von WPAD . . . . .	30
6	Einstellung des IE für die Nutzung von WPAD . . . . .	31
7	Namensauflösung über TOR als Socks4a-Proxy. . . . .	32
8	Namensauflösung mit Socks4 Proxy, Datentransfer über TOR. . . . .	32
9	Zusammenspiel der Kommunikation aller Dienste auf dem Netcenter . . . . .	39
10	Busybox: Menu von “make menuconfig” . . . . .	49
11	Auswahl der Bibliotheken im “make menuconfig” von Freetz . . . . .	50
12	Ausführung von Prime auf dem Netcenter . . . . .	53
13	Ausführung von Busybox auf dem Netcenter . . . . .	57
14	heise.de/ip zeigt Exitnode . . . . .	75
15	Der Firefox Add-On Dialog zum Einstellen von TOR-Button . . . . .	78
16	Anpassen der TOR-Button Proxy-Einstellung . . . . .	79
17	Vergleich der Ladezeiten verschiedener Seiten über Privoxy/TOR lokal und auf dem Netcenter . . . . .	81
18	Googles Fehlerseite “automatische Anfrage” bei Suche über TOR, je nach Landessprache der Exit-Node . . . . .	84
19	Bless: Ablesen des Offset im unteren Bildschirmbereich nach Stringsuche . . . . .	95
20	Upload der Firmware über das Webinterface . . . . .	100
21	Stringsuche nach /dev/random . . . . .	105
22	/dev/random durch /dev/urandom ersetzen und Verkürzen des nachfol- genden Strings . . . . .	106
23	Zustände der Statusleuchte an der Gerätefront . . . . .	108

## Listings

1	IPKG-Konfiguration /opt/etc/ipkg.conf . . . . .	16
2	Samba-Konfiguration . . . . .	19
3	DNSmasq-Konfiguration . . . . .	22
4	Lighttpd-Konfiguration . . . . .	24
5	PAC-File des Netcenters . . . . .	26
6	Prime: Programm zur Primfaktorzerlegung . . . . .	44
7	Shell-Variablen für ./configure . . . . .	47
8	hnd-toolchain 3.23: Busybox Compilerfehler . . . . .	48
9	Kompilationsvorgang von Prime . . . . .	52
10	Fehlerhaftes Linken der Busybox mit Freetz . . . . .	54
11	Kompilation der Busybox . . . . .	55
12	Busybox 1.13.2: Probleme mit Insmod . . . . .	56
13	Arbeit mit ldd.sh . . . . .	63
14	GPG-Schlüsselimport für TOR . . . . .	65
15	Ausgabe der Signaturprüfung des TOR-Quelltextes . . . . .	66
16	TOR-Compiling und Stripping . . . . .	67
17	TOR-Konfiguration torrc . . . . .	68
18	Meldungen beim Starten von TOR auf dem Netcenter . . . . .	68
19	Ausgabe der Signaturprüfung des Privoxy-Quelltextes . . . . .	69
20	Privoxy Autoheader-Dialog . . . . .	70
21	Fehler im Privoxy-Configure Vorgang . . . . .	70
22	Privoxy-Compiling und Stripping . . . . .	71
23	Privoxy-Konfiguration . . . . .	73
24	Privoxy Beispielfilter . . . . .	74
25	Meldungen beim Starten von Privoxy auf dem Netcenter . . . . .	74
26	Log einer TOR-Geschwindigkeitsmessung . . . . .	82
27	Xinetd-Konfiguration . . . . .	91
28	Flashen der Firmware über Nasload.exe . . . . .	101
29	Vsftpd-Konfiguration . . . . .	102
30	Geschwindigkeitsvergleichen zwischen /dev/urandom und /dev/random .	103
31	/etc/profile . . . . .	105
32	ipupdate.sh . . . . .	109
33	xinetd-Eintrag für SSLwrap . . . . .	114
34	Bittorrent: btstart . . . . .	115

---

35	Bittorrent: btadd . . . . .	116
36	Bittorrent: bt . . . . .	116
37	Bittorrent: bt-da . . . . .	117
38	Bittorrent: btkill . . . . .	117
39	Startskript rc.start . . . . .	119
40	Dnsmasq-Compile: IPv6-Fehlermeldung . . . . .	121
41	Dnsmasq-Compile dynamisch gelinkt ohne IPv6 . . . . .	122
42	Dnsmasq dynamisch gelinkt: Fehlermeldung . . . . .	122

# 1 Einleitung

## 1.1 Titel der Arbeit

Der Titel der Arbeit steht für die Implementation des Anonymisierungsdienstes TOR zusammen mit dem Internetfilter Privoxy auf einem NAS-Device, das ursprünglich ausschließlich zum Abspeichern von Daten im Netzwerk konstruiert wurde. TOR und Privoxy werden dabei mittels Cross-Compiling in ein für die Netzwerkfestplatte verständliches Binärformat übersetzt. Cross-Compiling bedeutet, dass der Compiler auf einer anderen Rechnerarchitektur läuft, als die durch den Kompilervorgang erzeugte Binärdatei.

## 1.2 Zur Durchführung der Arbeit

Erste Teile dieser Arbeit wurden während der praktischen Durchführung geschrieben. Da stetig neuere Erkenntnisse gewonnen wurden, mussten daher Teile der Arbeit mehrmals überarbeitet werden. Um den daraus resultierenden Mehraufwand zu vermeiden, wurde nach einigen Überarbeitungsvorgängen zuerst der praktische Teil fertiggestellt und im Anschluss die gesamte Thesis überarbeitet. Neue Ideen, die ab diesem Zeitpunkt nicht mehr zur Umsetzung kamen, wurden im Fazit der Arbeit als Ausblick behandelt. Dieses Vorgehen bedingte, dass die Implementierung von TOR und damit der Beweis der technischen Machbarkeit dem Autor bereits zu diesem Zeitpunkt bekannt war, daher wurden Stellen mit Alternativmöglichkeiten im Falle einer aus welchen Gründen gearteten nicht-Machbarkeit gestrichen. Aber warum wird überhaupt ein Anonymisierungsdienst benötigt?

## 1.3 Anonymisierung

Im Monatsrhythmus werden von der amerikanischen, als auch von den europäischen Regierungen Pläne und Wünsche veröffentlicht, wie man die Überwachung des Bürgers ausbauen kann. Eine Zusammenfassung von Vorschlägen und Gesetzesentwürfen findet sich in [1]. Dazu gehören die Gesetze zur Vorratsdatenspeicherung (VDS), Fingerabdruck

und RFID-Chip im Reisepass, elektronische Gesundheitskarte, Mautdatenerfassung und Kennzeichen-Scanning, Bundestrojaner und die einheitliche Steuernummer. Erfahrungsgemäß geht mit der längerfristigen Speicherung der Daten eine Zweckentfremdung einher. Die ursprünglich zu Abrechnungszwecken gespeicherten Mautdaten werden mittlerweile für Kennzeichen-Scanning genutzt. Es ist davon auszugehen, dass die Zweckentfremdung weiter ausgebaut wird. [2] Teilweise wird die Nutzung dieser ursprünglich ausschließlich zu staatlichen Zwecken gesammelten Daten auch für private Institutionen gestattet. Darauf, dass diese Ausweitung der Zugriffsberechtigung ausschließlich eine Zeitfrage ist, gibt neben dem LKW-Mautsystem auch die Vorratsdatenspeicherung gute Hinweise [3].

Mit diesem Datenmissbrauch geht der Verlust auf informationelle Selbstbestimmung einher. Mit der zentralen und lebenslangen Steuernummer [4] ist die Identität jedes Menschen mitsamt seiner Adresse, seinem Einkommen, seinem Fingerabdruck und seiner Religion zusammen zentral gespeichert. Ausgerechnet die deutsche Geschichte liefert ein gutes Beispiel dafür, dass es möglich ist, Daten dieser Art gegen den Bürger eines Staates zu verwenden. Eine nach heutigem Gesetz noch zulässige Möglichkeit, private oder staatliche Institutionen daran zu hindern, persönliche Daten zu sammeln ist, sie nicht preiszugeben. Zwar ist das nicht das universelle Hilfsmittel für alle Einsatzlagen, leistet aber beim Surfen im Internet, also gegen die Vorratsdatenspeicherung gute Dienste. Anonymisierer wie Privoxy und Tor helfen dabei maßgeblich.

Verschlüsselung ist für die Verschleierung von persönlichen Informationen nicht ausreichend. Verschlüsselung verhindert zwar die Einsicht in die Nutzdaten für Dritte, sie verhindert jedoch nicht die Verknüpfung von Personendaten mit dem aufgerufenen Dienst. Am konkreten Beispiel bedeutet dies, das zwar beim Zugriff von einem Rechner auf einen Server einer Bank der Kontostand für Dritte nicht ersichtlich ist, einem Dritten aber sehr wohl ersichtlich ist, bei welcher Bank das entsprechende Konto liegt. Beim Aufruf von ohnehin frei zugänglichen Datendiensten ist daher Verschlüsselung nicht hilfreich. Zu diesen frei zugänglichen Diensten gehört, um bei dem Beispiel zu bleiben, die Namensauflösung der Bank-Webpräsenz zur IP-Adresse. Nur durch zusätzliche Verschleierung der Zugriffe ist anonymes Surfen im Internet gewährleistet.

Die Motivation für die Anonymisierung kann die verschiedensten Ursachen haben: Von der Preisgabe der eigenen sexuellen Orientierung bis hin zu politischen Interessen, die in China[21] mit der Todesstrafe geahndet werden, unterliegen dem Einsatzfeld kaum

Grenzen. Aber auch altruistische Gründe können Grund für den Wunsch nach Verschleierung sein: Möchte man der Nachbarschaft freies WLAN zur Verfügung stellen, kann man wegen Mitstörerhaftung herangezogen werden, falls einer der anonymen Gäste Straftaten unter Verwendung des eigenen Internetanschlusses begeht.[22] Schleust man den Datenverkehr durch einen Anonymisierer, kann das zur Stressvermeidung beitragen.

Wer den Artikel über die “Militante Gruppe” auf der BKA Website Anfang des Jahres 2007 besuchte, wurde pauschal als Anhänger dieser Gruppe verdächtigt. Das BKA fragte zu allen Besuchern die Personendaten anhand der IP-Adresse beim Provider an. Es fand eine Verdächtigung alleine aufgrund der Tatsache statt, eine Website mit Nachrichten einer staatlichen Organisation aufgerufen zu haben. Dieses Vorgehen spricht diametral gegen die Unschuldsvermutung.[19]

Da auch die Internetzensur in Deutschland unter dem Vorwand der Kinderpornographiebekämpfung<sup>1</sup> eine beschlossene Sache ist, ist auch der Zensuraspekt in Deutschland ein weiterer Grund zur Verschleierung. “Aber doch nicht in Deutschland!” gehört der Vergangenheit an.[20]

## 1.4 Zielsetzung

Der rote Faden dieser Abschlussarbeit ist die Einrichtung eines Anonymisierungsdienstes via Crosscompiling auf Basis von TOR und Privoxy auf einem NAS-Device, in diesem Fall dem Netcenter der Firma Western Digital.

Die Arbeit adressiert die Zielgruppe, die theoretische sowie praktische Information über die Modifikation von Embedded Devices auf Linux Basis sucht. Der Aufbau der Embedded Devices verschiedener Hersteller, von DSL-Routern bis hin zu Netzwerkfestplatten, ist ähnlich, insofern kann die hier vorgestellte Vorgehensweise auf andere Geräte übertragen werden.

---

<sup>1</sup>Sie wird auch als “Missbrauch des Kindesmissbrauch” bezeichnet. Ursachenbekämpfung oder internationale Zusammenarbeit beim Ergreifen der Täter findet nicht statt[75], lediglich das Konsumieren soll erschwert werden. Die ebenfalls geplante Erweiterung zur Sperrung sämtlicher Glückspielangebote mit Ausnahme der staatseigenen zur Suchtprävention ist ebenfalls angedacht.[20]

Diese Arbeit stellt zum Einen eine vollständige Anleitung dar, um ein Embedded Device auf Linux Basis um zusätzliche Software zu erweitern. Zum Anderen liefert diese Arbeit das theoretische Hintergrundwissen über die wichtigsten verwendeten Dienste. Weiterhin wird die Interaktion dieser Dienste in der vom Autor implementierten Form erklärt, dazu gehören:

- DNS, DHCP
- WPAD; bestehend aus HTTP, PAC
- TOR
- Privoxy

Ebenfalls geht diese Arbeit auf die prinzipiellen Schwachstellen von TOR ein. Dieses Wissen ist unerlässlich, um sich vollständig anonym im Internet zu bewegen.

Um besseres Verständnis für das (Cross-)Compiling zu vermitteln, werden Cross-Compiling und natives Compiling verglichen, sowie die Unterschiede zwischen statischem und dynamischen Linken erläutert. Aufgrund der geringen Leistungsfähigkeit des Gerätes erfolgte am Ende der Implementierungsphase eine Geschwindigkeitsmessung.

Erste Änderungen an den auf dem Netcenter laufenden Diensten wurden bereits Anfang 2007 durchgeführt. Im Rahmen dieser Arbeit wurden viele dieser Änderungen von Grund auf neu implementiert. Damit die große Anzahl der Arbeitsstunden an Wissensaufbereitung auch anderen Menschen von Nutzen sind, wurde mit dieser Arbeit ein Kompromiss aus wissenschaftlicher Informationsaufbereitung und Best-Practise-Handbuch versucht. Informationen, die für das Titel-Thema dieser Arbeit von untergeordneter Relevanz sind, sowie viele Teile, die unter den Aspekt des Best-Practise fallen, wurden in den Anhang ausgegliedert. Auf sie wird an entsprechender Stelle innerhalb der Arbeit verwiesen.

An den Leserkreis dieser Arbeit, der sich in das Cross-Compiling einarbeiten möchten, folgt noch eine kleine Warnung: Das Verwenden von Linux als Basissystem für das Cross-Compiling ist notwendige Voraussetzung. Wurden eine Vielzahl an Programmen unter Verwendung einer Vielzahl an Toolchains unter Linux erfolgreich kompiliert, kann - sofern dann noch erwünscht - das Kompilieren unter Windows mit Hilfe von Cygwin ge-

testet werden. Die Probleme der Cygwin-Inkompatibilitäten addieren sich dann zu den üblichen Schwierigkeiten mit der Cross-Compiler Toolchain hinzu. Wer unter Linux nie kompiliert hat, sollte es unter Windows nicht versuchen. Wer Angst vor Linux auf der eigenen Festplatte hat und mit der Performanz von virtuellen Maschinen nicht zufrieden ist, dem hilft eine Wubi-Installation von Ubuntu. Diese schreibt ein vollwertiges Linux ohne Umwege über eine virtuelle Maschine in eine große NTFS-Datei und verändert so das bestehende System mit Ausnahme eines Windows-Bootmenü-Eintrages nicht. Der gravierende Nachteil diverser Live CDs, die fehlende "bequeme" Schreibunterstützung, besteht ebensowenig.[23]

Da der Wissensstand des Leser in verschiedenen Bereichen unterschiedlich stark ausgeprägt ist, versucht diese Arbeit eine nahezu lückenlose Erklärung innerhalb der Arbeit oder im Glossar zu liefern. Dennoch wird vom Leser erwartet, dass er nicht nur Befehlszeilen herauskopieren und wieder einfügen kann, sondern den Hintergrund seiner Tätigkeit versteht. Um das zu Unterstützen, werden insbesondere die Punkte, bei denen der Autor selbst Schwierigkeiten hatte, oder bei denen er Schwierigkeiten vermutet, besonders erläutert und mit typischen Fehlermeldungen untermauert. Um die Fachbegriffe großzügig und lückenlos verständlich zu machen, finden sich diese am Schluss der Arbeit im Glossar. Um die Arbeit für letzteres in Grenzen zu halten, wurden - wenn vorhanden - Begriffsbeschreibungen oder Erläuterungen aus freien Quellen geprüft und in dieser Arbeit in das Glossar übernommen und entsprechend gekennzeichnet.

## 1.5 Textgestalterische Konventionen

*Kommandozeileingaben* sind in kursiv abgedruckt

Fließtext ist in Serifenschrift geschrieben

Quellcode und Bildschirmausgaben sind in Fixschrift gehalten

## 2 Verwendete Hard- und Software

### 2.1 Hardware

#### 2.1.1 Netzwerkkumgebung

In dem vom Autor verwendeten Heimnetzwerk befinden sich neben einigen Windows- und einem Linux-Client noch weitere Geräte, die hier der Vollständigkeit halber aufgezählt werden. Als Internetrouter für eine DSL-Verbindung ist ein SMC Barricade 7404brb zu nennen, über dessen paralleler Schnittstelle ein Laserdrucker via LPR Dämon freigegeben ist. Der DSL-Router diente ursprünglich als DHCP-Server und als DNS-Relay. Eine tiefgreifende Konfiguration dieser Dienste war nicht möglich. Es lies sich ein alternativer DNS-Server eintragen, der dann vom Relay des Routers genutzt wurde. Nicht ändern lies sich der dem Client via DHCP mitgeteilte DNS-Server, um den fehlerhaften Relay-DNS des Routers zu umgehen. Dieser lieferte nach einiger Zeit unter nicht reproduzierbaren Umständen mx-Records in umgekehrter Oktettfolge zurück. Der Fehler lies sich nur durch einen Neustart des Routers beheben. Daher wurde die DHCP- und DNS-Funktionalität abgeschaltet und stattdessen die neu implementierten Dienste auf der Netzwerkfestplatte genutzt.

Die Clients sind je nach räumlicher Position über nicht managbare Layer-2-Switches mit dem DSL-Router verbunden. Das Netz ist ein homogenes 100 MBit Ethernet mit Full-Duplex Kommunikation. Die Geräte setzen hierbei auf Auto-Negotiation. Ein WLAN ist nicht vorhanden. Die Netzwerkfestplatte hängt an einem der Switches.

#### 2.1.2 Western-Digital Netcenter

Die Netzwerkfestplatte ist aus der Reihe „Netcenter“ von Western Digital. Es handelt sich um eine passiv gekühlte 3,5” Festplatte mit Netzwerkanchluss.

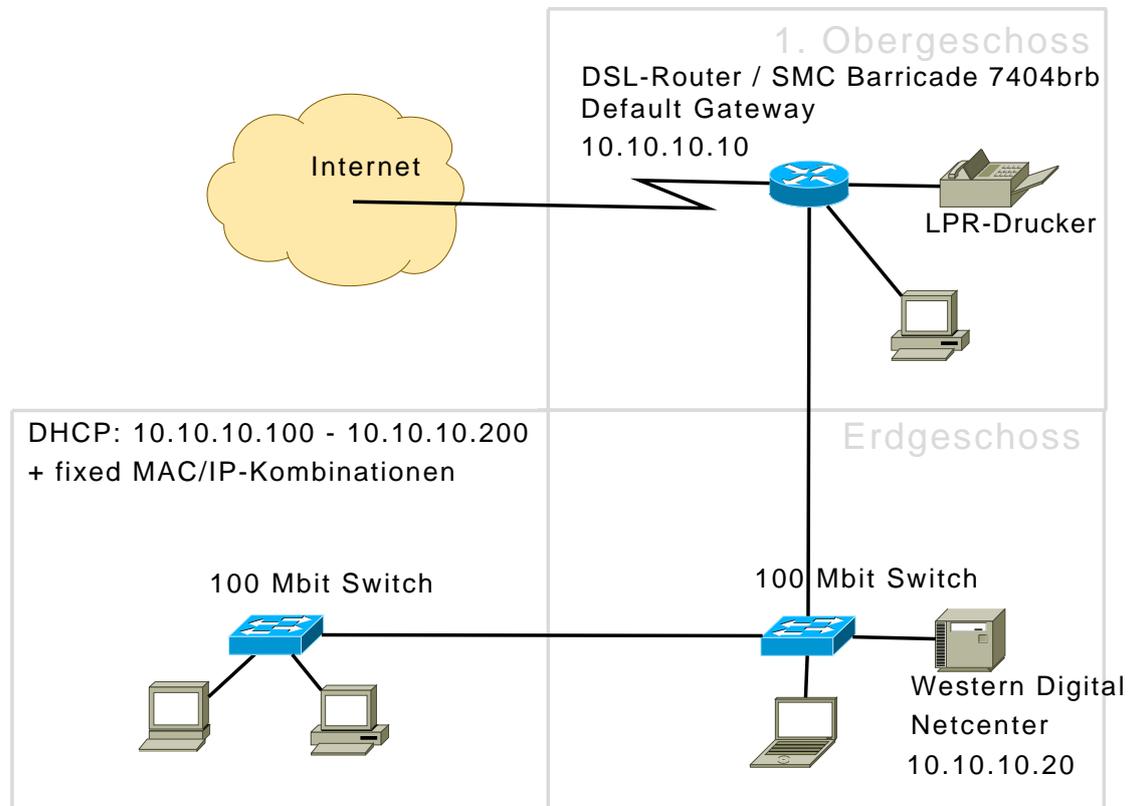


Abbildung 1: Die Netzwerk-Infrastruktur

Das Netcenter ist mit einem Energieverbrauch von ca. 25 Watt unter Last <sup>2</sup> sparsam im Energieverbrauch, vergleicht man es mit Server-Hardware, die ähnliche der in dieser Thesis vorgestellten Dienste zur Verfügung stellt. Aufgrund seiner Einsatzart als ständig verfügbarer Netzwerkspeicher ist es unter ökologischen Aspekten ein idealer Kandidat für einen Anonymisierungsdienst im Heimnetzwerk, unterstellt man, dass das Gerät als Netzwerkspeicher für den gesamten Haushalt ohnehin dauerhaft in Betrieb ist und somit keine zusätzlichen Energiekosten generiert werden.

Die vom Hersteller vorgesehene Aufgabe beschränkt sich auf die Freigabe von Speicherplatz der integrierten Festplatte als Netzlaufwerk über die Protokolle Samba und NFS, sowie die Namensauflösung via Windows Naming Service (WINS). Die Konfiguration findet über ein Web-Interface statt, auf dem Nutzer angelegt, als auch Freigaben erstellt und mit Passwort versehen werden können. Das Gerät besitzt einen MIPSel Prozessor,

<sup>2</sup>Der Energiebedarf wurde mit einem Messgerät an der Steckdose gemessen, das Netzteil ist für eine Leistung von  $12V \times 3A = 36$  Watt ausgelegt.

der mit 266 MHz getaktet ist. Es verfügt über 32 MB Arbeitsspeicher und nutzt als Betriebssystem ein abgespeckten und modifizierten Linux 2.4 Kernel. Als Dateisystem auf der internen 320 GB IDE Festplatte kommt ein ReiserFS-Dateisystem zum Einsatz. Da der Master-Boot-Record (MBR) der Festplatte proprietär ist, in welchem sich die Partitionstabelle befindet, kann die Datenpartition der Festplatte nicht ohne Anpassung des MBR an einem Desktop-System gelesen werden. Ein gültiger MBR kann mit Recoveryprogrammen wie Testdisk [76] erstellt werden.

Zur Konsistenzprüfung des Dateisystems ist ein Programm namens “reiserfsck” verfügbar. Nutzbar ist dieses nicht: Die Platte kann nicht ausgehängt werden, da dort zum Systembetrieb notwendige Bibliotheken liegen. Aus den oben genannten Gründen kann der File-System-Check auf einem Desktop System nur mit hohem Aufwand und unter dem Risiko des Datenverlustes durchgeführt werden.

Durch den Einsatz der MIPSel Prozessors ist das System nicht Binärkompatibel mit gängigen Desktop-Rechnern, die bis auf wenige Ausnahmen einen Prozessor mit Intel x86-Architektur besitzen. Nähere Informationen zu den unterschiedlichen Prozessordesigns finden sich unter [16].

## 2.2 Software

### 2.2.1 Kernel

Die auf dem Gerät vorinstallierte Software beschränkt sich auf einen Linux Kernel Version 2.4.20 mit Samba in Version 3.0.2, einem NFS-Daemon und einem HTTP-Server als Web-Konfigurationsoberfläche. Im ersten Schritt wurde durch eine modifizierte Firmware, welche in einem deutschsprachigen Internetforum[6] gefunden wurde, der Zugang via Telnet auf das Gerät freigeschaltet und ein FTP-Server nachgerüstet.

### 2.2.2 Paketmanager IPKG

Im zweiten Schritt wurde der Paketdienst IPKG installiert. IPKG steht für “Itsy Package Management System” und ist ein vom Debian Packet Management System abgeleiteter Paketmanager. Als Grundlage zur Durchführung der Installation wurde eine Beschreibung im Netcenter-Forum[6] verwendet.

Das ipkg-Binary wurde via SMB auf einen freigegebenen Ordner der Festplatte kopiert und über die Kommandozeile in den /opt/bin Ordner verschoben, der auch in der PATH-Variable enthaltenen ist. Im Anschluss wurden Paketquellen in die Konfigurationsdatei des IPKG eingetragen, welche sich in einem Internetforum für ein Mitbewerberprodukt der Firma Maxtor mit gleicher Prozessorarchitektur fanden[12].

Listing 1: IPKG-Konfiguration /opt/etc/ipkg.conf

---

```
1 src bryan http://ipkg.openmss.org/bryan
2 src experimental http://ipkg.openmss.org/experimental
3 src unslung http://ipkg.openmss.org/unslung
4 dest root /
```

---

Die dort verfügbaren Pakete wurden ursprünglich für das OpenWRT-Projekt erstellt und von den jeweiligen Autoren mit den von Maxtor bereitgestellten Toolchain Cross-kompiliert. Durch die Automatisierung dieses Prozesses und die dadurch fehlenden Tests wird von den Autoren keine Funktionsgarantie für die Pakete übernommen und die Paketquelle als „unstable“ deklariert.

Der Aufruf zum Abgleich der Paketquellen, dem anschließenden Installationvorgang oder dem Entfernen von Paketen geschieht mittels:

```
ipkg-cl -f /opt/etc/ipkg.conf update
ipkg-cl -f /opt/etc/ipkg.conf install PAKETNAME
ipkg-cl -f /opt/etc/ipkg.conf remove PAKETNAME
```

### 2.2.3 Administrative Software: Bash, SSH-Server und FTP-Server

Als Basis zur Administration stellten sich neben einigen Bibliotheken aus der von Western Digital bereitgestellten Toolchain[24] besonders die Busybox sowie die Bash heraus. Weiterhin wurde Dropbear als SSH-Server und Vsftpd als FTP-Server installiert. Näheres zur Konfiguration von Dropbear und Vsftpd findet sich im Anhang [A.5](#)

### 2.2.4 Startup Parameter und Bug-Workarounds

Einige Einstellungen müssen bei jedem Systemstart erneut gesetzt werden. Weiterhin besitzt das Netcenter Fehler, für deren Umgehung einige Workarounds implementiert werden müssen. Die dafür nötigen Befehle und Parameter wurden in die Startup-Datei „rc.start“ eingetragen. Sowohl bei der modifizierten EPIAS-Firmware, als auch bei der im Rahmen dieser Thesis vom Autor erweiterten Version findet sich die Datei unter */opt/etc/rc.start*

Ein Bug sorgt dafür, dass das Netcenter bei jedem Neustart die Routing-Informationen verliert. Nach einem Neustart ist der Zugriff auf das Gerät daher ausschließlich aus dem lokalen Netz und nicht mehr aus dem Internet möglich. Die Routinginformationen müssen daher mit dem folgenden Befehl über die rc.start nach jedem Start gesetzt werden. Die IP-Adresse nach dem Stichwort „gw“ muss durch das Standardgateway des lokalen Netzes ersetzt werden:

```
route add -net 0.0.0.0 gw 10.10.10.10 netmask 0.0.0.0
```

Um das Netcenter aus dem Internet zu erreichen, müssen neben den Routinginformationen für das Standardgateway noch zwei weitere Bedingungen erfüllt sein:

1. Die öffentliche IP-Adresse des Internet-Service-Providers wird benötigt. Diese teilt man am Besten bei jeder Änderung einem dynamischen DNS Provider mit. Wie das mithilfe des Netcenters geht, wird im Kapitel [A.8](#) erläutert.

2. Für jeden Dienst, der auf dem Netcenter läuft und der aus dem Internet erreicht werden soll, muss eine Weiterleitung der dazu nötigen TCP- oder UDP-Ports im DSL-Router eingerichtet werden, sodass externe Anfragen an diese Dienste vom DSL-Router an die im internen Netz vergebene IP des Netcenters weitergereicht werden.

In der rc.start muss die Pfad-Variable auf die Binärverzeichnisse der Festplatte erweitert werden, gleiches gilt für den Bibliothekspfad. Im Anschluss wird die TZ Variable für die Zeitzone festgelegt.

```
export PATH=/sbin:/bin:/usr/sbin:/usr/bin:/opt/bin
export LD_LIBRARY_PATH=/lib:/usr/lib:/opt/lib
export TZ=CET-1CEST
```

Da die über das herstellereitige Webinterface getätigte Samba-Konfiguration nur wenige Einstellungen ermöglichte und zusätzlich an das Ende jeder erstellten Freigabe ein „Carriage-Return“ setzte, wurde auf die Samba-Konfiguration über das Webinterface gänzlich verzichtet und der interne Webserver beendet. Sollte dieses zu einem späteren Zeitpunkt zum Beispiel für ein Firmwareupdate wieder benötigt werden, kann es per Telnet nachträglich erneut gestartet werden, sofern kein neuer http-Dienst am gleichen Port lauscht. In der Standardeinstellung wurde der HTTP-Dienst vom init-Prozess überwacht und nach dem manuellen Beenden automatisch erneut gestartet. Um das zu verhindern, muss ein nvram-Parameter gesetzt werden. Um diesen auch nach einem Hardreset des Netcenters wieder zu setzen, wurde dieser ebenfalls in die rc.start eingetragen. Weiterhin wurde Samba mit eigener Konfiguration gestartet.

```
nvram set let_httpd_die=enabled
killall httpd
/usr/local/samba/sbin/smbd -D -s /opt/etc/smb.conf
```

Die Samba-Konfiguration sieht in verkürzter Form wie folgt aus:

Listing 2: Samba-Konfiguration

---

```
1 [global]
2     netbios name = Netzfestplatte
3     server string = "Netzfestplatte"
4     workgroup = Arbeitsgruppe
5     security = share
6     guest account = root
7     log file = /var/log/samba.log
8     socket options = TCP_NODELAY SO_RCVBUF=16384 SO_SNDBUF=8192
9     encrypt passwords = yes
10    use spnego = no
11    client use spnego = no
12    wins server =
13    wins support = no
14    host msdfs = no
15    interfaces = eth0
16    qos enable = no
17    level1 file extensions =
18    level2 file extensions =
19    os level = 20
20    preferred master = auto
21    domain master = auto
22    local master = yes
23    domain logons = no
24    log level = 0
25    max log size = 960
26    null passwords = yes
27    passdb backend = smbpasswd:/opt/etc/smbpasswd
28    use client driver = yes
29    printer admin = root, guest
30    show add printer wizard = no
31    load printers = no
32    default devmode = yes
33    printcap name = /tmp/etc/printcap
34
35 [share]
36    path = /shares/Main/share
37    writeable = yes
38    browsable = yes
39    inherit permissions = yes
40    inherit acls = yes
41    msdfs root = no
42    guest ok = yes
43    guest only = yes
```

---

Im Anschluss wurde die Disk-Spindown-Zeit auf 2 Stunden festgesetzt, das bedeutet: Die eingebaute Festplatte schaltet nach zwei Stunden der Inaktivität ab. Die eingestellte Zeit von zwei Stunden stellt dabei eine Optimierung dar, basierend auf den Faktoren

- möglichst langer Inaktivität
- möglichst seltenes Wiederanfahren

Ersteres spart Strom, letzteres schont das interne Festplattenlaufwerk. Das Festplattenlaufwerk wird bei Zugriffsversuchen automatisch angefahren. Dies geschieht auch, wenn der Arbeitsspeicher knapp wird und das Netcenter Swapping betreibt, also den Arbeitsspeicher auf die Festplatte auslagert. Daher wurde versucht, den von den Diensten permanent in Anspruch genommenen Arbeitsspeicher auf ein Minimum zu reduzieren. Nähere Informationen hierzu finden sich in Kapitel [A.1](#).

```
nvram set initial_disk_spin_down_setting=3600
```

Zum Schluss wurde der geänderte NVRAM Inhalt gespeichert:

```
nvram commit
```

Im EPIAS-Firmwareimage wurden wichtige Systemdateien aus dem schreibgeschützten `/etc/` -Ordner in den `/tmp/` -Ordner im RAM verlinkt. Hierzu gehören:

- `/etc/passwd`
- `/etc/shadow`
- `/etc/hosts`
- `/etc/resolv.conf`

Diese wurden via `cp` Befehl bei jedem Systemstart aus dem `/opt/etc/`-Verzeichnis in das `/tmp/`-Verzeichnis kopiert.

Nicht benötigte Dienste, unter anderem die Namensauflösung von Samba, wurden beendet, um Arbeitsspeicher zu sparen. Da die PID des Samba-Prozesses nach jedem Neustart variiert, wurde diese über den Prozessmonitor „ps“ ausgelesen und dem kill-Befehl als

Parameter übergeben.

```
ps | grep wdns | cut -d -f 3 | xargs kill  
ps | grep nmbd | cut -d -f 3 | xargs kill
```

Alternativ würde auch ein “killall” funktionieren, aus historischen Gründen wird noch auf den obigen Befehl zurückgegriffen. Zu guter Letzt wird bei jedem Bootvorgang via NTP (Network Time Protocol) die Uhr aktualisiert:

```
ntpclient -d -s -i 1 -h ptbtime1.ptb.de&
```

### 2.2.5 DHCP und DNS

Eine kurze Beschreibung der Funktionalität von DHCP findet sich im Glossar. Nähere Informationen finden sich in den zugehörigen RFCs für DHCP (2131, 2132) [25][29] und DNS (1034, 1035) [26][27]. Als DHCP und DNS-Server kommt auf dem Netcenter Dnsmasq zum Einsatz. Dieser beherrscht DHCP, DNS und TFTPd-Funktionalität. Letztere ist ebenfalls aktiviert und ermöglicht den Einsatz des Netcenters als PXE-Server.

Dnsmasq wurde via Paketquellen mit dem folgenden Befehl installiert:

```
ipkg-cl -f /opt/etc/ipkg.conf install dnsmasq
```

Der Aufruf über die rc.start geschieht mit dem Befehl “nice”, der dem Dienst eine höhere Priorität zuweist, um DHCP- und DNS-Anfragen mit größerer Priorität zu beantworten. Ein negativer Nice-Wert, in diesem Fall -10, steht für höhere Priorität.

```
nice -n -10 /usr/sbin/dnsmasq
```

Die Konfigurationsdatei von Dnsmasq befindet sich unter `/opt/etc/dnsmasq.conf`. Die Standardconfig von dnsmasq enthält wie bei vielen Unix-Tools üblich eine sehr lan-

ge Beispiel-Konfiguration mit Standardeinstellungen, die durch Entfernen der Auskommentierung am Zeilenanfang aktiviert werden. Der Übersicht halber wurden aus der Konfigurationsdatei alle Zeilen mit Kommentarzeichen am Zeilenanfang entfernt. Dies geschah mit dem folgenden Shell-Befehl, der sich reguläre Ausdrücke zunutze macht:

```
cat /opt/etc/dnsmasq.conf | grep "^[^#]"
```

Im Anschluss wurde die relevante Konfiguration durch eigene Kommentare ergänzt. Diese ist im Folgenden abgebildet:

---

Listing 3: DNSmasq-Konfiguration

---

```
1 #Interface, an dem DNSmasq tätig ist
2 interface=eth0
3
4 #Setzt Standardeinträge für PXE Unterstützung
5 dhcp-boot=/pxelinux.0,0.0.0.0
6
7 #Microsoft-Option: Client führt DHCP-Release beim Herunterfahren aus
8 dhcp-option=vendor:MSFT,2,1i
9
10 #Standard-DHCP-Range
11 dhcp-range=10.10.10.100,10.10.10.200,255.255.255.0,10.10.10.255,12h
12
13 #Setzen der Client MTU, um Fragmentierung über PPPOE und unnötige Path-MTU-
    Erkennung zu vermeiden
14 dhcp-option=26,1492
15
16 #Standardgateway
17 dhcp-option=3,10.10.10.10
18
19 #PXE-Serveradresse, 0.0.0.0 bedeutet "eigene IP"
20 dhcp-option=66,0.0.0.0
21
22 #DNS-Server, der dem Client mitgeteilt wird
23 dhcp-option=6,10.10.10.20,195.50.140.178
24
25 #Domäne
26 domain=meinedomaene.dyndns.org
27
28 #TFTP-Optionen
29 enable-tftp
30 tftp-root=/shares/Main/christian/tftpboot
31
32 #Datei mit Informationen über die aktuellen Clients
33 dhcp-leasefile=/var/log/dhcp-leases
```

---

Die Konfiguration definiert ein DHCP-Range und setzt die MTU auf 1492, um eine Fragmentierung von Paketen, die über die DSL-Leitung gesendet werden zu verhindern. Als DNS-Server wird das Netcenter sowie ein alternativer DNS-Server des Providers angegeben. Der PXE-Server wird auf die Netcenter-eigene IP gesetzt, weiterhin wird eine Lease-File definiert, in welcher die aktuell vergebenen DHCP-Adressen vermerkt werden. Nähere Informationen zu den verfügbaren DHCP-Optionen finden sich unter[\[30\]](#)

### 2.2.6 Lighttpd

Im ersten Schritt wurde Lighttpd installiert, dieser ist über die “bryan”-Quelle via ipkg verfügbar:

```
ipkg-cl -f /opt/etc/ipkg.conf install lighttpd
```

Im zweiten Schritt wurde die Konfiguration eingepflegt. Im ersten Absatz der Konfiguration wird das http-Root-Verzeichnis, der Serverport und die Datei mit enthaltener Process-ID angegeben. Der zweite Absatz gibt die Mime-Type-Zuweisungen an. Der dritte Absatz definiert den Namen der angezeigten Standard-HTML-Seite, in diesem Fall wurde nach der Konvention “index.html”, “index.htm” oder “index.php” genutzt. Weiterhin wurde das Verzeichnis-Listing aktiviert. Das bedeutet: Falls in einem Verzeichnis keine Index-Datei vorhanden ist, werden die Namen aller im Verzeichnis vorhandenen Dateien angezeigt. Dies stellt unter Umständen ein Sicherheitsrisiko dar. Der vierte Absatz definiert die Eigenschaften des PHP-Interpreters, auf den hier nicht näher eingegangen wird, da PHP für das Funktionieren von WPAD nicht benötigt wird. PHP wird für das Webinterface des installierten Bittorrent-Dämons genutzt, siehe Kapitel [A.10](#).

Listing 4: Lighttpd-Konfiguration

```
1 server.document-root = "/shares/Main/creon/wwwroot"
2 server.port = 80
3 server.pid-file = "/opt/bin/lighttpd/.pid"
4
5 # Mimetype mapping
6 mimetype.assign          = (
7   ".gif"                 =>   "image/gif",
8   ".jpg"                 =>   "image/jpeg",
9   ".png"                 =>   "image/png",
10  ".css"                 =>   "text/css",
11  ".html"                =>   "text/html",
12  ".js"                  =>   "text/javascript",
13  ".asc"                 =>   "text/plain",
14  ".txt"                 =>   "text/plain",
15  ".xml"                 =>   "text/xml" )
16
17 dir-listing.activate = "enable"
18 index-file.names = ( "index.php", "index.html", "index.htm" )
19
20 static-file.exclude-extensions = ( ".php" )
21 server.modules = ( "mod_fastcgi" )
22 fastcgi.server = (
23   ".php" => ((
24     "bin-path" => "/opt/bin/php",
25     "socket" => "/opt/bin/lighttpd/php.socket",
26     "max-procs" => 2,
27     "bin-environment" => (
28       "PHP_FCGI_CHILDREN" => "4",
29       "PHP_FCGI_MAX_REQUESTS" => "32"),
30     "bin-copy-environment" => ( "PATH", "SHELL", "USER"),
31     "broken-scriptfilename" => "enable"
32   ))
33 )
```

Eine anfängliche Fehlerquelle stellten die Mime-Typen dar: Durch fehlende Definition des CSS-Mime-Type wurde das eigens entworfene Webinterface ohne das gewünschte Layout angezeigt.

Gestartet wird Lighttpd über den folgenden Eintrag in der rc.start:

```
/opt/bin/lighttpd/sbin/lighttpd -f /opt/bin/lighttpd/lighttpd.conf &
```

### 2.2.7 Httpd-Performanceoptimierung

Lighttpd kam aufgrund der einfachen Installation und der vorhandenen Installationsbeschreibung unter[31] zum Einsatz. In einem späteren Optimierungsprozess in Bezug auf den Arbeitsspeicher wurde nach ressourcensparsameren Programmen Ausschau gehalten. Um nicht sämtliche verfügbaren http-Server manuell zu kompilieren und zu testen, wurde eine Suche in einschlägigen Suchmaschinen durchgeführt. Als Ergebnis kam heraus, dass es eine Vielzahl an ressourcensparsamen HTTP-Servern gibt, die größtenteils aus ihrer Testphase nicht erwachsen sind und teilweise keinen PHP-Support mitbringen. Auf diesen kann aufgrund diverser Steuerelemente des selbstprogrammierten Webinterface nicht verzichtet werden[32]. Zwei der am weitesten entwickelten Projekte sind thttpd und lighttpd. Das thttpd Projekt wurde im Jahre 2005 aufgegeben, sodass nur noch Lighttpd weiterentwickelt wird. Zwar ist nach Aussage der Quelle thttpd etwas genügsamer in Bezug auf den Arbeitsspeicher, Lighttpd ist dafür schneller, besser skalierbar und einfacher zu konfigurieren. Beide Dienste sind nicht (x)inetd-kompatibel. Da die Argumente mit Ausnahme des leicht höheren Speicherverbrauches für Lighttpd sprachen, wurde ein Umstieg auf thttpd nicht erwogen.[33][34][35]

## 2.3 Einführung in Proxy Auto Configuration

PAC steht für Proxy Auto Configuration. Es handelt sich um eine Textdatei, die die JavaScript-Funktion "FindProxyForURL(url, host)" definiert. Einmal dem Browser bekannt, durchläuft dieser die Funktion vor jedem Aufruf einer URL. Das bedeutet, es wird für jede im Browser eingegebene URL zur Laufzeit eine separate Entscheidung getroffen. Die Funktion liefert dem Browser als Rückgabewert üblicherweise entweder ein "DIRECT" zum direkten Aufruf der Seite oder ein "PROXY *url:port*" zum Aufruf der entsprechenden Seite über den zurückgelieferten Proxyserver. Die Javascript-Funktion überprüft dafür die übergebene URL nach bestimmten Kriterien, in der Regel via If-Konstrukt. Informationen zum Schreiben einer PAC-Datei finden sich in [18]

Eine dem Browser mitgeteilte PAC kann lokal gespeichert oder via HTTP bezogen werden. Der Name der Datei ist nicht relevant, üblich sind „proxy.pac“ oder „wpad.dat“. Im Folgenden findet sich das Beispiel für die auf dem Netcenter eingesetzte PAC-

Datei:

Listing 5: PAC-File des Netcenters

---

```
1 function FindProxyForURL(url, host)
2 {
3     url = url.toLowerCase();
4     if (
5         (url.substring(0, 5) == "http:") ||
6         (url.substring(0, 6) == "https:")
7     )
8         if (
9             isPlainHostName(host) ||
10            isInNet(host, "10.10.10.0", "255.255.255.0") ||
11            (host.substring(0, 4) == "127.")
12        )
13            return "DIRECT";
14        else
15            return "PROXY 10.10.10.20:8118; DIRECT";
16    else
17        return "DIRECT";
18 }
```

---

In diesem Beispiel wird die übergebene URL in Kleinbuchstaben umgewandelt und dann geprüft, ob die Ziel-URL eine HTTP oder HTTPS-Adresse ist. Andernfalls, zum Beispiel bei FTP-Adressen, wird DIRECT zurückgegeben. Handelt es sich um eines der oben genannten Protokolle, wird geprüft, ob es sich um eine Adresse im lokalen Netz oder um eine Localhost-Adresse handelt. Falls ja, wird ebenfalls DIRECT zurückgegeben, da davon ausgegangen wird, dass für lokale Adressen keine Verschleierung und Filterung nötig ist. In allen anderen Fällen wird als Proxy der Privoxy-http-Proxyserver auf dem Netcenter zurückgegeben: „PROXY 10.10.10.20:8118“, falls dieser nicht erreichbar ist, wird dieser mit DIRECT umgangen und eine direkte Verbindung hergestellt.

Dieses Vorgehen erhöht die Ausfallsicherheit des Internetzuganges für den Fall, dass der Privoxy- oder TOR-Dienst nicht verfügbar ist, führt auf der anderen Seite dazu, dass auf den ersten Blick nicht ersichtlich ist, ob ein anonymer Internetzugang gewährleistet wird. Das ist unter dem Gesichtspunkt der Anonymität ein fatales vorgehen. Es wurde in der Testkonfiguration in dieser Form implementiert, um die Verfügbarkeit des lokalen Netzwerkes nicht herabzusetzen.

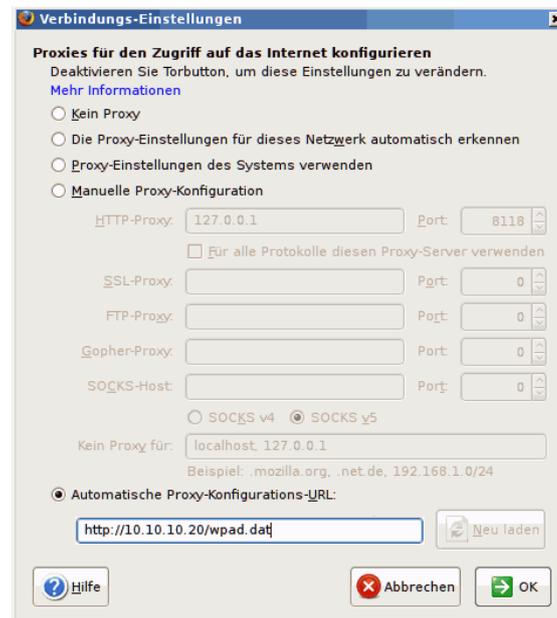


Abbildung 2: Eingabe der URL für die PAC-Datei beim Firefox.

## 2.4 Einführung in das Web Proxy Autodiscovery Protocol

Um den Netzwerkverkehr auch ohne Konfiguration einer PAC-Datei über den Privoxy/TOR-Dienst zu leiten und somit zu verschleiern, soll das Web Proxy Autodiscovery Protocol, kurz WPAD, eingesetzt werden. WPAD ermöglicht einem Internetbrowser, den zu nutzenden Proxyserver automatisch zu erkennen, indem es mittels bestimmter Vorgehensweisen im Netz nach einer PAC-Datei sucht.

Dafür macht sich der Internetbrowser entweder DHCP oder DNS zunutze. Wird DHCP genutzt, informiert der DHCP-Server den Browser via DHCP-Option 252 über eine URL, unter der dieser eine PAC-Datei mit beliebigem Namen findet. Diese DHCP-Option wird dem Client-Computer beim Rechnerstart mitgeteilt, alternativ kann der Internetbrowser diese per DHCP-Inform erfragen.

Als Alternative zu DHCP kann der Internetbrowser eine DNS-Anfrage an die „wpad“-Subdomain der eigenen Domäne stellen, in diesem Falle kann der Pfad der PAC-Datei nicht frei gewählt werden, die PAC-Datei liegt dann als wpad.dat im Wurzelverzeichnis. Es gibt weitere Möglichkeiten der Abfrage der PAC-Datei,

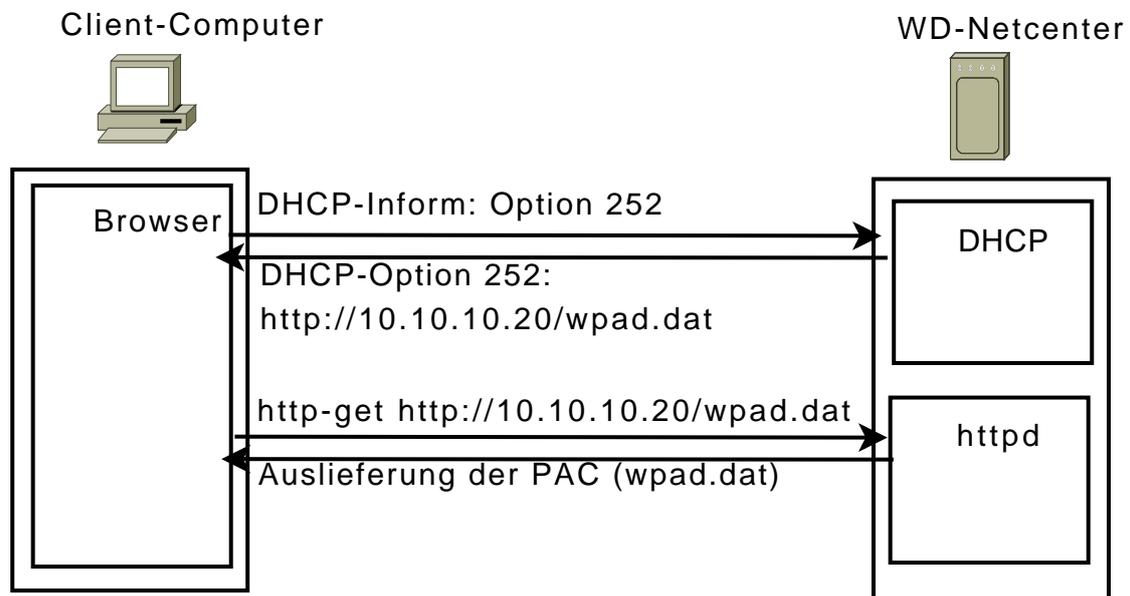


Abbildung 3: Abfrage der PAC via DHCP

sie finden sich im Draft des WPAD-Standards [5]. Diese werden hier nicht näher erläutert.

Obwohl die Adressabfrage für die “wpad.dat”-Datei via DHCP aufgrund der Vorteile in Bezug auf die Flexibilität gerne verwendet wird, unterstützt der Internetbrowser Firefox diese Abfrageform nicht. Das Verhalten ist in [14] beschrieben. Daher wurde DNS als Erkennungsmethode konfiguriert, indem die PAC-Datei als wpad.dat im lighttpd-Rootverzeichnis abgelegt wurde. In der /etc/hosts des Netcenters - in diesem Fall ein Softlink nach /opt/etc/hosts - wurde die Zuordnung “wpad.domain.local” zur IP “10.10.10.20”, der IP des Netcenters, eingetragen. Da das Netcenter sich selbst als DNS-Server via DHCP mitteilt, wie im Kapitel 2.2.5 beschrieben, ist das korrekte Auflösen dieser Adresse im Netz gewährleistet, solange das Netcenter netztechnisch erreichbar ist. Ist das Netcenter ausgeschaltet, sind der Lighttpd, TOR und Privoxy ebenfalls nicht erreichbar.

Die Browsereinstellung für den Internet Explorer sowie den Firefox werden unabhängig vom genutzten Betriebssystem wie in der Abbildung beschrieben gesetzt.

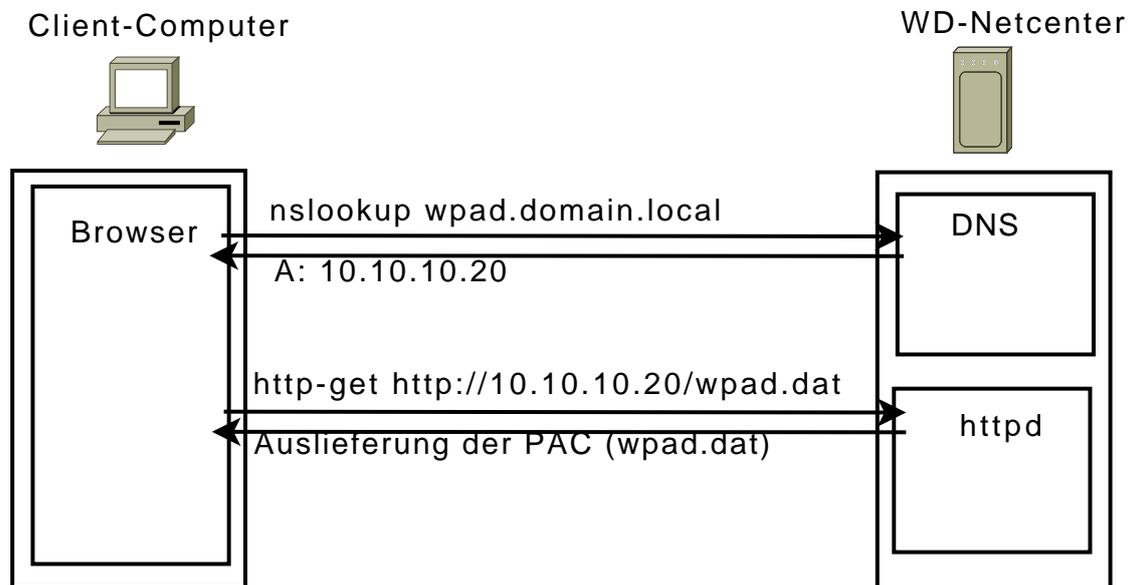


Abbildung 4: Abfrage der PAC via DNS

Weitere auf dem Netcenter installierten Anwendungen und die Optimierung des Netcenters in Bezug auf den Speicherverbrauch mittels Xinetd sind im Anhang A beschrieben.

## 2.5 Alternative Lösungsmöglichkeit: Zwangsproxy

Statt den Proxyserver über WPAD erreichbar zu machen, wäre auch der Einsatz eines Zwangsproxys eine mögliche Lösungsform. Im Gegensatz zum WPAD ist die Nutzung des Zwangsproxy obligatorisch, sodass ein versehentliches, nicht-anonymes Surfen unterbunden wird. Auf der Gegenseite bedeutet die Einrichtung eines Zwangsproxys höheren Aufwand, wobei zwischen zwei möglichen Verfahren zu unterscheiden ist. Beiden gleich ist die Idee, das Netcenter physikalisch zwischen das Heimnetzwerk und den DSL-Router zu positionieren und den eintreffenden Netzwerkverkehr über den TOR-Proxy zwangszuleiten, ohne eine Umgehung direkt in das Internet zu ermöglichen. Ebenfalls bedeuten beide Verfahren für das Netcenter eine höhere Auslastung, da sämtliche ausgehenden Pakete über das Netcenter geleitet werden.

Im Promiscuous-Mode betrieben, wäre das Netcenter IP-technisch unsichtbar. Es besäße in diesem Modus keine IP-Adresse und wäre daher nicht mehr als Datenträger im Netz

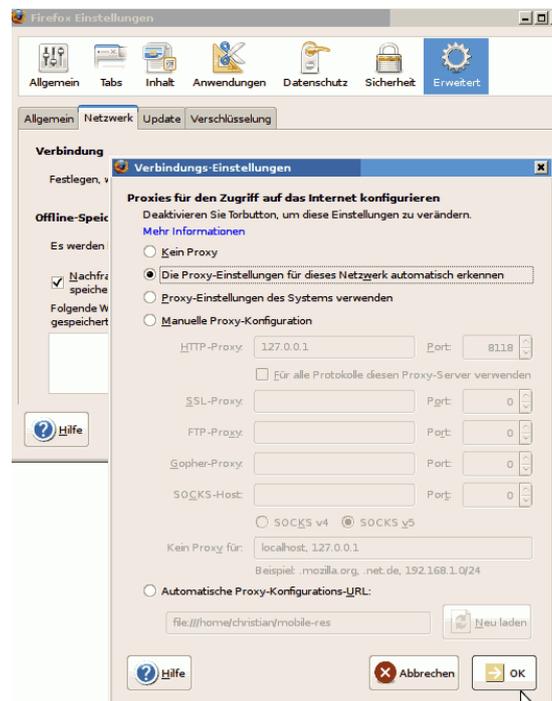


Abbildung 5: Einstellung des Firefox für die Nutzung von WPAD

verfügbar. Um diesen Modus verwenden zu können, benötigt das Netcenter zwei Netzwerkkarten. Daten, die an der einen Netzwerkkarte aus dem LAN eintreffen, werden über die zweite Netzwerkkarte zum DSL-Router weitergeleitet - und umgekehrt. Aufgrund nicht ausreichender Netzwerkschnittstellen ist dieses Verfahren in der Form nicht implementierbar.

Die zweite Möglichkeit ist die Nutzung als Router. Auch hier benötigt das Gerät zwei Netzwerkkarten oder die Unterstützung für virtuelle LANs (VLANs). Im zweiten Fall müsste der verbundene LAN-Switch ebenfalls VLAN-Unterstützung mitbringen. Da das nicht der Fall ist, wäre eine Neuinvestition in Hardware nötig. In diesem Modus fungiert das Netcenter als Standardgateway im Client-Netz und routet den Traffic zum DSL-Router in einem zweites LAN, sodass der DSL-Router physikalisch von den Clients getrennt wäre.

Der deutlich höhere Aufwand, auch auf finanzieller Seite, rechtfertigt die Implementierung in den Augen des Autors nicht. Die aktuelle Gesetzeslage in Deutschland rechtfertigt

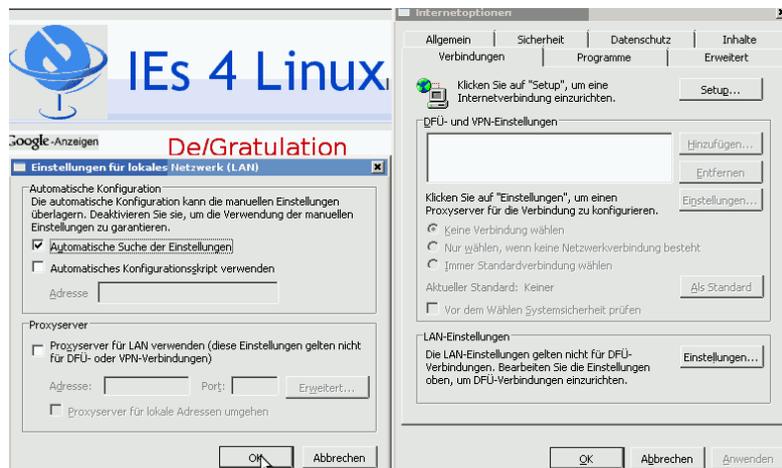


Abbildung 6: Einstellung des IE für die Nutzung von WPAD

bei der weit überwiegenden Anzahl an Webseitenbesuchen den Einsatz eines Anonymisierers nicht. Wenn als Resultat der Nutzung der Zugang um mehrere Größenordnungen langsamer vonstatten geht, ist Anonymisierung eine Funktion, die man Nutzen können sollte; eine Verpflichtung zur Nutzung führt den Freiheitsbegriff allerdings ad absurdum.

### 3 TOR und Privoxy, Cross-compiling

#### 3.1 Funktionsweise und Schwächen von TOR

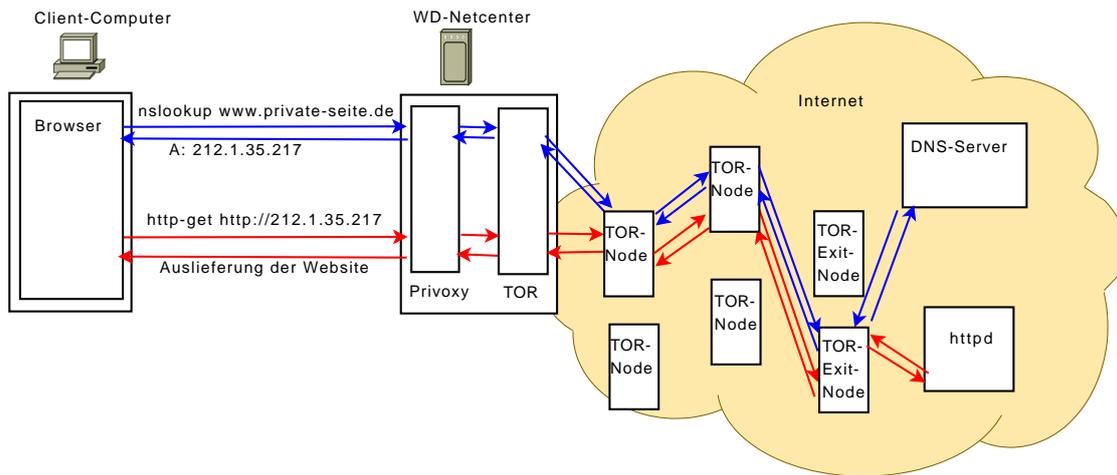


Abbildung 7: Namensauflösung über TOR als Socks4a-Proxy.

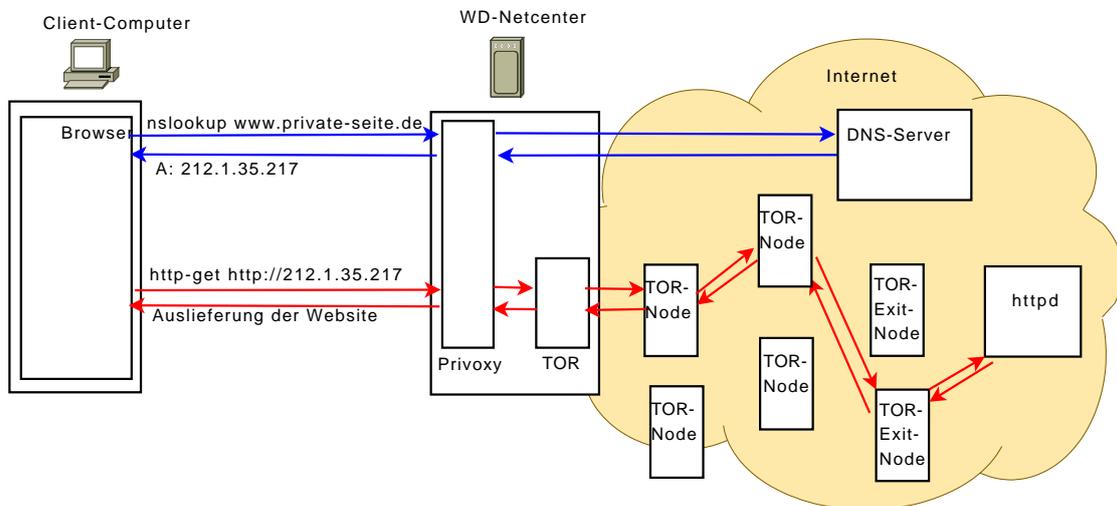


Abbildung 8: Namensauflösung mit Socks4 Proxy, Datentransfer über TOR.

TOR steht für "The Onion Router", der Name rührt von dem Zwiebelschalen-Prinzip der Anonymisierung. Daten werden vom Client erst mehrfach mit verschiedenen Schlüsseln verschlüsselt und an eine Kette von Onionroutern (Tor-Nodes) weitergesandt, von denen jeder Onionrouter nur seine jeweiligen Nachbarn kennt und jeder nur in der Lage ist, die jeweils äußere Verschlüsselung zu entschlüsseln. Der Einstiegspunkt für Clients ist

der Onionproxy. Dieser wird auf dem Netcenter installiert, um anonymen Zugang ins Internet zu gewähren. Das TOR Netzwerk besteht neben den Onionproxys aus den schon eingangs erwähnten Onionroutern. Zwei spezielle Typen von Onionroutern sind der Entryguard und die Exitnode. Um den Netzwerkverkehr zu anonymisieren, verbindet sich der Onionproxy mit einem Entryguard und handelt einen symmetrischen Schlüssel über ein Schlüsselaustauschverfahren aus. Im Anschluss handelt er mit einem beliebigen weiteren Onionrouter ebenfalls einen symmetrischen Schlüssel aus, wobei die Verbindung zu dem zweiten Onionrouter über den ersten Onionrouter (den Entryguard) hergestellt wird. Die Daten werden dabei verschlüsselt an den ersten Router geschickt, dieser leitet sie unverschlüsselt an den zweiten.

Nun wählt der Onionproxy eine beliebige Exitnode und handelt mit dieser ebenso einen Schlüssel aus. Die Verbindung wird dabei erst mit dem Schlüssel des zweiten Routers und im Anschluss mit dem Schlüssel des ersten Routers verschlüsselt und nun an den ersten Router geschickt. Dieser entschlüsselt die äußere Verschlüsselung und schickt sie an den zweiten Router, dieser die entschlüsselten Daten an den dritten, die Exitnode.

Nach erfolgreichem Aufbau dieser Router-Kette werden alle für das Internet bestimmten Daten nun erst mit dem Schlüssel der Exitnode, dann mit dem Schlüssel des "mittleren" Routers und im Anschluss mit dem Schlüssel des Entryguards verschlüsselt und an diesen geschickt. Der Entryguard entschlüsselt die äußere Verschlüsselung und findet dort Informationen über den nächsten Router. Die Entschlüsselung wird in dieser Methode fortgesetzt bis zur Exitnode. Diese entschlüsselt die Daten gänzlich, sofern nicht verschlüsselte Protokolle zwischen Quelle und Ziel eingesetzt werden, und sendet sie von dort aus zum Empfänger. Da jeder Onionrouter nur die Nachbarrouter kennt, ist die Quelle selbst dann nicht mit dem Ziel in Verbindung zu bringen, wenn einer der Onionrouter kompromittiert wurde. Diese Kette kann für sämtlichen Netzwerkverkehr unabhängig vom Protokoll verwendet werden. Zur Erhöhung der Sicherheit wird nach ca. 10 min eine neue Kette an TOR-Nodes aufgebaut.

Diese Kette besteht aus mindestens drei Onionroutern, über die vom Onionproxy alle Daten, die für das Internet bestimmt sind, verschickt werden. Daten, die vom Empfänger zurückgesendet werden, laufen die Kette der Router in umgekehrter Reihenfolge ab. Informationen über verfügbare TOR-Nodes, die zum Aufbau der Kette zur Auswahl stehen, werden über Verzeichnisserver erfragt. Diese befinden sich hartkodiert

in TOR[54].

TOR ist für Anwendungen als SOCKS-Proxy erreichbar. Alle Anwendungen, die SOCKS-Proxys unterstützen, können TOR verwenden. Unterstützt eine Anwendung keinen SOCKS-Proxy, ist es möglich, Anwendungen mit dem Befehl “torify” aufzurufen. Torify ist ein Wrapper, der den Netzwerkstack der Anwendung abfängt und durch TOR leitet. Die Nutzung von Torify hat zur Bedingung, dass die Anwendung auf dem gleichen Gerät läuft wie TOR. Torify ist auf dem Netcenter nur dann nutzbar, wenn auch der Dienst, der torify in Anspruch nimmt, ebenfalls auf dem Netcenter läuft. Soll der Dienst auf einem der Clients laufen, muss TOR separat auf dem Client installiert werden. Das ist möglich, allerdings nicht Bestandteil dieser Arbeit.

Mit der oben vorgestellten Methode ist die Ziel-Adresse bekannt, lediglich die Quelle wird anonymisiert. Um auch das Ziel anonym zu halten, beispielsweise um sensible Informationen anzubieten, bietet TOR die Möglichkeit “versteckter Services”. Diese sind über eine ausschließlich im TOR-Netzwerk verfügbare top-level-Domain mit der Endung “.tor” erreichbar. Die Namen müssen zum Erreichen dieser Domain über das TOR-Netzwerk aufgelöst werden. Auf die Funktionsweise der versteckten Services wird in dieser Arbeit nicht näher eingegangen, nähere Informationen findet man in [54].

Bei dem Startvorgang von TOR meldet sich dieses mit der Meldung: “This is experimental software. Do not rely on it for strong anonymity.” Roger Dingledine, einer der Gründer und Hauptentwickler von TOR sagte hierzu in einem Vortrag des 25. Chaos Communication Congress in Berlin, dass zwar TOR noch lange nicht fertig und perfekt sei, es aber auf der anderen Seite keine Software gibt, die es besser kann.[55]

Einige der Schwächen werden in diesem Kapitel aufgezählt. Das Wissen um sie ist wichtig, um sich den möglichen Gefahren im Umgang mit TOR bewusst zu werden und damit umzugehen, anstatt sich auf volle Anonymität zu verlassen.

**Unverschlüsselte Übertragung** Eine der bekanntesten Angriffsmethoden ist es, eine Exitnode zu betreiben und den ausgehenden Verkehr nach persönlichen Informationen zu durchsuchen. TOR verschlüsselt, bedingt durch das Funktionsprinzip, den Datenverkehr zwischen dem Onionproxy und der Exidnode. Von dort bis zum Ziel wird unverschlüsselt übertragen. Daher ist es notwendig, nur verschlüsselte Protokolle über TOR zu übertragen. Unverschlüsselte Protokolle wie http, smtp oder ftp lassen Rückschlüsse auf den Urheber zu und beinhalten unter Umständen Klartext-Zugangsdaten und -passwörter.

**Namensauflösung** Neben dem Transfer der Nutzdaten über TOR muss auch die Namensauflösung verschleiert werden. Eine nicht verschleierte Namensauflösung verrät jene Zielrechner, die nicht direkt über ihre IP-Adresse angesprochen wurden. Während im Browser hinterlegte HTTP-Server wie Privoxy die Namensauflösung automatisch übernehmen, muss bei deren Konfiguration in Bezug auf die Weiterleitung an TOR darauf geachtet werden, TOR als Socks4a-Proxyserver anzusprechen. Socks4a-Proxyserver übernehmen die Namensauflösung grundsätzlich. Socks4-Proxyserver sind hingegen nicht in der Lage die Namensauflösung zu übernehmen, der HTTP-Proxy löst diese also über das nicht anonymisierte Internet auf. Socks5 unterstützt die Namensauflösung optional. Ebenso muss bei allen Anwendungen, die direkt mit TOR als SOCKS-Proxy kommunizieren, die Version auf Socks4a oder Socks5 gesetzt werden. Dazu gehören Tauschbörsen, Telnet- oder SSH-Clients und Instant-Messenger, sofern sie nicht über einen HTTP-Proxy kommunizieren.

**Angriffe auf die Quelle** In Ländern wie China werden “unerwünschte” Verbindungen durch die große chinesische Firewall mittels TCP-Reset zurückgesetzt. Ebenfalls wurden SSL-Man-in-the-Middle-Attacken in China beobachtet, die von den chinesischen ISPs gefahren wurden. Wird eine Exitnode in einem solchen Gebiet genutzt, werden Angriffe durch das TOR-Netz an den Client geschleust. Hierbei handelt es sich nicht um einen Angriff auf das TOR-Netz selbst. Unter Umständen birgt der Einsatz von TOR jedoch ein höheres Gefahrenpotential, von Angriffen dieser Art betroffen zu sein.<sup>[55]</sup>

**geschwätzige Plugins** Ist der nicht-anonymisierte Internetzugang aus technischer Sicht nicht gesperrt, sondern nur die Anwendungen auf die Nutzung von TOR eingerichtet, so besteht das Risiko, dass einige Anwendungen entgegen dieser Konfiguration direkt mit dem Internet kommunizieren. Das ist unter anderem dann der Fall, wenn TOR auf dem gleichen Rechner läuft wie die Internetapplikation oder neben dem Proxy noch eine direkte Internetverbindung besteht, wie im Falle des Netcenters. Applikationen können dann durch Softwarefehler, fehlkonfigurierte Plugins oder fehlende Konfiguration direkt auf das Internet zugreifen, ohne von einer Verschleierung gebrauch zu machen. Davon betroffen sind beispielsweise Antivirusprogramme, die Anwendungsaktualisierungen durchführen und den im System eingetragenen Proxy nicht verwenden, oder Plugins für den Firefox oder Internetexplorer. Bei letzterem sind bekannte Beispiele das Adobe-Reader- und das Flash-Plugin. Werden über diese Plugins Daten aus dem Netz geladen, indem man ein PDF betrachtet oder eine Flash Animation mit dem Netz kommuniziert, werden die Daten nicht über TOR bezogen und somit die tatsächliche Identität preisgegeben. Daher wird empfohlen, sämtliche Plugins zu deaktivieren. TOR-Button, ein Firefoxplugin, sorgt bei Nutzung für die entsprechende Browserkonfiguration. Weil TOR-Button über Javascript abschaltbar ist, wird in neueren Versionen auch Java-Script von TOR-Button deaktiviert. Die Einrichtung von TOR-Button wird in Kapitel 3.10 beschrieben.

**statistische Angriffe** Gelingt es, Netzwerkverkehr vom Entryguard als auch von der Exitnode aufzuzeichnen, so kann man über statistische Analysen gegebenenfalls Information über den Sender herausfinden. Zielgerichtete Angriffe sind mit dieser Methode schwieriger, da Entryguard und Exitnode für den Onionproxy frei wählbar sind.

**kontrolliertes Netz** Ein hypothetischer Angriff basiert darauf, einen großen Teil des Netzes zu kontrollieren. Jemand, der einen Großteil des Netzes kontrolliert und überwachen kann, kann damit die oben erwähnten statistischen Angriffe auf TOR fahren. Wer selbst viel Netzwerkverkehr über TOR-Nodes verschicken kann, kann über statistische Analysen fremden Verkehr heraus filtern. Insbesondere mit Botnets kann man viele Rechner kontrollieren und sich eine große Anzahl von Exitnodes aufbauen, sowie viel Netzwerkverkehr erzeugen. Hier gab es verschiedene Angriffsszenarien zur Kontrolle von (Exit)-Nodes. Einige konnten durch Protokolländerungen in TOR unterbunden oder zumindest entschärft werden. Insbe-

sondere werden Entryguard und Exitnodes nicht mehr im gleichen IP-Bereich gewählt. Die maximale Kettenlänge an hintereinandergeschalteten TOR-Nodes wurde auf acht begrenzt, um zu verhindern, dass ein manipulierter Client durch eine überlange Router-Kette das gesamte TOR-Netz auslasten kann.[55]

**hartkodierte Verzeichnisse** Eine der Schwächen sind die eingangs erwähnten, hartkodierte Verzeichnisse. Auf der einen Seite ist bei einem komprimierten Internetzugang möglich, Anfragen an diese umzuleiten, was allerdings nur in Zusammenhang mit dem Fälschen der Zertifikate erfolgreich funktioniert. Auf der anderen Seite ist es möglich, diese Verzeichnisse über eine Firewall zu sperren und somit den Zugriff auf das TOR-Netzwerk zu verhindern, um den Nutzer zur unverschleierte Kommunikation zu zwingen.[54]

Im nächsten Kapitel folgt eine Beschreibung des Internet-Filterproxy Privoxy, der dem TOR-Proxy als HTTP-Proxy vorgeschaltet wird.

### 3.2 Funktionsweise von Privoxy

Eine häufig anzutreffende Kombination von Anonymisierungsdiensten besteht aus dem für diese Bachelorthesis ausgewähltem TOR-Dienst und Privoxy. Privoxy ist ein Proxyserver aufbauend auf dem "Internet Junkbuster", dessen Filter auf regulären Ausdrücken basiert. Er wird dazu genutzt, Werbung und (unter Umständen schädliche) Skripte aus Internetseiten herauszufiltern. Das Zwischenspeichern von Internetseiten, sogenanntes Cachen, unterstützt Privoxy nicht. Für die Kombination von Privoxy und TOR gibt es mehrere Gründe:

**Vermittlung** Privoxy ist als http-Proxy der Vermittler zwischen dem Internetbrowser und dem SOCKS-Proxy TOR. Alternativ können andere http-Proxy-Server wie Polipo [56] verwendet werden. Beide erfüllen die Aufgabe, den http-Datenstrom für den SOCKS-Proxy TOR zu übersetzen. Polipo besitzt Pipelining- und Caching-Funktion, dafür fehlen ihm sämtliche Filtereigenschaften.

**Sicherheit** Privoxy bringt Filterfunktionalität für Werbung, Skripte und Cookies mit, auf die im Abschnitt 3.7 genauer eingegangen wird. Dadurch wird der ungewollte Versand von persönlichen Daten verhindert. Zusammen mit einer solchen Schutzvorkehrung ist der Einsatz eines Anonymisierungsdienstes sicherer.

**Geschwindigkeit** Die für ein TOR-Netz zur Verfügung stehende Internetbandbreite stellt ein kostbares Gut dar. Daraus ergibt sich die Handlungsmaxime für jeden Nutzer, dieses Netz effizient zu nutzen. Privoxy leistet hier seinen Beitrag, indem Werbung herausgefiltert wird. Weiterhin nimmt die Geschwindigkeit des Seitenaufbaus über TOR ab, da die Datenpakete mehrere TOR-Nodes als Zwischenstationen passieren, bis sie das jeweilige Ziel erreichen. Die dadurch entstehende hohe Latenz ist umso größer, je höher die Anzahl der Server ist, von denen geladen wird. Verzögert sich die Verbindung zu einem Webserver, so hat dies Auswirkungen auf den Aufbau der gesamten Webseite. Da die Werbung üblicherweise von anderen Servern geladen wird als der Inhalt, stellt das Filtern eine sehr effektive und effiziente Methode dar, die Anzahl der Verbindungen zu begrenzen und die Latenzzeit beim Seitenaufbau herabzusenken.

Bilder und andere interaktive Inhalte, beispielsweise Flash-Animationen, werden von Privoxy gefiltert, wenn sie bestimmte Kriterien erfüllen. Dazu gehört neben der Bildgeometrie auch die Domain, von der sie geladen werden. Liegt die Graphik auf einer anderen Domain, einer Subdomain oder einem Unterordner mit Begriffen wie „ad“<sup>3</sup> im Namen und entspricht die Bildgeometrie typischen Bannergrößen, wird das Bild nicht geladen. Weiterhin ist Privoxy in der Lage, Cookies webseitenbasiert zu erlauben respektive zu verbieten. Neben eingebauten Standardfiltern in verschiedenen starken Filterstufen können zusätzlich oder alternativ eigene Filter genutzt werden.

Konfiguriert werden die Filter über ein eigenes Webinterface. Über die Konfigurationsdateien kann ebenfalls bestimmt werden, ob bestimmte Einstellungen über das Webinterface steuerbar sind, beispielsweise Privoxy's Filterfunktionen komplett zu deaktivieren. Die grundlegenden Konfigurationen werden im nachfolgenden Abschnitt beschrieben, für erweiterte Einstellungen gibt es sehr detaillierte Quellen wie [57].

---

<sup>3</sup>Abkürzung des englischen Wort „Advertising“ für Marketing, Werbung

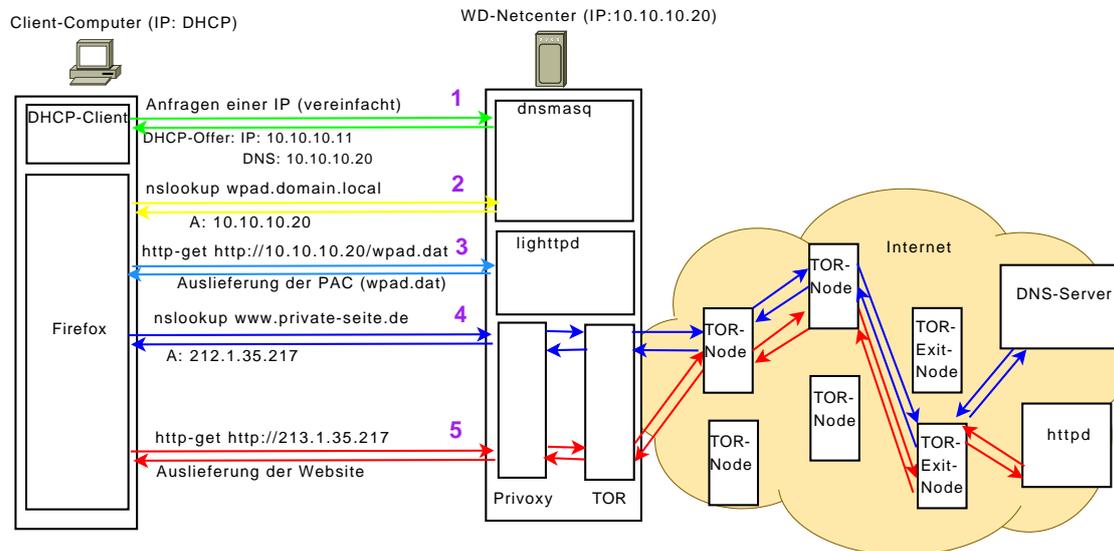


Abbildung 9: Zusammenspiel der Kommunikation aller Dienste auf dem Netcenter

1. Der DHCP-Client auf dem Client-Computer fragt per Broadcast nach einem DHCP-Server. Der Dnsmasq-Dienst auf dem Netcenter antwortet als einziger Server im Netz und vergibt die IP-Adresse, Subnetmask, Standardgateway, den DNS-Server, die Domäne (nicht in der Skizze, domain.local) und weitere per DHCP-Option festgelegte Parameter.
2. Der Browser auf dem Client-Computer fragt den per DHCP zugewiesenen DNS-Server nach der IP-Adresse des Rechners mit dem Namen wpad gefolgt von der eigenen Domäne (beispielsweise wpad.domain.local). Dnsmasq gibt hier die eigene IP-Adresse, also die IP-Adresse des Netcenters zurück.
3. Der Browser fordert per http-get-Request die Datei wpad.dat (PAC-Datei) im Root-Verzeichnis dieses Rechners an. Der lighttpd-Server des Netcenters liefert die entsprechende Datei aus.
4. Der Browser setzt nach Auswertung der PAC-Datei den Privoxyserver des Netcenters als http-Proxy fest. Über Privoxy wird die IP-Adresse der Domäne www.private-seite.de angefragt, Privoxy löst aufgrund der SOCKS4a-Konfiguration den Namen über das TOR-Netzwerk auf und liefert ihn zurück.
5. Der Browser fordert per http-get-Request die Startseite vom http-Dienst der zurückgelieferten IP-Adresse.

Um die hier vorgestellten Dienste auf dem Netcenter ausführen zu können, müssen diese zuerst für die vorliegende MIPSel-Architektur kompiliert werden.

### 3.3 Natives vs. Cross-Compiling

Das Kompilieren für die MIPSel-Architektur ist auf zwei Arten realisierbar:

**Nativ** Bei nativem Compiling läuft die Kompilationsumgebung auf der gleichen Architektur wie das resultierende Kompilat. Ein fertig kompiliertes Programm kann also auf der gleichen Maschine ausgeführt werden.

**Cross-Compiling** Beim Cross-Compiling wird die Compiling-Software auf einer anderen Architektur ausgeführt als das resultierende Kompilat. Eine kompilierte Anwendung kann auf der gleichen Maschine nicht ausgeführt werden, weil der ausführende Prozessor die Maschinenbefehle im resultierenden Programmcode nicht interpretieren kann.

Es gibt zwei Fälle, in denen Cross-Compiling hauptsächlich zum Einsatz kommt: Der erste Fall ist, wenn noch keine native Software existiert. Wird eine neue Architektur konzipiert, muss der erste Compiler für diese Architektur via Cross-Compiling kompiliert werden, möchte man ihn nicht in Assembler programmieren. Der zweite Fall tritt dann ein, wenn das Zielsystem nicht leistungsfähig genug für einen nativen Compiler ist. Embedded-Devices wie programmierbare Taschenrechner oder Steuerprozessoren fallen in diese Kategorie. Ein grenzwertiges Beispiel ist das Netcenter. Einen nativen Compiler auf das Netcenter zu portieren ist möglich. Die Zeitdauer für das Kompilieren einer Anwendung wie TOR liegt allerdings aufgrund des knappen Hauptspeichers in der Größenordnung von einem Tag bis zu einer Woche. Auf einem Desktop-Rechner läuft dieser Vorgang innerhalb von Minuten ab. Auf der anderen Seite führt natives Kompilieren das Problem ad absurdum, denn der native Compiler müsste erst via Cross-Compiling erstellt werden.

### 3.4 Toolchain und Compiling

Zum Durchführen des Cross-Compilings wird eine Toolchain benötigt. Bei einer Toolchain handelt es sich um einen Compiler mitsamt vorkompilierter Bibliotheken, Header-Dateien<sup>4</sup> und zusätzlicher Werkzeuge, die für das Portieren von Anwendungen auf das Embedded Device nützlich sind. Auf einige wird im Laufe des Kapitels eingegangen.

Toolchains werden teilweise von den Herstellern der Embedded-Devices zur Verfügung gestellt. Das liegt unter anderem an der GPL-Lizenzbestimmung, die die Hersteller verpflichtet, ihre Softwarequellen zu veröffentlichen, wenn sie selbst GPL-Software als Grundlage für ihre Entwicklung verwendet haben.

Es stellte sich heraus, dass die Anzahl der im Internet kursierender Toolchains für MIPSel-Geräte im einstelligen Bereich lag. Geräte, die MIPSel-Prozessoren verwenden, beschränken sich auf wenige, dafür sehr bekannte Geräte: Dazu gehören:

- Fritzbox
- Linksys WRT-Router
- Maxtor-Shared-Storage

Die folgenden Toolchains wurden als potentielle Kandidaten ausgewählt:

- hnd\_toolchain\_3.0
- hnd\_toolchain\_3.2.3
- uclibc-mipsel-4.2.1
- Freetz\_1.01

Die hnd-toolchain wird von Linksys, Maxtor und Western-Digital für die oben aufgeführten Geräte zur Verfügung gestellt. Diese ist in zwei verschiedenen Versionen verfügbar. Die Version 3.0 bzw. 3.2.3 bezieht sich dabei auf die zugrundeliegende Versi-

---

<sup>4</sup>Bei Header-Dateien handelt es sich um C-Quellcode, der Funktionen bereitstellt, die über Bibliotheken eingebunden werden. Header Dateien werden über "include"-Anweisungen vom Präprozessor des Compilers eingebunden.

on der GNU-C-Compilers. Version 3.0 ist die von Western Digital für das Netcenter zur Verfügung gestellte Version. Version 3.2.3. wird von Maxtor als auch von Linksys zur Verfügung gestellt. Die von Linksys zur Verfügung gestellte Version ist umfangreicher und enthält neben den Sourcen des Linksys-Kernels ebenfalls nützliche Werkzeuge, die das Zusammenbauen der Geräte-Firmware aus ihren Bestandteilen ermöglichen. Diese können zum Verändern der bestehenden Firmware genutzt werden, indem die bestehende Firmware zerlegt und angepasst wird. Zu den Tools gehört zum einen “mkcramfs”, ein Werkzeug zum Erstellen eines komprimierten ROM-Filesystems, wie es auf dem Netcenter zum Einsatz kommt. Zum anderen ist “trx” inkludiert, welches die Firmware aus einem Header, dem Kernel und dem CramFS zusammensetzt. Der Einsatz dieser Programme wird im Anhang [A.2](#) beschrieben.

Die Entwickler der uclibc-Bibliothek stellen ebenfalls einen Cross-Compiler zur Verfügung. Diesen gibt es neben der Sourcecode-Version auch als fertig kompiliertes Binary zum Download. Die Anzahl der von diesem Compiler mitgelieferten Bibliotheken ist begrenzt, aufgrund der Bibliothekenvielfalt wurde daher der Freetz-Compiler bevorzugt. Allerdings ist der uclibc-Compiler der einzige, mit dem sich die Busybox fehlerfrei übersetzen lässt. Nähere Informationen zu der uclibc-Bibliothek findet sich im Abschnitt [3.6](#). Die Toolchain gibt es unter [\[51\]](#) zum Download.

Die Freetz Toolchain ist eine freie Toolchain, die in der aktuellen Version 1.0.1 auf dem GCC 4.2.1 aufsetzt. Das Paket von Freetz ist umfangreicher, aber auch komplexer in der Anwendung. Freetz ist ein Wortspiel aus “Fritz” und “free”. Die Freetz-Toolchain wurde zum Kompilieren freier Software, respektive zum Entwickeln einer freien Firmware für die Fritzbox erstellt.

Um ein besseres Verständnis für die nachfolgenden Unterkapitel zu gewährleisten, wird in diesem Kapitel erläutert, wie ein Kompilervorgang abläuft. Nur so sind die auftretenden Fehlermeldungen sinnvoll für den Anwender interpretierbar.

Compiling bedeutet im ersten Schritt, dass ein im Quellcode vorliegendes Programm vom Compiler gelesen und interpretiert wird und in einem für den Prozessor lesbaren Format ausgegeben wird. Als Beispiel für ein Programm, bei dem der Kompilervorgang sehr einfach vonstatten geht wird hier das später vorgestellte “prime.c”-Programm gewählt. Es gibt die Primfaktorzerlegung einer eingegebenen Zahl aus und entstand im Rahmen des

Studiums. Das Programm besteht lediglich aus einer Quellcode-Datei, dementsprechend muss der Compiler nur einmal aufgerufen werden.

Listing 6: Prime: Programm zur Primfaktorzerlegung

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <math.h>
4 int main()
5 {
6     unsigned int iEingabe = 1, iIntsize, i, j, k, iModIntsize;
7     iIntsize=sizeof(int)*8;
8     iModIntsize = iIntsize-1;
9
10    while (0 < iEingabe)           //Berechnungsschleife solange bis 0 eingegeben
        wird
11    {
12        printf ("Bitte geben sie eine Zahl ein, die sie faktorisieren moechten (exit =
            0):\n");
13        scanf ("%i",&iEingabe);
14        printf ("Primfaktorzerlegung: 1 ");
15
16        k = iEingabe/iIntsize+1;    //Siebgröße bestimmen
17        unsigned int sieb[k];
18        for (j=0; j<k; sieb[j++]=0); //Sieb löschen
19        k=0; i=2;                   //Laufvariablen initialisieren
20        while (1 < iEingabe)       //Abbruchbedingung Zahl = 1
21        {
22            if (0 == (sieb[i/iIntsize]&1<<(i&iModIntsize))) //Laufvariable = Primzahl?
23            {
24                while (0 == iEingabe%i) //Wenn Eingabe durch Laufvariable teilbar...
25                {
26                    iEingabe/=i;      //...dann so oft teilen wie möglich, Anzahl der
                        Teilungen merken
27                    k++;
28                }
29                if (0 < k)           //Anzahl der Teilungen > 0? => Ausgabe des Teilers
                    , Anzahl Teilungen
30                {
31                    printf("* %i~%i ",i,k);
32                    k=0;
33                }
34                j=2*i;              //Primzahlensieb aufbauen, für obige
                        Primzahlenerkennung
35                while (j < pow(iEingabe,1/2))
36                {
37                    sieb[j/iIntsize]|=1<<(j&iModIntsize);
38                    j+=i;
39                }
40            }
41            i++;
42        }
43        printf("\n\n");
44    }
45 }
```

Der Aufruf des Compilers geschieht mittels:

```
mipsel-linux-uclibc-gcc prime.c -oprime
```

Bei “mipsel-linux-uclibc-gcc” handelt es sich um den Compiler, in diesem Falle einen der Cross-Compiler. “prime.c” ist die Eingabedatei und “-oprime” teilt dem Compiler mit, als Ausgabedatei “prime” zu wählen. Es ist korrekt, dass zwischen dem “o” und “prime” kein Leerzeichen steht.

Der Compiler besteht aus mehreren Teilprogrammen, die automatisch - sofern dem Compiler nichts anderes über einen Parameter mitgeteilt wird - nacheinander ausgeführt werden. In der korrekten Reihenfolge sind das:

- Präprozessor
- Compiler
- Assembler
- Linker

Der Name “Compiler” hat dabei mehrere Bedeutungen. Zum einen steht er für den Teil des Programmes, welcher den oben beschriebenen “Compiler”-Vorgang ausführt, zum anderen für das gesamte Compiler-Executable, welches alle vier Schritte auszuführen vermag.

In dem oben abgebildeten Programm “prime” werden Funktionen verwendet, welche nicht Grundbestandteil von C sind, sondern aus Bibliotheken stammen. Hierzu gehört die Funktion “pow”, die eine Potenz errechnet. In diesem Fall ist es die Potenz von einhalb, das entspricht der Quadratwurzel. Um diese Funktion dem Compiler zugänglich zu machen, muss sie diesem mitgeteilt werden. Hierzu dienen die #include-Statements zu Beginn des Quelltextes. Der Präprozessor inkludiert an diese Stellen den Inhalt der dort aufgeführten Header-Dateien. Die Header-Dateien werden mit den zugehörigen Bibliotheken mitgeliefert und enthalten die Funktionen in Form von Prozeduren, auf die dann der Compiler im “Compile”-Vorgang zurückgreift. In diesem Fall ist es die “math”-Bibliothek.

Aus dem Kompilervorgang gehen Assembler-Dateien hervor, diese wiederum werden beim Assemblieren in Objekt-Dateien konvertiert, einer Binärversion unseres Quellco-

des. Dabei handelt es sich um “losen” Binärcode, der schon prozessorspezifisch ist und im Gegensatz zum Quellcode nicht binärkompatibel mit anderen Architekturen. Bibliotheken werden häufig in diesem Format ausgeliefert, um Zeit für den Kompilervorgang zu sparen, bei kommerziellen Anwendungen aber auch, um den Quellcode geheim zu halten. Aus jeder C-Datei entsteht eine Objekt-Datei mit der Endung “.o”. Zwischen Assembler und Compiler wird nicht immer unterschieden, auch in dieser Thesis wird nicht auf den Unterschied eingegangen.

Im letzten Schritt erfolgt das Linken. Im Falle des “Prime”-Programmes wird aus der Objekt-Datei prime.o ein Executable. Besteht das Programm aus mehreren c-Dateien, so werden diese zu einem einzigen Executable zusammengefügt. Bibliotheken, besser: Teile davon, werden, je nachdem ob dynamisch oder statisch gelinkt wird, mit in das Executable hineinkopiert oder extern referenziert. [58]

Besteht das Programm aus vielen C-Dateien, muss der Compiler für jedes C-File einmal ausgeführt werden. Es müssen für diesen Fall erst alle “.c”-Dateien in Objekt-Dateien umgewandelt werden um diese dann im letzten Schritt zu einem großen Executable zu linken. Um diesen Schritt zu automatisieren, wurde die Programmiersprache “Make” entwickelt. In dieser werden auszuführende Schritte definiert und Abhängigkeiten zu Source-Dateien, Header-Dateien und Bibliotheken definiert, die, wenn sie nicht erfüllt sind, zu einem Fehler führen.

Innerhalb dieser Make-Skripte werden dem Compiler Parameter, sogenannte “Compiler-Flags”, übergeben, die den Compiler im ersten Schritt aus allen “.c”-Dateien eine Objekt-Datei erzeugen lassen und im zweiten Schritt diese zu einem fertigen Programm linken lassen. Ebenso gibt es Compiler-Flags für Optimierungen auf Größe oder Geschwindigkeit des Executable. Sie werden an zentraler Stelle im Make-File definiert und von Make bei jedem Compileraufruf übergeben. Ebenso können dem Make-Befehl Parameter übergeben werden, mit denen nur Teile des Programmes übersetzt werden. Die für die Fritzbox bestimmte Freetz-Umgebung kann beispielsweise mit “make precompiled” dazu angewiesen werden, nur die Programme zum Übersetzen der Firmware zu generieren, nicht aber die Firmware selbst, die für das Netcenter nicht benötigt wird. Den zu nutzen Compiler bestimmt make in der Regel über die Shell-Variable “CC”. Diese muss für das Cross-Compiling angepasst werden, ansonsten wird der Standard-GCC verwendet, der nativen Code für x86-Prozessoren oder x64-Prozessoren erzeugt. Dieser läuft nicht auf dem Netcenter.

Größere Projekte bringen häufig auch noch ein `./configure`-Skript mit. Dieses kann verschiedene Parameter des späteren Executable anpassen und generiert meist das Makefile. Folgende Eigenschaften werden vom `./configure`-Skript geändert oder geprüft [59]:

- Installationsort des Programmes
- Bitbreite des Betriebssystems um ein passendes Makefile auszuwählen oder zu generieren
- Vorhandensein von Quelldateien und Bibliotheken
- Versionsprüfung von Compiler und Bibliotheken auf Kompatibilität
- Ändern der Programmfeatures durch anpassen der Makefile: “`--with-ssl-option`”, “`--without-zlib`”

Ein Beispiel für verschiedene Shell-Parameter, die von `./configure`-Skripten ausgelesen und an Make übergeben werden, liefert das dem Privoxy mitgelieferte `./configure`-Skript mit dem Parameter `help`:

“`./configure -help`”

Listing 7: Shell-Variablen für `./configure`

---

```
1 ./configure --help
2
3 Some influential environment variables:
4 CC          C compiler command
5 CFLAGS      C compiler flags
6 LDFLAGS     linker flags, e.g. -L<lib dir> if you have libraries in a
7             nonstandard directory <lib dir>
8 LIBS        libraries to pass to the linker, e.g. -l<library>
9 CPPFLAGS    C/C++/Objective C preprocessor flags, e.g. -I<include dir> if
10            you have headers in a nonstandard directory <include dir>
11 CPP         C preprocessor
12
13 Use these variables to override the choices made by 'configure' or to help
14 it to find libraries and programs with nonstandard names/locations.
```

---

Weiterhin gibt es Tools mit dem Namen Autoheader und Autoconf, welche in der Lage sind, typische “`./configure`”-Skripte zu generieren. Privoxy macht hiervon gebrauch.

Von dem Einsatz der hnd-toolchain Version 3.0 wird in verschiedenen Internetforen abgeraten, da die Toolchain diverse Fehler aufweist, die das korrekte Kompilieren diverser Software verhindert. Daher wurde für erste Versuche die hnd-toolchain 3.23 gewählt und das bereits erwähnte Programm Prime erfolgreich kompiliert.

Das Kompilieren von Busybox schlug im Gegensatz dazu mit dieser Toolchain reproduzierbar fehl. Getestet wurden diverse als “stable” gekennzeichnete Versionen. Der Kompilierungsvorgang brach noch beim Erstellen der Objekt-Files ab, lief allerdings bis zum Abbruch eine ganze Weile durch. Die Quelldatei, in der der Fehler gemeldet wurde, war abhängig von der kompilierten Busybox-Version, trat bei mehrfachem Kompilieren einer Version allerdings reproduzierbar an der gleichen Stelle auf. Ebenso keine Rolle spielte das verwendete Linux-Betriebssystem: Der Fehler trat sowohl unter einer KDE-basierten 32 Bit Knoppix-Version als auch auf einer Gnome basierten 64 Bit Ubuntu-Version auf - stets an der gleichen Stelle, je nach Busybox-Version. Der Fehler wurde daher dem Compiler zugeschrieben und stattdessen die Freetz-Toolchain und im Falle der Busybox die uclibc-Toolchain verwendet. Es verwunderte den Autor, dass im Internet mehrfach vom erfolgreichen Einsatz der Toolchain die Rede war.

Listing 8: hnd-toolchain 3.23: Busybox Compilerfehler

---

```
1 make defconfig
2 make
3
4 AR      modutils/lib.a
5 CC      networking/arp.o
6 CC      networking/arping.o
7 CC      networking/brctl.o
8 networking/brctl.c: In function 'brctl_main':
9 networking/brctl.c:186: 'SIOCBRADDR' undeclared (first use in this function)
10 networking/brctl.c:186: (Each undeclared identifier is reported only once
11 networking/brctl.c:186: for each function it appears in.)
12 networking/brctl.c:186: 'SIOCBRDELBR' undeclared (first use in this function)
13 networking/brctl.c:202: 'SIOCBRADDIF' undeclared (first use in this function)
14 networking/brctl.c:202: 'SIOCBRDELIF' undeclared (first use in this function)
15 make[1]: *** [networking/brctl.o] Fehler 1
16 make: *** [networking] Fehler 2
```

---

Um die Freetz-Toolchain zu erstellen, muss die aktuelle Version der Freetz-Umgebung heruntergeladen werden. Diese findet sich auf [\[60\]](#) und lag zum Zeitpunkt des Entstehens dieser Arbeit in der Version 1.01 vor. Die Installation von Freetz wurde weitestgehend über die Anleitung in [\[61\]](#) durchgeführt.

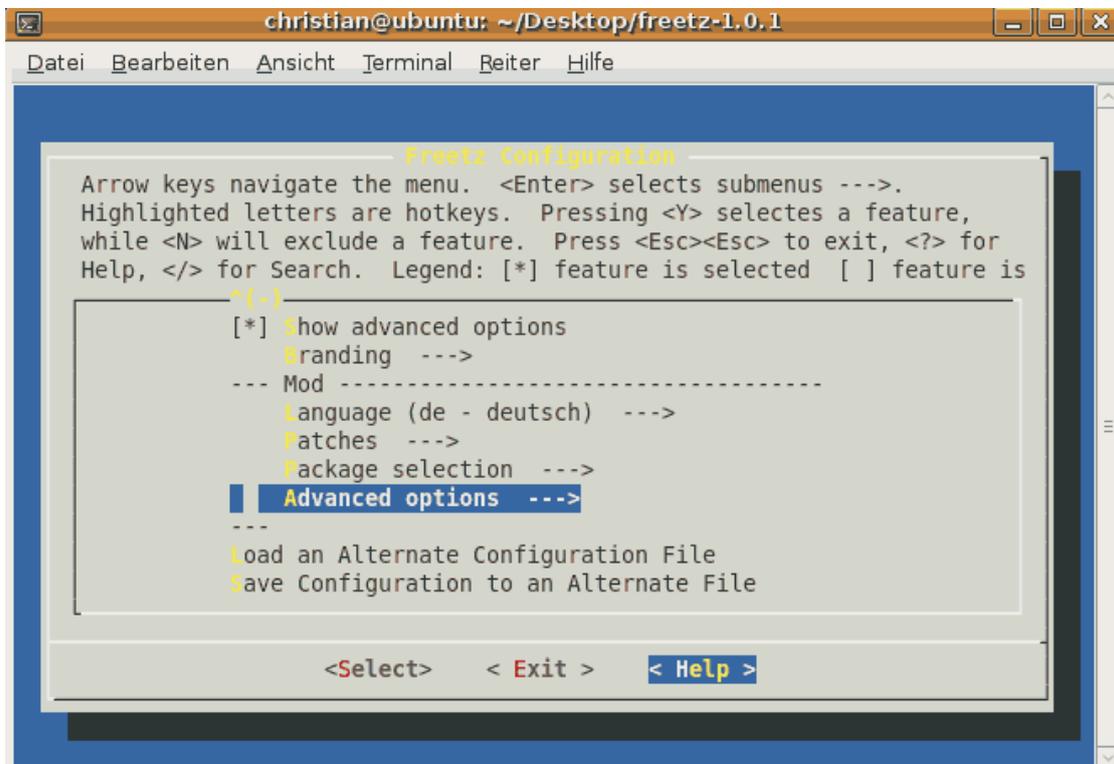


Abbildung 10: Busybox: Menu von “make menuconfig”

Nach dem Entpacken des Archives ist sicherzustellen, dass mit dem folgenden Befehl (unter Debian-basierten Linux-Umgebungen) alle für Freetz nötigen Pakete installiert sind. Auf dem Rechner des Autors gab es bei dem Lösen der Abhängigkeiten Probleme mit dem Paketmanager, sodass auf eine zweite Ubuntu-Installation via Wubi zurückgegriffen werden musste. Die Ursache hierfür wurde noch nicht gefunden.

```
sudo apt-get install gcc g++ binutils autoconf automake libtool make bzip2 libncurses5-dev zlib1g-dev flex bison patch texinfo tofrodos gettext jam pkg-config ecj perl libstring-crc32-perl
```

Nach wechseln in das entpackte Freetz-Verzeichnis wurde die Konfiguration von Freetz erstellt mittels

```
make menuconfig
```

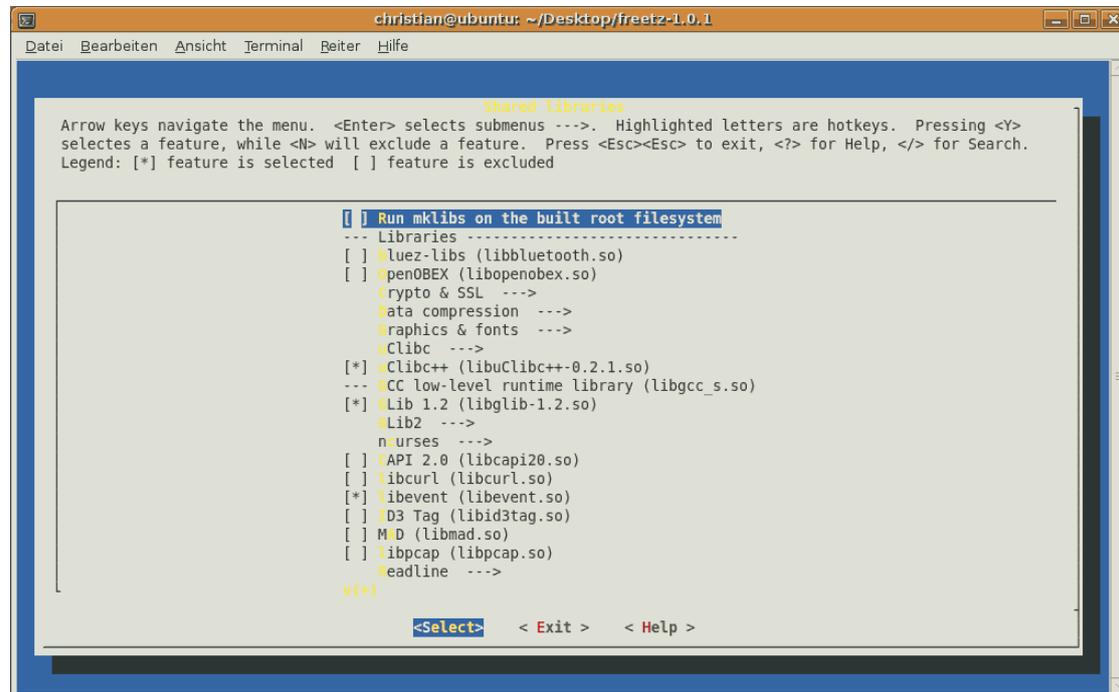


Abbildung 11: Auswahl der Bibliotheken im “make menuconfig” von Freetz

Von Bedeutung sind die Einstellungen unter “Advanced Options” – “Shared Libs”. Empfohlen, da für viele Programme benötigt, sind die folgenden Bibliotheken. Die ersten drei dieser Liste werden für das Übersetzen von TOR zwingend benötigt:

- “Crypto” – “OpenSSL”
- “libevent”
- “compression” – “zlib”
- “uClibc” – “libresolv”, “libthread”, “libutil”
- “ncurses” – “ncurses”
- “libcurl”

Die für TOR benötigten Pakete werden automatisch ausgewählt, wählt man unter “Packet selection” – “Standard packages” das Paket TOR aus. Mit den im nächsten Abschnitt aufgeführten “make”-Kommandos wird daraus jedoch kein Executable generiert. Da die TOR Version 2.0.31 ohnehin veraltet ist, wird TOR in einem der folgenden Unterkapitel manuell kompiliert. Die Auswahl des Fritzboxmodells ist im Falle des Net-

centers nicht von Bedeutung, da weder der Quellcode für den Kernel benötigt wird, noch ein fertiges Firmwareimage erstellt wird.

Nach korrekter Auswahl wird das Erstellen der Toolchain gestartet. Je nach Auswahl im “menuconfig“ wird die Toolchain aus dem Quelltext kompiliert oder in der Binärversion heruntergeladen. Ersteres benötigt 4 Gigabyte freien Festplattenspeicher, der auf der Wubi-Platte des Autors nicht vorhanden war, sodass die vorkompilierte Version gewählt wurde. Der Download der Toolchain und der Bibliotheken nimmt einige Zeit in Anspruch und wird mit dem folgenden Befehl gestartet:

```
make precompiled
```

Soll das Erstellen der Toolchain und das Kompilieren der Bibliotheken getrennt durchgeführt werden, so sind die folgenden Befehle zu verwenden:

```
make toolchain  
make libs
```

Der Compiler ist nun im Unterverzeichnis “/toolchain/target“ des aktuellen Verzeichnisses installiert. Kopiert man das toolchain-Verzeichnis samt Inhalt nach “/opt“, besitzt die Compiler-Executable den Pfad “/opt/toolchain/target/bin/mipsel-linux-uclibc-gcc“. Alle Pfade der im nächsten Kapitel folgenden Konfiguration müssen an die Pfade der eigenen Installation angepasst werden.

Sollte man alle durch “make“ erstellten Programme nicht mehr benötigen und den Zustand nach dem Entpacken der Quellcode-Dateien wiederherstellen, hilft folgendes Kommando weiter:

```
make clean
```

### 3.5 Erste erfolgreiche Compiling-Versuche: prime und busybox

Das selbstgeschriebene, sehr kurze Programm “prime“ wurde erfolgreich mit folgendem Kommando kompiliert:

*mipsel-linux-uclibc-gcc prime.c -oprime*

Ein kompiliertes Programm enthält noch Debugging-Informationen, die zum Ausführen nicht benötigt werden. Da auf Embedded-Devices der Speicherplatz sehr knapp ist, wird dieser vor dem Kopieren auf das Gerät mit dem Befehl “strip” entfernt. Strip ist ein Teil der Toolchain und ist architekturabhängig. Zur Überprüfung, welcher Dateityp nach der Kompilierung vorliegt, eignet sich das Programm “file”.

Das Programm Prime wurde zweimal kompiliert. Das erste Mal wurde dynamisch, das zweite Mal statisch gelinkt. Vor und nach dem Striping wurde jeweils die Dateigröße und der Dateityp (mittels “file”) ausgelesen. Nähere Informationen zu dynamischem und statischem Compiling finden sich im Kapitel Bibliotheken 3.6.

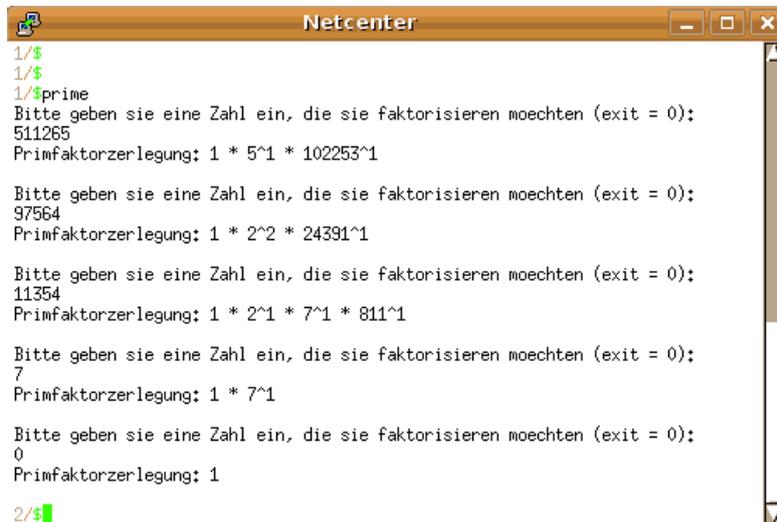
Listing 9: Kompilationsvorgang von Prime

---

```
1 ~/prime$ mipsel-linux-uclibc-gcc prime.c -oprime
2 ~/prime$ ls -lh prime
3 -rwxr-xr-x 1 christian christian 23K 2009-02-03 15:19 prime
4 ~/prime$ file prime
5 prime: ELF 32-bit LSB executable, MIPS, MIPS32 version 1 (SYSV), dynamically
   linked (uses shared libs), not stripped
6
7 ~/prime$ mipsel-linux-uclibc-strip prime
8 ~/prime$ ls -lh prime
9 -rwxr-xr-x 1 christian christian 8,0K 2009-02-03 19:36 prime
10 ~/prime$ file prime
11 prime: ELF 32-bit LSB executable, MIPS, MIPS32 version 1 (SYSV), dynamically
   linked (uses shared libs), stripped
12
13 ~/prime$ mipsel-linux-uclibc-gcc prime.c -oprime --static
14 ~/prime$ ls -lh prime
15 -rwxr-xr-x 1 christian christian 90K 2009-02-03 19:37 prime
16 ~/prime$ mipsel-linux-uclibc-strip prime
17 ~/prime$ ls -lh prime
18 -rwxr-xr-x 1 christian christian 48K 2009-02-03 19:38 prime
19
20 ~/prime$ file prime
21 prime: ELF 32-bit LSB executable, MIPS, MIPS32 version 1 (SYSV), statically linked
   , stripped
22 ~/prime$
```

---

Im Anschluss wurde die statisch kompilierte Datei auf das Netcenter kopiert und dort ausgeführt:



```
Netcenter
1/$
1/$
1/$prime
Bitte geben sie eine Zahl ein, die sie faktorisieren moechten (exit = 0):
511265
Primfaktorzerlegung: 1 * 5^1 * 102253^1

Bitte geben sie eine Zahl ein, die sie faktorisieren moechten (exit = 0):
97564
Primfaktorzerlegung: 1 * 2^2 * 24391^1

Bitte geben sie eine Zahl ein, die sie faktorisieren moechten (exit = 0):
11354
Primfaktorzerlegung: 1 * 2^1 * 7^1 * 811^1

Bitte geben sie eine Zahl ein, die sie faktorisieren moechten (exit = 0):
7
Primfaktorzerlegung: 1 * 7^1

Bitte geben sie eine Zahl ein, die sie faktorisieren moechten (exit = 0):
0
Primfaktorzerlegung: 1

2/$
```

Abbildung 12: Ausführung von Prime auf dem Netcenter

Das Vorgehen zum Kompilieren der Busybox-Quellen stammt von [50]. Der Freetz-Compiler ist von Haus aus nicht in der Lage, die Busybox mit den Standard-Optionen zu linken. Es kommt stets zu einem Abbruch mit der folgenden Fehlermeldung. Eine Recherche nach der Fehlerursache führte zu keinem Ergebnis, die Ursache scheint in der uclibc-Bibliothek des Freetz-Compilers zu liegen. Daher wurde zum Kompilieren der Busybox der uclibc-Crosscompiler verwendet.[51]

Listing 10: Fehlerhaftes Linken der Busybox mit Freetz

```
1 AR      util-linux/volume_id/lib.a
2 LINK    busybox_unstripped
3 Trying libraries: crypt m
4 Failed: -Wl,--start-group -lcrypt -lm -Wl,--end-group
5 Output of:
6 mipsel-linux-gcc -Wall -Wshadow -Wwrite-strings -Wundef -Wstrict-prototypes -
  Wunused -Wunused-parameter -Wmissing-prototypes -Wmissing-declarations -
  Wdeclaration-after-statement -Wold-style-definition -fno-builtin-strlen -
  finline-limit=0 -fomit-frame-pointer -ffunction-sections -fdata-sections -fno-
  guess-branch-probability -funsigned-char -static-libgcc -falign-functions=1 -
  falign-jumps=1 -falign-labels=1 -falign-loops=1 -Os -L/opt/toolchain/kernel/
  lib -o busybox_unstripped -Wl,--sort-common -Wl,--sort-section,alignment -Wl
  ,--gc-sections -Wl,--start-group applets/built-in.o archival/lib.a archival/
  libunarchive/lib.a console-tools/lib.a coreutils/lib.a coreutils/libcoreutils/
  lib.a debianutils/lib.a e2fsprogs/lib.a editors/lib.a findutils/lib.a init/lib
  .a libbb/lib.a libpwdgrp/lib.a loginutils/lib.a mailutils/lib.a miscutils/lib.
  a modutils/lib.a networking/lib.a networking/libiproute/lib.a networking/udhcp
  /lib.a printutils/lib.a procps/lib.a runit/lib.a selinux/lib.a shell/lib.a
  sysklogd/lib.a util-linux/lib.a util-linux/volume_id/lib.a archival/built-in.o
  archival/libunarchive/built-in.o console-tools/built-in.o coreutils/built-in.
  o coreutils/libcoreutils/built-in.o debianutils/built-in.o e2fsprogs/built-in.
  o editors/built-in.o findutils/built-in.o init/built-in.o libbb/built-in.o
  libpwdgrp/built-in.o loginutils/built-in.o mailutils/built-in.o miscutils/
  built-in.o modutils/built-in.o networking/built-in.o networking/libiproute/
  built-in.o networking/udhcp/built-in.o printutils/built-in.o procps/built-in.o
  runit/built-in.o selinux/built-in.o shell/built-in.o sysklogd/built-in.o util
  -linux/built-in.o util-linux/volume_id/built-in.o -Wl,--end-group -Wl,--start-
  group -lcrypt -lm -Wl,--end-group
7 =====
8 miscutils/lib.a(readahead.o): In function 'readahead_main':
9 readahead.c:(.text.readahead_main+0x6c): undefined reference to 'readahead'
10 collect2: ld returned 1 exit status
11 make: *** [busybox_unstripped] Fehler 1
```

Im ersten Schritt nach dem Herunterladen und Entpacken des Quellcodes von [52] werden Compiler-Optionen in der Shell gesetzt, welche Make den Cross-Compiler und die Verzeichnisse für Bibliotheken und Header-Dateien (Include-) mitteilen. Im Anschluss wurde in das Busybox-Verzeichnis gewechselt.

```
export PATH=/opt/cross-compiler-mipsel/bin:$PATH
export CC=/opt/cross-compiler-mipsel/bin/mipsel-gcc
export LDFLAGS=-L/opt/cross-compiler-mipsel/lib
export CPPFLAGS=-I/opt/cross-compiler-mipsel/include
export CPP=/opt/cross-compiler-mipsel/bin/mipsel-cpp
export CROSS_COMPILE=mipsel-
export ARCH=mipsel
```

Im Anschluss wird eine Standard-Konfiguration erzeugt:

```
make defconfig
```

Als mögliche Konfiguration zur Auswahl stehen:

- defconfig** Erstellt eine Standard-Konfiguration, die fast alle Funktionen implementiert.
- menuconfig** Ruft ein Menü auf, um manuell Funktionen zu aktivieren oder deaktivieren.
- allyesconfig** Erstellt eine Konfiguration, die jede verfügbare Funktion aktiviert, unter anderem die Build-in-shell-Funktion.[\[62\]](#)
- allnoconfig** Erstellt eine Konfiguration, die keine Busybox-Funktion aktiviert. Nützlich als proof-of-concept des Kompiliervorganges oder als Vorlage für menuconfig

Nun wird das Programm kompiliert mit dem Befehl:

```
make
```

---

Listing 11: Kompilation der Busybox

---

```
1 CC      util-linux/volume_id/xfs.o
2 AR      util-linux/volume_id/lib.a
3 LINK    busybox_unstripped
4 Trying libraries: crypt m
5 Library crypt is not needed, excluding it
6 Library m is needed, can't exclude it (yet)
7 Final link with: m
8 DOC     busybox.pod
9 DOC     BusyBox.txt
10 DOC    BusyBox.1
11 DOC    BusyBox.html
```

---

Busybox führt automatisch einen Strip der Datei durch. Nach dem Kopieren der Busybox-Executable auf das Netcenter und dem Vergeben der Ausführungsrechte kann die Datei gestartet werden. In diesem Fall wurde ein Nightly-Build von Anfang Januar verwendet, diese wurde jedoch aufgrund eines Fehlers in der integrierten Login-Shell später durch das letzte Stable-Release 1.13.2 ersetzt. In dieser wurden einige Anpassungen über “menuconfig” durchgeführt, beispielsweise das “colorful”-ls als Standard aktiviert. Das Ersetzen der mitgelieferten Busybox-Version des Netcenters durch die neue Version erwies sich als Fehlschlag, das Netcenter startete nicht mehr und mußte über den Recovery-Modus mit der vorherigen Firmware bespielt werden. Der Fehler konnte auf die integrierte “insmod”-Funktion zum Einbinden der Kernelmodule eingegrenzt werden. Auch durch Anpassen via menuconfig lies sich die neue Busybox-Insmod-Funktion bis zum Abschluss der Arbeit nicht zur Zusammenarbeit mit dem Netcenter-Kernel überreden. Näheres über das Verändern der Firmware findet sich in [A.2](#), das zurückspielen einer Funktionsfähigen Firmware im Recoverymodus unter [A.4](#)

Listing 12: Busybox 1.13.2: Probleme mit Insmod

```
1 #/lib/modules/2.4.20/kernel/arch/mips/brcm-boards/bcm947xx$/sbin/busybox insmod
   bcmaecgpo.o
2 insmod: cannot insert 'bcmaecgpo.o': unknown symbol in module or invalid parameter
3 #/lib/modules/2.4.20/kernel/arch/mips/brcm-boards/bcm947xx$/bin/busybox insmod
   bcmaecgpo.o
4 Using bcmaecgpo.o
```

Software, die mit den hier behandelten Toolchains erstellt wurde, ist binärkompatibel über die oben aufgeführten MIPSel-Geräte hinweg. Kompatibilität gilt allerdings aus den folgenden Gründen nicht uneingeschränkt:

**Root-Filesystem** Die Geräte besitzen unterschiedliche Root-Filesysteme. Während bei dem Netcenter und dem Maxtor Shared Storage der Ordner /opt/ auf den beschreibbaren Teil der eingebauten Festplatte verweist, nutzt die Fritzbox den Ordner /usr/ zur Installation von Software. Bei Auslieferung der Software als IPKG-Paket muss dieses umgeschrieben werden, damit die Software sich im korrekten Pfad installiert. Bei Installation als Binary kann der Ablageort frei gewählt werden, allerdings erwartet das Programm Konfigurationsdateien an bestimmten Orten: Cross-kompilierte Anwendungen enthaltenen einen angepassten Konfigurationspfad, der ebenfalls auf das Zielgerät abgestimmt ist. IPKG-Pakete für das Netcenter oder MSS erwarten die Konfiguration statt in /etc/ in /opt/etc. Wird die



weniger gefährlich, näheres hierzu findet sich im Anhang [A.4](#).

**Hardwareausstattung** Die oben aufgeführten Geräte bringen zwar den gleichen Prozessor mit, erfüllen aber einen grundlegend anderen Zweck und besitzen somit eine andere Hardwareausstattung. Während die Fritzbox und der Linksys-Router für das Masquerading, Routing und Bridging konstruiert wurden, dient das Netcenter und das MSS zum Zugriff auf Netzwerkspeicher. Letztere benötigen zum Cachen deutlich mehr Arbeitsspeicher als erstere, weiterhin wird auch der zur Verfügung stehende Platz für zusätzliche Programme auf den Geräten mit eingebauter Festplatte nicht knapp. Die ersten Fritzbox- und Linksysgeräte brachten lediglich einen zwischen 2 MB und 8 MB kleinen Flash-Speicher mit, der Arbeitsspeicher lag zwischen 8 MB und 16 MB. Hier war es nicht möglich, mehrere Programme in den Flash zu integrieren. Durch stetig günstigeren Speicher und immer größere Funktionsvielfalt wird der Unterschied mit zukünftigen Gerätegenerationen immer geringer. Neuere Fritzboxen oder Linksys-Router bieten USB-Support für Massenspeicher, sodass der Programmspeicher nicht mehr auf den kleinen Flash-Speicher begrenzt ist. Auch der Ausbau des Arbeitsspeichers nimmt zu, moderne Netzwerkfestplatten bieten meist schon 64 oder 128 MB. Hier reicht der Speicher aus, um auch komplexe Programme auszuführen, ohne auf Erweiterung des Arbeitsspeichers durch eine SWAP-Partition angewiesen zu sein. SWAP-Speicher bietet sich deshalb nicht an, da das integrierte Festplattenlaufwerk bei Nutzung des SWAPs nicht in den Schlafzustand übergeht.

**Kernel** Die Geräte nutzen unterschiedliche Kernelversionen. Diese bereiten bei hardwarenahen Programmen Probleme, da Kernelmodule kernelversionsspezifisch sind. Sind entsprechenden Module nicht auf dem Gerät vorhanden, müssen sie für die auf dem Zielgerät vorhandene Kernelversion mitgeliefert werden. Das ist in der Regel nicht der Fall. Ein Programm, welches auf Kernelmodule angewiesen ist, ist beispielsweise IP-Tables. Die Fritzbox nutzt je nach Firmwareversion einen 2.4-Kernel oder einem 2.6-Kernel. Das MSS, das Netcenter sowie die Linksys WRT-Router nutzen jeweils Kernelversion 2.4.20 und sind somit auf dieser Ebene kompatibel.

**Bibliotheken** Das Problem verschiedener, inkompatibler Bibliotheken ist ein immer wieder anzutreffendes Linux Problem, es wird im nächsten Kapitel erläutert.

### 3.6 Bibliotheken

Eine häufig genutzte Bibliothek ist die glibc. Da diese recht umfangreich ist - eine für das Netcenter vollständig kompilierte Version benötigt 10 MB - gibt es abgespeckte Versionen, die einen verminderten Funktionsumfang bei deutlich reduzierter Dateigröße besitzen. Angelehnt an das Zeichen “ $\mu$ ” wurde sie uclibc genannt. Programme, die für die glibc geschrieben wurden, kommen mit der uclibc unter Umständen nicht zurecht. Auch Versionsunterschiede können Probleme bei der gemeinsamen Nutzung einer Bibliothek verursachen, denn Bibliotheken sind nicht immer rückkompatibel. Aufgrund von Softwarefehlern ist dieser Umstand nicht immer beabsichtigt. Sollten zwei Programme zwei zueinander inkompatible Bibliotheksversionen benötigen, die den gleichen Namen tragen, ist das erst einmal problematisch. Bei eigens kompilierter Software besteht dann die Möglichkeit, die Programme statisch zu linken.

Wird ein Programm geschrieben, welches beispielsweise Kommunikation über das Internet ermöglicht, ist unter Umständen abhörsichere Kommunikation erwünscht. Viele kluge Entwickler haben sich über dieses Thema bereits Gedanken gemacht und diese Ideen als “Fertiglösung” in Form einer Bibliothek im Internet bereitgestellt. Es ist nicht nötig, das Rad immer neu zu erfinden.

Bei der Programmentwicklung gibt es jene zwei Möglichkeiten, um bestimmte Funktionen zu realisieren: Man schreibt sie selbst, oder man nutzt vorhandene Funktionen in Form von Bibliotheken. Aus mehreren Gründen ist es sinnvoll, bestehende Funktionen zu nutzen, statt sie selbst zu schreiben:

**Entwicklungsaufwand** Was schon vorhanden ist, muss nicht erneut entwickelt und programmiert werden. Daher ist der Entwicklungsaufwand geringer, greift man auf bestehende Funktionen zurück.

**Sicherheit** Bestehende Bibliotheken werden von fachlich versierten Entwicklern programmiert und gewartet. Die Entwickler kennen sich mit der Materie sehr gut aus. Insbesondere im kryptographischen Bereich sind Entwicklungen über viele Jahre von klugen Köpfen entwickelt und optimiert worden. Man sollte nicht ernsthaft glauben, mathematische Algorithmen mit ähnlicher Sicherheit als Nebenprodukt der eigenen Anwendung dahinschreiben zu können.

**Geschwindigkeit** Bibliotheken werden von Entwicklerteams betreut, die sich auf das spezialisiert haben, was man selbst lediglich als “Randfunktion” oder nettes Feature benötigt. Durch den hohen Entwicklungsaufwand, im Gegensatz zur Eigenentwicklung, besitzen diese Funktionen in der Regel deutlich höhere Geschwindigkeiten. Dies gilt insbesondere für Datenkompressionsalgorithmen.

**Interoperabilität** Wer auf Bibliotheken baut, der ist dazu gezwungen, sein Programm in verschiedene Schichten aufzuteilen. Die oberste Schicht stellt im Falle des eingangs erwähnten Kommunikationsprogrammes die eigens entwickelte Anwendungslogik, gefolgt von eventueller Kompression, Verschlüsselung, am Schluss der Datenversand oder Speichervorgang. Wird eine Bibliothek von mehreren Anwendungsprogrammen genutzt, ist es für die Entwickler einfach, Kompatibilität zwischen verschiedenen Anwendungen zu wahren. Das führt in der Regel zu einem höheren Verbreitungsgrad der Programme. Die erzwungene Modularisierung führt zu leichter wartbarem Programmcode.

Insbesondere kommerzielle Entwicklungen fürchten wegen Patentstreitigkeiten die Nutzung solcher offenen Bibliotheken. Häufig werden Funktionen selbst entwickelt, deren Aufgabe keine Überschneidung mit der Kernkompetenz der Entwickler besitzt. Aufgrund der nicht öffentlichen Quellen bleiben auch Entwicklungsfehler längere Zeit unentdeckt. Gute Beispiele liefert die Praxis: Die Funkverschlüsselung des drahtlosen Türöffners Keeloq [63] sowie die RFID-Verschlüsselung Mifare [64] wurden auf eben jene Weise entwickelt. Die eigens entworfenen Sicherheitsfunktionen stellten sich als nicht praxistauglich heraus.

Die Nutzung von Bibliotheken kann auch gänzlich andere Gründe haben: So stellen viele Hersteller Programmschnittstellen oder Funktionen nur über dokumentierte Bibliotheken zur Verfügung, um den Quellcode nicht preisgeben zu müssen. Intel stellt beispielsweise mathematische Bibliotheken zur Verfügung, die stark auf das Prozessordesign aus dem eigenen Hause optimiert wurden. [77]

Die Einbindung dieser Bibliotheken kann bei der Kompilierung des Quellcodes auf zwei verschiedene Arten geschehen: Statisch oder dynamisch.

**statisch** Sind Bibliotheken statisch eingebunden, kopiert der Compiler beim Linken des Programms die Bibliotheksfunktionen in das ausführbare Programm mit hinein. Dadurch entsteht eine einzige, große, ausführbare Datei. Man ist nicht auf zusätzliche, externe Bibliotheken angewiesen. Dadurch, dass die Bibliotheksfunktionen im Programm integriert sind, wird die Programm-Executable größer. Werden die Bibliotheken einzig für dieses Programm benötigt, kann es dennoch zu einer Platzersparnis kommen, da nur die für das Programm notwendigen Funktionen der Bibliothek integriert werden.

**dynamisch** Dynamische Bibliotheken haben ebenso Vor- und Nachteile. Greifen viele Programme auf eine Bibliothek zu, entsteht eine große Platzersparnis. Nicht nur auf der Festplatte, sondern auch im Hauptspeicher können die Programme auf gemeinsame Funktionen zugreifen. Werden Fehler in einer Bibliothek behoben, reicht es, die fehlerbereinigten Bibliotheksversion aufzuspielen, um den Fehler in allen darauf zugreifenden Anwendungen zu beheben. Es ist nicht nötig, die darauf zugreifenden Programme neu zu übersetzen. Gibt es hingegen neue Version, deren Veränderungen sich in einem Programm positiv bemerkbar macht, beispielsweise durch höhere Geschwindigkeit, in anderen Programmen hingegen negativ, zum Beispiel durch Abstürze, so hat man nur über Umwege die Möglichkeit, beide Bibliotheksversionen parallel zu betreiben.

Bibliotheken auf Desktop-Computern werden durch eine Vielzahl von Anwendern getestet, enthaltene Fehler werden daher zeitnah gefunden und korrigiert. Die gleichen Bibliotheken werden vergleichsweise selten über Cross-Compiler-Toolchains kompiliert. Durch die große Anzahl von Toolchains und eingesetzten Compiler-Versionen kommt es hier umso häufiger zu Inkompatibilitäten zwischen verschiedenen Programmen und ihren abhängigen Bibliotheken. Die akute Speicherknappheit auf Embedded-Devices macht das dynamische Linken von häufig eingesetzten Bibliotheken auf der anderen Seite besonders empfehlenswert. Aufgrund des großen Fehlerpotentials und mehrerer eingesetzter Compiler-Versionen wurden alle in dieser Thesis kompilierten Programme dennoch statisch gelinkt.

Inkompatible Bibliotheken sind auf dem Netcenter fatal: Fährt die Festplatte aufgrund von fehlerhaften Bibliotheken, die nicht im Flash, sondern auf dem Festplattenlaufwerk liegen, nicht mehr hoch, bleibt nur der Einbau des Laufwerkes in den Desktop-Rechner. Hier ist der Zugriff jedoch nur über einen korrigierten Master-Boot-Record möglich, da dieser - wie in Kapitel 2.1.2 beschrieben - nicht dem Standard-MBR entspricht. Ein korrekter MBR kann beispielsweise mit dem Datenrestaurationsprogramm Testdisk erstellt werden[76]. Nach beheben des Problems muss der vorherige MBR zurückgespielt werden. Diese Prozedur ist fehleranfällig, aufwändig und birgt die Gefahr des Datenverlustes. Es bietet sich daher an, einen separaten Ordner für Test-Bibliotheken zu erstellen, den man manuell nach dem Starten der Festplatte als Bibliotheksordner mit höherer Priorität als den Standard-Ordner einbindet. Geht etwas schief, ist nach einem Reboot der alte Zustand wiederhergestellt. Startet das Netcenter aufgrund falscher Bibliotheken nicht mehr und ein Recovery über einen Desktop-Rechner gelingt nicht, hilft nur das Aufspielen eines fabrikneuen Festplattenabbildes unter Verlust aller auf dem Netcenter enthaltenen Daten. Das Abbild findet sich auf der dieser Thesis beiliegenden CD unter Binary-Software/HDD-RAW-Recoveryimage als .gz Archiv und kann über einen Desktop-Rechner zurückgespielt werden. Alternativ kann auf diese Weise eine neue Festplatte für das Netcenter eingerichtet werden:

```
sudo gunzip WDC-Netcenter.gz -c > /dev/sdb
```

/dev/sdb steht für das extern angeschlossene Laufwerk. Achtung: Dieser Befehl überschreibt den Zieldatenträger unumkehrbar und ohne Rückfrage!

Läuft ein dynamisch gelinktes oder ein aus einer Paketquelle installiertes Programm aufgrund fehlender Bibliotheken nicht, hilft das Programm ldd.sh. Dieses Skript listet vorhandene und fehlende Bibliotheken der als Parameter übergebenen Binärdatei, sodass die fehlenden Bibliotheken auf das Netcenter kopiert oder, falls schon vorhanden, korrekt verlinkt werden können. Das Skript ist Bestandteil gängiger Linux-Distributionen und wurde um einige Funktionen entlastet, um es auf der Busybox-msh-Shell lauffähig zu machen. Es wurde in einem Internetforum gefunden, die Quelle ist nicht mehr bekannt. Die Nutzung wird im Folgenden Shell-Ausschnitt beschrieben:

---

Listing 13: Arbeit mit ldd.sh

---

```
1 28/$curl
2 curl: error while loading shared libraries: cannot open shared object
3   file: cannot load shared object file: No such file or directory
4 29/$
5 29/$ldd.sh 'which curl'
6  libssl.so.0.9.7 => /opt/lib/libssl.so.0.9.7 (0x2aac0000)
7  libcurl.so.4 => /opt/lib/libcurl.so.4 (0x2ab80000)
8  libz.so => /opt/lib/libz.so (0x2ac40000)
9  libc.so.6 => /lib/libc.so.6 (0x2ad00000)
10 libcrypto.so.0.9.7 => /opt/lib/libcrypto.so.0.9.7 (0x2af00000)
11 libdl.so.2 => /lib/libdl.so.2 (0x2b100000)
12 libldap-2.3.so.0 => /opt/lib/libldap-2.3.so.0 (0x2b180000)
13 /lib/ld.so.1 => /lib/ld.so.1 (0x00020000)
14 liblber-2.3.so.0 => not found
15 libresolv.so.2 => /lib/libresolv.so.2 (0x2b240000)
16 libsasl2.so.2 => /opt/lib/libsasl2.so.2 (0x2b2c0000)
17 30/$
18 30/$
19 30/$ls -lh /opt/lib/liblber*
20 -rwxrwxrwx   1 root   root  70.0K Dec 21 03:38 /opt/lib/liblber-2.3.so.0.2.26
21 31/$
22 31/$
23 31/$ln -s /opt/lib/liblber-2.3.so.0.2.26 /opt/lib/liblber-2.3.so.0
24 32/$
25 32/$ldd.sh 'which curl'
26  libssl.so.0.9.7 => /opt/lib/libssl.so.0.9.7 (0x2aac0000)
27  libcurl.so.4 => /opt/lib/libcurl.so.4 (0x2ab80000)
28  libz.so => /opt/lib/libz.so (0x2ac40000)
29  libc.so.6 => /lib/libc.so.6 (0x2ad00000)
30  libcrypto.so.0.9.7 => /opt/lib/libcrypto.so.0.9.7 (0x2af00000)
31  libdl.so.2 => /lib/libdl.so.2 (0x2b100000)
32  libldap-2.3.so.0 => /opt/lib/libldap-2.3.so.0 (0x2b180000)
33  /lib/ld.so.1 => /lib/ld.so.1 (0x00020000)
34  liblber-2.3.so.0 => /opt/lib/liblber-2.3.so.0 (0x2b240000)
35  libresolv.so.2 => /lib/libresolv.so.2 (0x2b2c0000)
36  libsasl2.so.2 => /opt/lib/libsasl2.so.2 (0x2b340000)
37 33/$curl
38 curl: try 'curl --help' or 'curl --manual' for more information
```

---

1. Der Aufruf des Programmes “curl” scheitert aufgrund einer fehlenden Bibliothek.
2. ldd.sh wird aufgerufen. Als Parameter wird der exakte Pfad zum Programm angegeben, unabhängig davon, ob das Programm in einem Ordner der Pfad-Variable liegt oder nicht. Hier wird ‘which curl’ genutzt, welches den exakten Pfad zurückliefert.

3. Die Ausgabe von ldd.sh zeigt alle Bibliotheken mitsamt ihres Pfades an, auf die das Programm zurückgreift. Nicht gefunden wird die Bibliothek mit dem Namen "liblber-2.3.so.0".
4. Ein Verzeichnislisting des Bibliothekspfades wird erstellt. Es sollen alle Dateien angezeigt werden, die mit "liblber" beginnen. Eine Datei wird gefunden, sie heißt ähnlich der gesuchten Datei, besitzt allerdings eine Versionsbezeichnung im Namen.
5. Ein Softlink auf die gefundene Datei wird erstellt, der Name des Softlinks ist der von "curl" gesuchte Dateiname.
6. ldd.sh wird erneut ausgeführt. Diesmal werden alle Bibliotheken gefunden.
7. curl wird testweise gestartet, die Ausgabe zeigt, dass das Programm läuft, allerdings keine Parameter übergeben wurden. Das entspricht den Erwartungen.

### 3.7 Cross-Compiling und Konfiguration von TOR

Zuerst wurden die Quellen von der TOR-Webseite [7] heruntergeladen. Die verwendete Version ist Version 0.2.0.33. Weiterhin wurde die Signatur des Paketes heruntergeladen, um die Echtheit des TOR-Quellcodes zu bestätigen. Hierfür mussten via GPG die PGP-Schlüssel der TOR Entwickler heruntergeladen [8] und importiert werden.<sup>5</sup>

```
gpg -keyserver subkeys.gpg.net --recv-keys 0x28988BF5 0x165733EA 0x31B0974B  
0x94C09C7F 0x5FA14861 0xAFA44BDD
```

Die folgende Ausgabe zeigt den Import des Schlüssels:

Listing 14: GPG-Schlüsselimport für TOR

```
1 gpg: Schlüssel 28988BF5 von hkp Server subkeys.gpg.net anfordern  
2 gpg: Schlüssel 165733EA von hkp Server subkeys.gpg.net anfordern  
3 gpg: Schlüssel 31B0974B von hkp Server subkeys.gpg.net anfordern  
4 gpg: Schlüssel 94C09C7F von hkp Server subkeys.gpg.net anfordern  
5 gpg: Schlüssel 5FA14861 von hkp Server subkeys.gpg.net anfordern  
6 gpg: Schlüssel AFA44BDD von hkp Server subkeys.gpg.net anfordern  
7 gpg: /home/christian/.gnupg/trustdb.gpg: trust-db erzeugt  
8 gpg: Schlüssel 28988BF5: Öffentlicher Schlüssel "Roger Dingledine <arma@mit.edu>"  
importiert  
9 gpg: Schlüssel 165733EA: Öffentlicher Schlüssel "Nick Mathewson <nickm@alum.mit.  
edu>" importiert  
10 gpg: Schlüssel 31B0974B: Öffentlicher Schlüssel "Andrew Lewman (phobos) <  
phobos@rootme.org>" importiert  
11 gpg: Schlüssel 94C09C7F: Öffentlicher Schlüssel "Peter Palfrader" importiert  
12 gpg: Schlüssel 5FA14861: Öffentlicher Schlüssel "Matt Edman <edmanm@rpi.edu>"  
importiert  
13 gpg: Schlüssel 94C09C7F: "Peter Palfrader" nicht geändert  
14 gpg: kein uneingeschränkt vertrauenswürdiger Schlüssel 00000000 gefunden  
15 gpg: Anzahl insgesamt bearbeiteter Schlüssel: 6  
16 gpg: importiert: 5 (RSA: 1)  
17 gpg: unverändert: 1
```

Nach Hinzufügen der Signatur kann mit dem folgenden Befehl die Echtheit des Quellarchivs überprüft werden:

```
gpg tor-0.2.0.33.tar.gz.asc
```

<sup>5</sup>GPG steht für Gnu Privacy Guard und ist die freie Implementierung von PGP (Pretty Good Privacy)

Die folgende Ausgabe zeigt, dass die Unterschrift korrekt ist, aber der Absender nicht vertraulich. Das ist in diesem Fall richtig, da der Absender nicht als vertrauenswürdig eingestuft wurde. Weitere Hintergrundinformationen zur Funktionsweise von GPG / PGP findet sich unter [\[78\]](#)

Listing 15: Ausgabe der Signaturprüfung des TOR-Quelltextes

---

```
1 gpg: Signatur am Mi 21 Jan 2009 19:51:33 CET mit DSA Schlüssel, ID 28988BF5,
  erfolgt
2 gpg: Korrekte Unterschrift von 'Roger Dingledine <arma@mit.edu>'
3 gpg: WARNUNG: Dieser Schlüssel trägt keine vertrauenswürdige Signatur!
4 gpg:      Es gibt keinen Hinweis, daß die Signatur wirklich dem vorgeblichen
  Besitzer gehört.
5 Haupt-Fingerabdruck = B117 2656 DFF9 83C3 042B C699 EB5A 896A 2898 8BF5
```

---

Nun wird das Archiv entpackt und kann mit dem folgenden Kommando über den Freetz-Compiler kompiliert werden. Es wird statisch kompiliert, um Abhängigkeiten von Bibliotheken zu vermeiden. Daher wird dem Linker der “-static”-Parameter übergeben. Mit der “target” und “host” Variable wird dem ./configure-Skript der Cross-Compiler mitgeteilt.

```
export PATH=/media/ubuntu/opt/toolchain/target/bin:$PATH
LDFLAGS=-static ./configure -target=mipsel-linux -host=mipsel-linux
make
```

Hier abgebildet sind die letzten Ausgaben des Compilers. Im Anschluss wird mit “file” geschaut, welcher Dateityp die Zielfeile ist. Das Executable ist statisch, aber noch nicht gestript. Das bedeutet, es stehen noch Debugging-Informationen im Executable, diese können aus Platzgründen mit dem “strip”-Werkzeug entfernt werden. Im Anschluss ist die Datei gestript. Weiterhin in der Ausgabe zu sehen ist die Größenangabe vor und nach dem Strip.

---

Listing 16: TOR-Compiling und Stripping

---

```
1 make[3]: Betrete Verzeichnis '/home/christian/Desktop/tor-0.2.0.33/contrib'
2 make[3]: Für das Ziel »all-am« ist nichts zu tun.
3 make[3]: Verlasse Verzeichnis '/home/christian/Desktop/tor-0.2.0.33/contrib'
4 make[2]: Verlasse Verzeichnis '/home/christian/Desktop/tor-0.2.0.33/contrib'
5 make[2]: Betrete Verzeichnis '/home/christian/Desktop/tor-0.2.0.33'
6 make[2]: Für das Ziel »all-am« ist nichts zu tun.
7 make[2]: Verlasse Verzeichnis '/home/christian/Desktop/tor-0.2.0.33'
8 make[1]: Verlasse Verzeichnis '/home/christian/Desktop/tor-0.2.0.33'
9
10 ~/Desktop/tor-0.2.0.33/src/or$ ls -lh tor
11 -rwxr-xr-x 1 christian christian 4,2M 2009-02-02 22:24 tor
12
13 ~/Desktop/tor-0.2.0.33/src/or$ file tor
14 tor: ELF 32-bit LSB executable, MIPS, MIPS32 version 1 (SYSV), statically linked,
    not stripped
15
16 ~/Desktop/tor-0.2.0.33/src/or$ mipsel-linux-strip tor
17
18 ~/Desktop/tor-0.2.0.33/src/or$ ls -lh tor
19 -rwxr-xr-x 1 christian christian 2,4M 2009-02-02 22:31 tor
20
21 ~/Desktop/tor-0.2.0.33/src/or$ file tor
22 tor: ELF 32-bit LSB executable, MIPS, MIPS32 version 1 (SYSV), statically linked,
    stripped
```

---

Neben der tor-executable wurden weitere Executables erzeugt, die nicht benötigt werden:

- Tor-Test: Ein Testprogramm
- Tor-gencert: Ein Programm zum generieren von Zertifikaten. Diese werden nicht benötigt, wenn TOR ausschließlich als Onionproxy verwendet wird.
- Torify: Ein Wrapper für nicht Socks-kompatible Anwendungen, die nähere Funktion wurde in Kapitel 3.1 beschrieben.

Die Konfiguration von TOR geschieht - wie üblich - über eine Beispielkonfiguration. Sie findet sich unter /src/config/torrc.sample im Quellcodearchiv. Die Konfiguration gestaltet sich sehr einfach, da im Vergleich zu anderer Software nur wenige Einstellungen zu treffen sind. Der überwiegende Teil betrifft die Sockets, auf denen TOR nach Verbindungsanfragen lauscht. Die Erklärungen der Parameter stehen auskommentiert in der Beispiel-Datei. Hier abgedruckt sind lediglich die tatsächlich genutzten Parameter, ergänzt um knappe Kommentare. Die vollständige Version findet sich auf der CD im

Binary Verzeichnis. Die TOR-Konfiguration liegt in `/opt/etc/torrc` und wird TOR über den Parameter `“-f”` mitgegeben.

Listing 17: TOR-Konfiguration `torrc`

---

```
1 # what port to open for local application connections
2 SocksPort 9050
3 # accept connections from everywhere (localhost + localnet; internet is denied
   cause of NAT)
4 SocksListenAddress 0.0.0.0
5 SocksPolicy accept 0.0.0.0/24
6 RunAsDaemon 1
7 DataDirectory /opt/var/tor
```

---

Der folgende Parameter aus der Konfiguration dient dazu, TOR über Programme wie TOR-Button zu steuern. Das problematische: Die Nutzung von einem Client aus würde den TOR-Proxy auch für alle anderen Clients im Netz steuern.

---

```
1 # Controlport for TOR-button
2 ControlListenAddress 127.0.0.1
3 ControlPort 9051
```

---

Das weiß auch TOR, sodass es beim Starten die folgende Meldung ausgibt, sollte man diese Listener IP auf eine Adresse außerhalb des Loopback-Adressbereiches setzen. Interessant ist Zeile 4.

Listing 18: Meldungen beim Starten von TOR auf dem Netcenter

---

```
1 7/$/opt/bin/tor -f /opt/etc/torrc
2 Feb 06 00:05:35.628 [notice] Tor v0.2.0.33 (r18212). This is experimental software
   . Do not rely on it for strong anonymity. (Running on Linux mips)
3 Feb 06 00:05:35.653 [warn] You specified a public address '0.0.0.0' for a SOCKS
   proxy. Other people on the Internet might find your computer and use it as an
   open SOCKS proxy. Please don't allow this unless you have a good reason.
4 Feb 06 00:05:35.658 [warn] You have a ControlListenAddress set to accept
   connections from a non-local address. This means that any program on the
   internet can reconfigure your Tor. That's so bad that I'm closing your
   ControlPort for you.
5 Feb 06 00:05:35.701 [notice] Initialized libevent version 1.3e using method poll.
   Good.
6 Feb 06 00:05:35.718 [notice] Opening Socks listener on 0.0.0.0:9050
```

---

Die Warnung in Zeile 3 ist hingegen insofern nicht korrekt, als das ein Zugriff aufgrund des Masquerading-Service auf dem DSL-Router einen Zugriff aus dem Internet ohne

entsprechende Port-Weiterschaltung unterbindet, sodass tatsächlich nur der Zugriff aus dem lokalen Netzwerk möglich ist.

Damit TOR nach jedem Neustart des Gerätes verfügbar ist, wird der Dienst in die `rc.start` eingetragen. Ein Starten über `Xinetd` ist von den Entwicklern nicht vorgesehen.

```
tor -f /opt/etc/torrc
```

### 3.8 Cross-Compiling und Konfiguration von Privoxy

Der Kompilationsvorgang von Privoxy gestaltet sich schwieriger, da für das Cross-Compiling einige manuelle Eingriffe in die stark automatisierten Compilerskripte von Nöten sind. Auch bei Privoxy wird zuerst der Quellcode über die Website heruntergeladen [53]. Die verwendete Version ist Version 3.0.10. Auch hier wird die Signatur heruntergeladen und die öffentlichen Schlüssel der Entwickler importiert. Da die Entwickler die verwendete PGP-ID nicht preisgeben, wurde im ersten Anlauf die Signatur des Hauptentwicklers Fabian Keil über die persönliche Website [65] ausfindig gemacht und importiert, anschließend verifiziert. Die Signatur erwies sich als gültig:

```
gpg --keyserver subkeys.pgp.net --recv-keys BF2EA563  
gpg tor-0.2.0.33.tar.gz.asc
```

Listing 19: Ausgabe der Signaturprüfung des Privoxy-Quelltextes

---

```
1 ~/Desktop$ gpg privoxy-3.0.10-stable-src.tar.gz.asc  
2 gpg: Signatur am Sa 16 Aug 2008 19:58:52 CEST mit DSA Schlüssel, ID BF2EA563,  
   erfolgt  
3 gpg: Korrekte Unterschrift von 'Fabian Keil <fk@fabiankeil.de>'  
4 gpg: WARNUNG: Dieser Schlüssel trägt keine vertrauenswürdige Signatur!  
5 gpg:      Es gibt keinen Hinweis, daß die Signatur wirklich dem vorgebliehen  
   Besitzer gehört.  
6 Haupt-Fingerabdruck = 8DA1 87F6 B624 9623 B98D 09BF 48C5 521F BF2E A563
```

---

Auch hier wird das Archiv entpackt und ein Compilingversuch mit dem Freetz-Compiler gestartet. Da die `./configure`-Datei erst von `Autoheader` und `Autoconf` erstellt wird, muss zum Ausführen die folgende Zeile in der "Makefile" in einem Texteditor an-

gepasst werden und gleichzeitig die Pfadvariable um den Crosscompiler ergänzt werden:

```
autoheader && autoconf && ./configure -- exit 1;
```

wird ergänzt zu:

```
autoheader && autoconf && LDFLAGS=-static ./configure cross_compiling=yes  
-enable-static -host=mipsel-linux-uclibc CC=mipsel-linux-uclibc-gcc -- exit 1;
```

Das setzen der Pfadvariable erfolgt mit:

```
export PATH=/media/ubuntu/opt/toolchain/target/bin:$PATH
```

Nach Eingabe von “make” kommt nun die Frage, ob Autoheader und Autoconf gestartet werden soll, dies wird mit “y” bestätigt. Nach kurzem Durchlauf von ./configure bricht dieses mit einer Fehlermeldung ab. Laut Aussage einiger Internetforen handelt es sich um einen Bug einiger Autoheader und Autoconf Versionen, der beim Cross-Compiling auftritt. Dieser Bug machte dem Autor auch bei anderen Programme das Leben schwer, lies sich dort allerdings nicht auf die folgende, sehr einfache Methode beheben:

---

Listing 20: Privoxy Autoheader-Dialog

---

```
1 christian@ubuntu:~/Desktop/privoxy-3.0.10-stable$ make  
2 ***  
3 *** To build this program, you must run  
4 *** autoheader && autoconf && ./configure and then run GNU make.  
5 ***  
6 *** Shall I do this for you now? (y/n)
```

---

---

Listing 21: Fehler im Privoxy-Configure Vorgang

---

```
1 checking for bcopy... yes  
2 checking for memmove... yes  
3 checking whether /media/ubuntu/opt/toolchain/target/bin/mipsel-linux-uclibc-gcc  
  needs -traditional... no  
4 checking whether setpgrp takes no argument... configure: error: cannot check  
  setpgrp when cross compiling
```

---

In der von Autoconf erzeugten ./configure-Datei muss die folgende Zeile gesucht und das “yes” durch ein “no” ersetzt werden:

```
if test "$cross_compiling" = yes; then
```

Nun kann der Vorgang mit eingabe von “make” erneut gestartet werden

Die folgende Ausgabe zeigt analog zum TOR-Compiling die Dateigröße und den Dateityp vor und nach dem “striping”, sowie die letzten Ausgaben des Compilers.

---

Listing 22: Privoxy-Compiling und Stripping

---

```
1 mipsel-linux-uclibc-gcc -c -pipe -O2 -I/media/ubuntu/opt/toolchain/target/include
  -pthread -Wall -Ipcr -I/media/ubuntu/opt/toolchain/target/include pcre/
  pcreposix.c -o pcre/pcreposix.o
2 mipsel-linux-uclibc-gcc -static -pthread -o privoxy actions.o cgi.o cgiedit.o
  cgisimple.o deanimate.o encode.o errlog.o filters.o gateway.o jbsockets.o jcc.
  o list.o loadcfg.o loaders.o miscutil.o parsers.o ssplit.o urlmatch.o pcrs.o
  pcre/get.o pcre/maketables.o pcre/study.o pcre/pcre.o pcre/pcreposix.o -lnsl
  -lz
3 grep -v '^#MASTER#' default.action.master > default.action
4 make[1]: Verlasse Verzeichnis '/home/christian/Desktop/privoxy-3.0.10-stable'
5
6 ~/privoxy-3.0.10-stable$ file privoxy
7 privoxy: ELF 32-bit LSB executable, MIPS, MIPS32 version 1 (SYSV), statically
  linked, not stripped
8
9 ~/privoxy-3.0.10-stable$ ls -lh privoxy
10 -rwxr-xr-x 1 christian christian 630K 2009-02-02 23:44 privoxy
11
12 ~/privoxy-3.0.10-stable$ mipsel-linux-strip privoxy
13
14 ~/privoxy-3.0.10-stable$ file privoxy
15 privoxy: ELF 32-bit LSB executable, MIPS, MIPS32 version 1 (SYSV), statically
  linked, stripped
16
17 ~/privoxy-3.0.10-stable$ ls -lh privoxy
18 -rwxr-xr-x 1 christian christian 529K 2009-02-02 23:44 privoxy
```

---

Die Einrichtung von Privoxy gestaltet sich etwas umfangreicher als jene von TOR. Auch hier wird auf eine dem Quelltext beiliegende Muster-Konfiguration zurückgegriffen, die über Sourceforge verfügbar ist: [\[53\]](#)

Die Einrichtung von Privoxy verlangt neben der Konfiguration des Dämons auch die Einrichtung der Filter. Daher gibt es neben der Konfigurationsdatei, welche auf dem Netcenter unter `/opt/etc/privoxy/config` abgelegt wurde, noch die “filter”-Dateien sowie die “action”-Dateien.

Die Konfigurationsdatei enthält - wie bei TOR und vielen anderen Programmen - viele sinnvolle Voreinstellungen, welche mit einem Kommentarzeichen auskommentiert sind und bei Bedarf aktiviert werden können. Die Konfigurationsdatei beschreibt sehr ausführlich (in Englisch) die Auswirkungen der verschiedenen Parameter. Die vollständige Konfigurationsdatei, welche als Konfigurationsvorlage für persönliche Veränderungen genutzt werden kann, befindet sich auf der beiliegenden CD im Binary Ordner. Die Kurzform mit knappen Hilfestellungen findet sich im Folgenden:

Listing 23: Privoxy-Konfiguration

---

```
1 # Konfigurationsverzeichnis für Action-Files etc.
2 confdir /opt/etc/privoxy
3
4 # Logging-Verzeichnis
5 logdir /opt/var/log/privoxy
6
7 # Actionsfiles
8 actionsfile standard # Internal purpose, recommended
9 actionsfile default # Main actions file
10 actionsfile user # User customizations
11
12 # Filter-Files
13 filterfile default.filter
14 filterfile user.filter # User customizations
15
16 # Logging Options
17 debug 8192 # only Errors - *we highly recommended enabling this*
18
19 # Listener
20 listen-address 10.10.10.20:8118
21
22 # Initial state of "toggle" status -> Default-Filter nach Privoxystart "ein"
23 toggle 1
24
25 # Privoxy-Status kann vom Nutzer gesteuert werden (Risiko mit mehreren Nutzern)
26 enable-remote-toggle 0
27 enable-remote-http-toggle 0
28
29 # Nutzer erlauben, die Filtereinstellungen zu setzen
30 enable-edit-actions 0
31
32 # Verfügbarer Arbeitsspeicher für Filter in kb (default 4096)
33 buffer-limit 2048
34
35 # SOCKS-Proxy -> TOR, kein weiterer Versuch im Fehlerfall. Auf Localhost selten.
36 forward-socks4a / 127.0.0.1:9050 .
37 forwarded-connect-retries 0
```

---

Filter-Dateien beinhalten eine Sammlung von verschiedenen Filter, über die Privoxy Webseiten manipulieren kann. Die Dateien enthalten reguläre Ausdrücke, in der Regel mit kurzer Funktionsbeschreibung. Vorgefertigte Filter finden sich im Netz in der Datei “default.filter”. Möchte man eigene Filter entwickeln, bietet sich die Datei “user.filter” an, um bei Erscheinen neuer “default-filter” nicht die eigenen Filter zu überschreiben. Ein Beispielfilter, der in einer html-Seite nach dem Content-Type “text/html” sucht und diesen durch “application/xhtml+xml” ersetzt:

---

Listing 24: Privoxy Beispielfilter

---

```
1 #####
2 #
3 # html-to-xml: Header filter to change the Content-Type from html to xml.
4 #
5 #####
6 FILTER: html-to-xml Header filter to change the Content-Type from html to xml.
7 s@(Content-Type:) text/html(.*)?@$1 application/xhtml+xml$2@
```

---

“Action”-Dateien definieren den Einsatz dieser Filterdateien mit Profilen. Sie können bei entsprechend gesetztem Eintrag in der Konfiguration per Weboberfläche verändert werden, während Filterdateien im Texteditor bearbeitet werden müssen. Auch “Action”-Dateien gibt es mehrere: “default.action”, “standard.action” und “user.action”. Erste enthält ein gutes Basisprofil, zweite drei mitgelieferte Profile unterschiedlicher Aggressivitätsstufen, letztere dient zur Anpassung persönlicher Filterprofile. Nähere Informationen zu den Action- und Filter-Dateien gibt die deutsche Seite von Fabian Keil [66].

Nach erfolgreicher Konfiguration kann Privoxy aufgerufen werden, bei gewünschtem Autostart kann dieser Befehl in die `/opt/etc/rc.start` eingefügt werden:

```
/opt/sbin/privoxy /opt/etc/privoxy/config &
```

---

Listing 25: Meldungen beim Starten von Privoxy auf dem Netcenter

---

```
1 10/$Feb 06 00:17:11.405 Privoxy(00000400) Info: Privoxy version 3.0.10
2 Feb 06 00:17:11.410 Privoxy(00000400) Info: Program name: /opt/sbin/privoxy
3
4 [1]+  Done /opt/sbin/privoxy /opt/etc/privoxy/config
5 10/$
```

---

Um nun zu testen, ob die aktuelle Surf-Sitzung über TOR geleitet wird, reicht ein Aufruf der Heise-Online IP-Seite. Neben der eigenen IP zeigt dieser die Information an, ob diese zu einer TOR-Exitnode gehört. Die tatsächliche vom Provider zugewiesene IP sollte hier nicht auftauchen! [67]



The screenshot shows the 'My-IP-Service' page on heise.de. The breadcrumb trail is 'Netze > Netzwerk-Tools > My-IP-Service'. Below this, it says 'English version: My-IP-Service at heise networks UK.' The main heading is 'My-IP-Service'. Below the heading, it says 'Ihre Anfrage kommt von der IP-Adresse:' followed by a large green box containing the IP address '62.141.58.13'. Below the box, it says 'Diese IP ist eine Tor-Exit-Node!'. At the bottom, there is a note: 'Fragen, Kritik und Anregungen bitte an netze@heise.de. Alle Angaben ohne Gewähr.' The left sidebar contains the heise Netze logo, a login/register section, a search bar with a 'los!' button, and a news section with links for '7-Tage-News', 'News-Archiv', and 'Newsletter'.

Abbildung 14: heise.de/ip zeigt Exitnode

### 3.9 Cross-Compiling: Fehlerbehebung

Die Zeit vom ersten Kompilier-Vorgang bis zum ersten erfolgreichen Durchlauf betrug beim Autor ungefähr einen Arbeitstag. Die Einarbeitung in die Theorie und das erstellen der Compiler-Toolchain ist hier noch nicht mit inbegriffen. Um den Frust beim Leser möglichst gering zu halten, zeigt dieses Kapitel eine Strategie auf, um Fehler einzugrenzen und Ursachenforschung zu betreiben, statt nach dem “Trial and Error”-Verfahren vorzugehen.

1. Readme lesen: Meist finden sich in der Readme-Datei wichtige Informationen darüber, wie ein Standard-Kompilervorgang abläuft und welche Standardparameter übergeben werden. Auch `./configure -help` gibt häufig wichtige Hinweise über mögliche Parameter.
2. Nativ kompilieren: Läuft der Compiler nicht sauber durch, liegt die Fehlerursache am Compiler, einer fehlenden Bibliothek oder an fehlerhaftem Quellcode. Möglicherweise ist der Quellcode für die entsprechende Architektur nicht ausgelegt, weil Teile des Programmes in Assembler mitgeliefert werden und dieser nicht für die gewünschte Architektur verfügbar ist. Läuft der Compiler nicht durch, hilft es, das Programm streng nach Readme-Datei (nativ) zu kompilieren - auch wenn das Programm auf dem lokalen Rechner nicht benötigt wird. Gegebenenfalls

benötigt das Programm zusätzliche Pakete auf der lokalen Maschine, um den Kompilationsvorgang durchzuführen. Beispiele hierfür sind Autoheader und Autoconf, welche für die Generierung des Make-File zuständig sind. Stehen hierzu keine Informationen in der Readme, helfen einschlägige Suchmaschinen oder FAQ-Seiten der jeweiligen Programme. Diese Pakete werden für den Durchlauf des Kompilierungsvorganges benötigt, sie müssen - im Gegensatz zu Bibliotheken - nur für die Build-Architektur vorliegen, auf der der Compiler läuft und nicht zusätzlich für die Host-Architektur, dem Embedded-Device.

3. Fehlende Bibliotheken: Fehler, die beim Linken auftreten, deuten auf fehlende Bibliotheken oder falsche Versionen dieser Bibliotheken hin. Bibliotheken müssen in einer developer-Version vorliegen, auch am “-dev” im Namen erkennbar, damit diese eingebunden werden können. “Lose” Bibliotheken können aufgrund fehlender Header- und Objekt- oder Assembler-Dateien beim Linken nicht eingebunden werden. Fehlende Bibliotheken werden möglicherweise beim Lauf des `./configure`-Skriptes entdeckt, sofern vorhanden. Falsche Versionen können dazu führen, dass der Linker trotz vorhandener Bibliothek mit einer Fehlermeldung abbricht, da die vom Source-Code benötigte Prozedur nicht vorhanden ist. Alternativ wird das Programm kompiliert, lässt sich auf dem Zielgerät jedoch nicht ausführen. Meldet ein Programm auf dem Zielgerät einen Fehler mit dem Inhalt “No such file or directory”, deutet dies ebenfalls auf nicht vorhandene Bibliotheken oder falsche Versionen hin. Gegebenenfalls sind die Bibliotheken in der Toolchain vorhanden, wurden jedoch nicht auf das Zielgerät kopiert. Als Hinweis: Statisches Compiling kann viele Probleme mit Bibliotheken aus dem Weg räumen. Statisches Compiling wird “make” über den Parameter “LDFLAGS=-static” mitgeteilt. Skripte wie “./configure” erwartet die Umgebungsvariable “LDFLAGS=-static“, aufgerufen mit `LDFLAGS=-static ./configure`. Alternativ durch: “-enable-static”. Wichtig ist, zwischen den Compiler-Optionen und den Linker-Optionen zu unterscheiden. “-static” ist ein Parameter für den Linker, nicht für den Compiler. Auch wenn Compiler und Linker in einer Executable zusammengefasst sind, werten diese verschiedene Umgebungsvariablen aus.
4. Pfade prüfen: “Compiler cannot make executables” oder ähnliche Meldungen deuten auf einen defekten Compiler hin. Möglicherweise werden Teile des Compilers, wie zum Beispiels das “Include”-Verzeichnis nicht gefunden, da die Pfadangaben in den Umgebungsvariablen falsch gesetzt sind. Pfadangaben werden getestet, indem

man sie per Copy&Paste mit einem vorangestellten “cd” in eine Shell kopiert und somit versucht, in das Verzeichnis zu wechseln. Leerzeichen und kryptische Sonderzeichen in Pfad- oder Dateinamen (ASCII-Zeichen größer 128) sollten in Pfadangaben vermieden werden. Funktioniert das Wechseln in den Pfad nicht, müssen die Pfadangaben korrigiert werden. Aufmerksamkeit gilt ebenso bei den Umgebungsvariablen “HOST”, “TARGET”, “BUILD”, “ARCH” und “CROSS\_COMPILE”. Zum einen kann eine zu viel gesetzte Variable den Kompilier-Vorgang abbrechen lassen, zum anderen erwarten verschiedene Programme auch unterschiedlich hinterlegte Inhalte. Bei einigen Programmen wird als Inhalt der Variable “CROSS\_COMPILE” der Architektur-Typ erwartet, bei anderen ein “yes”.

Vorsicht ist auch bei Groß- und Kleinschreibung angesagt. “HOST” und “Host” sind verschiedene Variablentypen. Teilweise ist Caps-Lock-Schreibweise, also durchgängig große Buchstaben, teilweise ausschließlich Kleinbuchstaben korrekt.

Der Architekturtyp (der auch manchmal über die Variable CROSS\_COMPILE erwartet wird) erwartet teilweise ein “-” als letztes Zeichen, teilweise nicht. Entsprechend kann dieser Wert “mipsel-linux” oder “mipsel-linux-” sein. Viele Programme nutzen diese Variable um den Compilernamen zu ergänzen. So heißt der GCC im Cross-Compiler-Format bei der überwiegenden Anzahl an Compilern nicht “gcc”, sondern “mipsel-linux-uclibc-gcc” oder “mipsel-linux-gcc”. Die ARCH- oder CROSS\_COMPILE-Variable gibt nun den Prefix des “gcc”-Strings an - allerdings nicht immer!

Weiterhin liefern viele Cross-Compiler zwei Compiler-Versionen mit: Eine dient zum Kompilieren des Kernels, die andere zum Kompilieren den Anwendungen. Freetz liefert nur eine Version, erzeugt aber dennoch Links für beide Compiler-Typen. Eine Verwechslung ist daher zumindest bei Freetz nicht relevant.

Die Pfade der Bibliotheken finden sich teilweise nicht im Unterordner “/libs” der Toolchain, sondern bei einigen Compilern im Ordner “mipsel-linux/libs” - im Gegensatz zum bin-Verzeichnis. Viele Verzeichnisse sind mehrfach vorhanden, in verschiedenen Unterordnern mit unterschiedlichen Inhalten. Dies sorgte auch beim Autor immer wieder für Verwirrung. Auch hier bildet Freetz eine löbliche Ausnahme. Zwar sind dort ebenfalls die oben beschriebenen Unterordner vorhanden, jedoch auch hier wieder als symbolische Links, deren Inhalt jeweils identisch ist.

Wird der Fehler partout nicht gefunden, hilft es unter Umständen, eine neue Shell zu öffnen und die Parameter neu zu setzen. Durch häufiges Umsetzen der Pfade in der Shell ist gegebenenfalls ein Parameter zu viel gesetzt.

5. Compiler testen: Funktioniert der Kompilier-Vorgang ausschließlich nativ, aber nicht für den Cross-Compiler, so könnte eine defekte Toolchain das Problem darstellen. Den Compiler wird am besten mit simplen C-Programmen wie "Hello-World" oder dem "Prime" getestet, funktioniert dieser Schritt, kompiliert man nach und nach komplexere Anwendungen.

Um tiefere Kenntnisse in Bezug auf Compileroptionen zu erhalten, sowie Schwierigkeiten beim Kompilieren zu meistern, gibt Anhang B.1 Hilfestellung beim Kompilieren weiterer Anwendungen.

### 3.10 TOR-Button

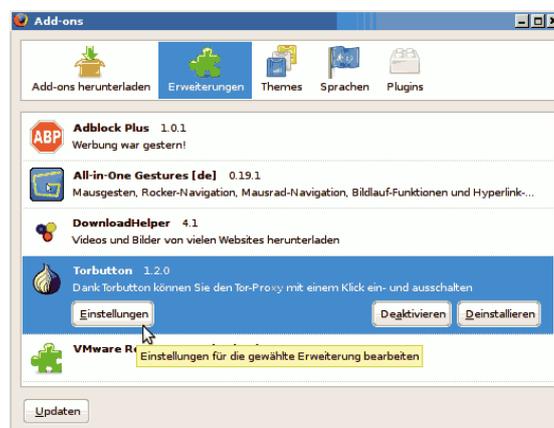


Abbildung 15: Der Firefox Add-On Dialog zum Einstellen von TOR-Button

Wie in Kapitel 3.1 beschrieben, wird von den TOR-Entwicklern die Nutzung von TOR-Button empfohlen [55], um die Umgehung des TOR-Proxys zu verhindern. TOR-Button erhöht die Sicherheit, läuft jedoch dem Ziel dieser Thesis zuwider, einen Proxyserver bereitzustellen, der clientseitig konfigurationsfrei genutzt werden kann. Da der Einsatz aus Sicherheitsgründen dennoch sehr sinnvoll ist, wird hier beschrieben, welche Einstellungen in dem Firefoxplugin zu ändern sind, um auf die entfernte Privoxy/TOR-Kombination

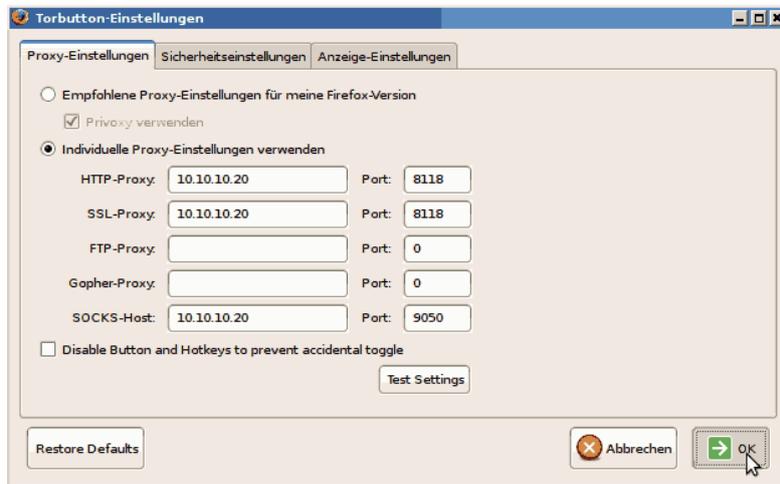


Abbildung 16: Anpassen der TOR-Button Proxy-Einstellung

statt die lokal installierte Version zuzugreifen.

Hierzu wird der Firefox-Add-On-Dialog geöffnet und die Einstellungen von TOR-Button ausgewählt. In dem sich nun öffnenden Einstellungs-Menü wird der http-Proxy manuell angegeben. Hier wird als Adresse die IP des Netcenters eingetragen, der Port von Privoxy wird wie in der Graphik übernommen, sofern die Standardeinstellungen nicht geändert wurden.

## 4 Geschwindigkeitsmessung

Um die Tauglichkeit des Netcenters in Bezug auf die Ladezeit in Zahlen zu fassen, wurden drei verschiedenartige Webseiten auf ihre Ladegeschwindigkeit getestet. Als Webseiten wurden folgende Seiten ausgewählt:

**heise.de** Die Seite stellt eine typische Website in Bezug auf die Größe sowie die Anzahl der Elemente dar, die von fremden Subdomains geladen werden. Viele eingebettete Werbebanner werden von Privoxy herausgefiltert.

**youtube.com** Die Seite youtube.com ist verhältnismäßig komplex, sie beinhaltet viele interaktive Elemente.

**google.de Suche nach "netcenter"** Google.de liefert eine sehr schlichte Internetseite, deren Elemente ausschließlich von der Domain google.de geliefert werden, somit ist die Anzahl der gleichzeitig aufgebauten Verbindungen gering. Die zeitliche Einflussnahme des Suchvorganges auf dem Google-Server ist auf der Suchseite angegeben. Die Zeit belief sich stets auf weit unter einer halben Sekunde und liegt somit um zwei Größenordnungen unter den Ladezeiten über das TOR-Netzwerk.

Für ein genaueres statistisches Mittel wurde jede Seite zweimal aufgerufen, der Browsercache wurde zwischen beiden Ladevorgängen geleert sowie TOR neugestartet, um TOR eine neue verschlüsselte Router-Kette aufbauen zu lassen. Die Anzahl der Testdurchläufe als auch die Anzahl der verschiedenen Testseiten ist gering, allerdings wurde aus Gründen des Aufwandes auf einen komplexeren Test verzichtet. Die sehr kurze Ladezeit ohne eingeschaltetes TOR-Netzwerk zeigt, dass der Einfluss des lokalen Internetaanschluss sowie die Bandbreite des Hostingbetreibers von untergeordneter Bedeutung sind.

Verglichen wurden die folgenden Zugangsmethoden:

- über Privoxy auf lokalem Client
- über Privoxy und TOR auf lokalem Client
- über Privoxy und TOR auf Netcenter

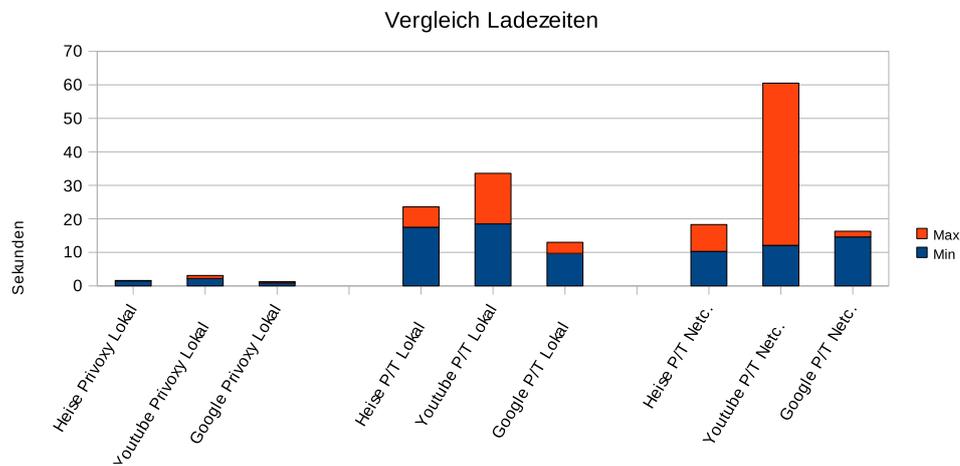


Abbildung 17: Vergleich der Ladezeiten verschiedener Seiten über Privoxy/TOR lokal und auf dem Netcenter

Der direkte Internetzugang ohne Privoxy konnte nicht getestet werden, da die Geschwindigkeitsmessung via Privoxy durchgeführt wurde. Dabei wurde Privoxy über die Konsole gestartet und die Webseite aufgerufen. Privoxy zeigt auf der Konsole Meldungen mitsamt Zeitstempel über die Ladevorgänge an. Die Differenz des Zeitstempels zwischen dem Ladevorgang des letzten und des ersten Elementes wurde als Ladezeit der Webseite interpretiert. Da die Ladezeit mit ausschließlich lokalem Privoxy ohne TOR subjektiv nicht die Zeit des direkten Ladevorganges übertraf und auch bei der im Vergleich großen und komplexen Youtube-Seite bei wenigen Sekunden lag, erhöht sich die Genauigkeit durch Hinzuziehen einer Messung ohne lokalen Privoxy nicht signifikant. Die Konfiguration von Privoxy und TOR ist auf dem Netcenter sowie lokal auf dem Client identisch. Zusätzliche Werbefilter des Browsers wurden für den Test deaktiviert. Die verwendete Privoxy- und-TOR Version unterschied sich auf dem Client zu der im Einsatz befindlichen Version des Netcenters, da auf dem Client die Versionen der Distribution Ubuntu 8.10 verwendet wurden. Auf dem Client kam Privoxy 3.0.8 und TOR 0.2.0.31 zum Einsatz. Auf dem Netcenter kam Privoxy 3.0.10 und TOR 0.2.0.33 zum Einsatz. Von einer Beeinflussung des Messergebnisses wird nicht ausgegangen, da Privoxy nur einen geringen Einfluss auf die Ladegeschwindigkeit hat und es sich bei der TOR Version 0.2.0.33 um eine Bugfixversion handelt, in welcher Sicherheitslücken gestopft wurden, jedoch keine Protokolländerungen durchgeführt oder Funktionen hinzugefügt

wurden.

Die Ladezeit entspricht nicht zwangsläufig der “gefühlten” Ladezeit, da Webseiten progressiv dargestellt werden, d.h. schon heruntergeladene Teile werden dargestellt, bevor die Webseite dem Client komplett bekannt ist. Fehlen nur noch einige Elemente, ist die Webseite bis auf diese wenigen, meist auf anderen Domains gehosteten Elemente bereits vom Browser dargestellt und “nutzbar”. Die fehlenden Elemente werden von Privoxy im Hintergrund heruntergeladen. Dieser Zeitpunkt kann je nach Anzahl der fremden Elemente deutlich vor dem in diesem Test definierten Ladeschluss eintreffen. Ein Auszug aus einer Zeitstempeldarstellung auf der Konsole sieht wie folgt aus.

Listing 26: Log einer TOR-Geschwindigkeitsmessung

---

```
1 10.10.10.13 - - [13/Jan/2009:21:42:42 +0100] "GET http://www.heise.de/ HTTP/1.0"
  200 17208
2 10.10.10.13 - - [13/Jan/2009:21:42:45 +0100] "GET http://www.heise.de/stil/
  standard2008.css HTTP/1.0" 200 0
3 10.10.10.13 - - [13/Jan/2009:21:42:45 +0100] "GET http://www.heise.de/stil/
  navi_top2008.css HTTP/1.0" 200 0
4 10.10.10.13 - - [13/Jan/2009:21:42:45 +0100] "GET http://www.heise.de/favicon.ico
  HTTP/1.0" 200 0
5 10.10.10.13 - - [13/Jan/2009:21:42:45 +0100] "GET http://www.heise.de/stil/ho/
  standard2008.css HTTP/1.0" 200 0
6 10.10.10.13 - - [13/Jan/2009:21:42:45 +0100] "GET http://www.heise.de/stil/drucken
  .css HTTP/1.0" 200 0
7 10.10.10.13 - - [13/Jan/2009:21:42:45 +0100] "GET http://www.heise.de/support/lib/
  external.js HTTP/1.0" 200 0
8 10.10.10.13 - - [13/Jan/2009:21:42:48 +0100] "GET http://www.heise.de/icons/ho/
  background_navi_top.gif HTTP/1.0" 200 0
9 10.10.10.13 - - [13/Jan/2009:21:42:49 +0100] "GET http://www.heise.de/icons/ho/
  heise_online_logo.gif HTTP/1.0" 200 0
10 10.10.10.13 - - [13/Jan/2009:21:42:49 +0100] "GET http://www.heise.de/icons/ho/
  midot.gif HTTP/1.0" 200 0
11 10.10.10.13 - - [13/Jan/2009:21:42:49 +0100] "GET http://www.heise.de/support/lib/
  login_ho.js HTTP/1.0" 200 0
```

---

Es wurde bei der Darstellung der zwei Messergebnisse pro Seite kein arithmetisches Mittel gebildet, da sich die Ladezeiten nicht selten signifikant unterschieden. In mehreren vorhergehenden Testfällen, die keinem dieser Tests zugeordnet werden konnten, war die Verbindung über TOR derart langsam, dass der Aufruf einer Webseite nach 2 Minuten manuell abgebrochen wurde. TOR wurde in diesem Falle zum Erzeugen einer neuen Router-Kette neugestartet. Diese Prozedur ging nicht in die Testwertung mit ein, trat jedoch unabhängig von der Nutzung von TOR auf dem Client oder auf dem Netcenter auf.

Der Test der Google-Seite wurde durch eine Blockade seitens Google aufgrund einer Fehlermeldung über eine “automatisierte Anfrage eines virusinfizierten Computers” gestört. Die Anzeige dieser Meldung scheint mit der gewählten TOR-Exitnode zusammenzuhängen. Wurde statt des Suchergebnisses diese Seite angezeigt, wurde der Test dennoch gewertet, da der Seitenaufbau sich von den erwarteten Suchergebnissen in Bezug auf die dargestellten Elemente nicht grundlegend unterschied. Eine Selektion der Exitnodes, mit denen die Google-Suche anstandslos funktioniert, hätte nach Ansicht des Autors einen ebenso großen Einfluss auf die Statistik wie das Einbeziehen der Fehlerseite in die Wertung. Zudem ist in Bezug auf die Geschwindigkeitsmessung zu beachten, dass Google durchgeführte Suchanfragen an ein für die TOR-Exitnode lokales Google-Portal umleitet. Auch dieses Verhalten kann sich auf die Ladezeit auswirken, wenn die lokale Google-Seite in einem Land mit weniger gut ausgebauter Netzstruktur steht. Auch dieser Einfluss wird als gering eingeschätzt. Davon unberührt bleibt der Einfluss der aus den gleichen Gründen langsameren Exitnodes in eben diesen Ländern - dieser wiederum ist unabhängig von der geladenen Internetseite und wird als einer der Hauptfaktoren für den verhältnismäßig langsamen Verbindungsaufbau über TOR gesehen. Durch diese schwer einschätzbaren Faktoren ist das Ergebnis der Google-Ladezeit mit Vorsicht zu genießen.

Auffällig bei Betrachtung der Graphik ist die um Größenordnungen höhere Ladezeit über TOR, sowie die hohe Streuweite der einzelnen Ladeversuche. Ein Unterschied der Ladezeit zwischen lokalem und auf dem Netcenter stationierten Privoxy und TOR ist hingegen nicht ersichtlich.

Die Zugangsgeschwindigkeiten über TOR, sowie eine gegebenenfalls mögliche Optimierung durch das Auswählen sehr performanter TOR-Nodes könnte sich als ein interessantes Gebiet für eine darauf aufbauende Abschlussarbeit herausstellen.

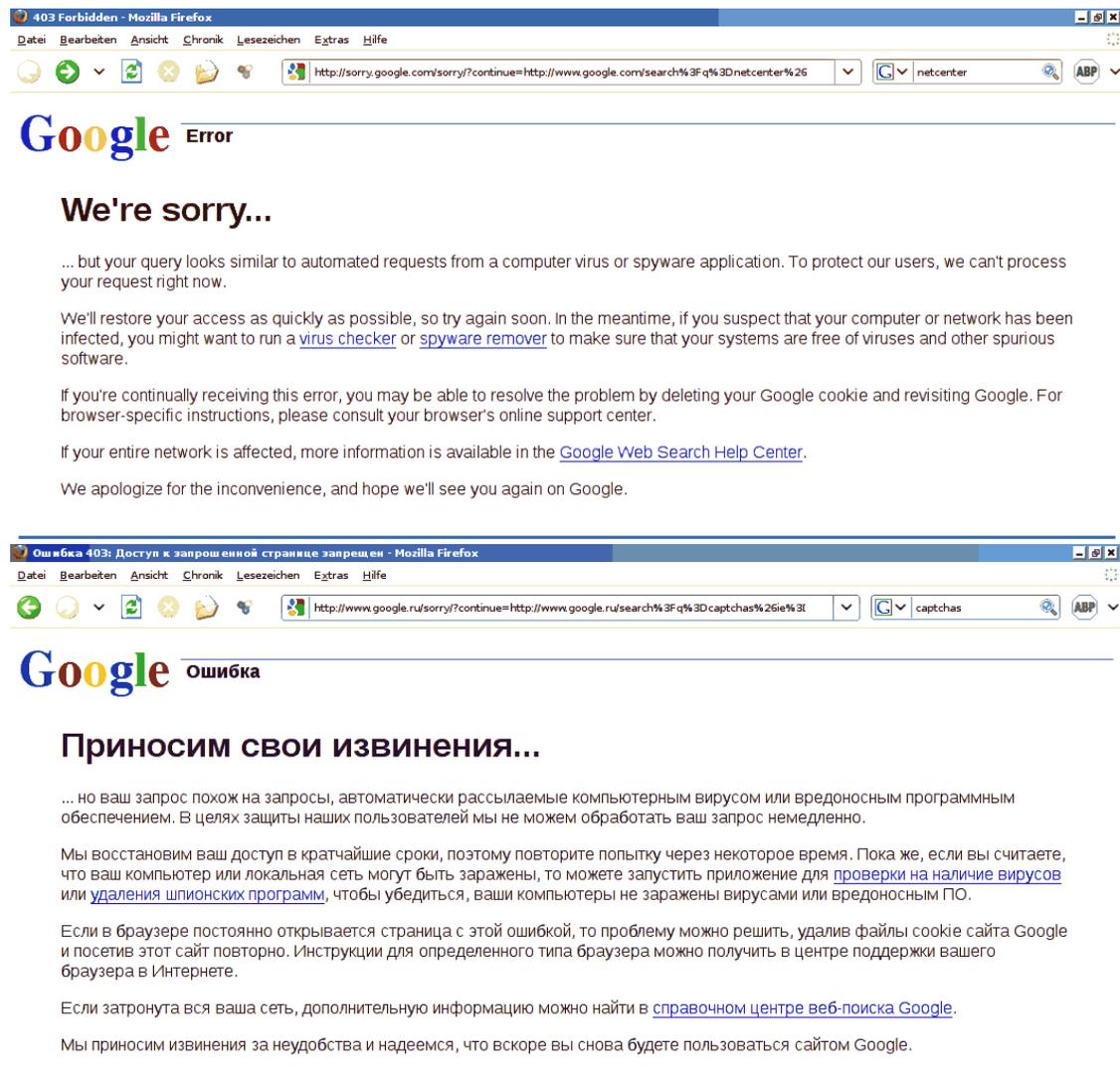


Abbildung 18: Googles Fehlerseite "automatische Anfrage" bei Suche über TOR, je nach Landessprache der Exit-Node

## 5 Fazit

### 5.1 Erreichte Ziele

Das Kernthema der Thesis, die Kompilation, Installation und Konfiguration von TOR und Privoxy stellte nur einen Bruchteil der durchgeführten Arbeit dar. So wurden im Rahmen der Durchführung neben dem Anonymisierungsdienst TOR und Privoxy einige weitere Programme auf das Netcenter portiert.

Die Geschwindigkeit des Anonymisierungsdienstes TOR ist um eine bis zwei Größenordnungen langsamer als das Aufrufen der Seiten ohne einen Anonymisierungsdienst. Daher empfiehlt es sich, das Datenaufkommen über einen Anonymisierer so gering wie möglich zu halten, zum Beispiels mithilfe eines lokalen, zusätzlichen Werbefilters wie Adblock Plus für den Firefox.[\[44\]](#)

Aufgrund der erfolgreichen Portierung und der zusätzlich entstandenen, nützlichen Nebenprodukte war die Arbeit ein großer Erfolg. Durch die Binärkompatibilität mit ähnlichen Geräten wie dem Maxtor-Shared-Storage, dem Linksys WRT54GS oder der Fritzbox können die neu entstandenen Produkte in einer erstaunlich großen Community Verbreitung finden. Die besten als Nebenprodukte entstandenen Features sind hier noch einmal kurz zusammengefasst:

- Der frei konfigurierbare DHCP-Server Dnsmasq auf dem Netcenter ermöglicht feste IP-Adressen auf Basis von MAC-Adressen und somit deutlich vereinfachtes Port-Forwarding über den DSL-Router. Gerade in Entwicklungsumgebungen, wie sie beim Autor vorherrschen, erspart das Arbeit an Netzwerkkonfiguration bei der Nutzung von Live-CDs oder Laptops, die an unterschiedlichen Netzwerken betrieben werden.
- Der FTP-Server eignet sich ideal zum Austausch von Dateien mit Freunden und Bekannten.
- Der Bittorrent-Downloader ist einfach zu nutzen, der Status der Downloads weltweit abrufbar und der Rechner muss während des Downloads nicht eingeschaltet

bleiben. Durch die Bedienung über das Webinterface ist er praxistauglich und auch für weniger geübte PC-Nutzer geeignet.

- Das integrierte Webinterface lässt sich nutzen, um eine private Homepage zu hosten. Alternativ ist der integrierte Webserver schnell genug, um die kleineren unter den freien Webforen wie z.B. FluxBB[45] in akzeptabler Geschwindigkeit zu hosten. So kann ein persönliches, von überall erreichbares Webforum zum Austausch von Informationen bereitgestellt werden, ohne einen Server zuhause oder bei einem Hostler bereithalten zu müssen.
- Zuguterletzt stellt das Netcenter eine vollwertige Linux Kommandozeile bereit. Diese kann genutzt werden, um beispielsweise mit dem integrierten Microperl reguläre Ausdrücke zu parsen oder Batch-Konvertierungen von Dateinamen auf der Festplatte durchzuführen, zum Beispiel um die dort abgelegten Urlaubsbilder in einem Rutsch umzubenennen. Weiterhin lassen sich von dem Gerät andere Rechner per Wake-on-LAN aufwecken.

## 5.2 Blockierte Google-Suche

Bedrohlich wirkt die Tatsache, dass Google Anfragen in geschätzten 60% aller Fälle über TOR als “automatisierte Abfragen” bezeichnet werden und sie daher nicht zulässt. Zeitliche Zusammenhänge lassen den kausalen Schluss zu, dass Anfragen in Bezug auf die genutzte Exitnode blockiert wird. Ob tatsächlich infizierte Clients über die gewählte Exitnode Suchanfragen stellen, die Masse an Abfragen verschiedener Nutzer über diese Exitnode zu dem Fehler führt oder ob Schadsoftware ihren eigenen TOR-Onionproxy mitbringt, kann zum aktuellen Zeitpunkt nicht beantwortet werden.

Die Häufigkeit, mit der Google die Suchanfragen als “schädlich” blockiert, scheint indes sehr hoch. Von einer Nutzung von TOR durch Trojaner wurde bisher nichts bekannt, die tatsächliche Infektion der TOR-Clients scheint auch unwahrscheinlich: So sollte doch gerade ein Nutzer einer Anonymisierungssoftware ein erhöhtes Gespür für Sicherheit und Anonymität mitbringen - und das fängt bei einem saubereren Client-PC an. Das gilt insbesondere dann, wenn der Preis für die Nutzung der Anonymisierung eine deutlich

reduzierte Verbindungsgeschwindigkeit bedeutet. Die Masse der Google-Anfragen über TOR als Grund zu sehen klingt ebenso ungläubwürdig, da die Anzahl der Exitnodes verhältnismäßig gering ist. Selbst das gegenüber Google kleine Redaktionshaus des Heise-Verlages hat sämtliche TOR-Exitnodes gelistet und gibt über ihren IP-Abfragedienst [67] zurück, ob es sich bei der IP-Adresse des Anfragers um eine TOR-Exitnode handelt. Für einen Branchen-Primus wie Google sollte es aus technischer Sicht daher erst Recht kein Problem sein, einer Exitnode eine Sonderrolle bei der Massenabfertigung von Google-Anfragen einzuräumen.

Für den TOR-Benutzer bedeutet das Unterbinden anonymer Suchanfragen die Verteilung zur temporären Deaktivierung von TOR. Wird die gleiche Suche erneut ohne TOR getätigt, lassen sich auf Seiten von Google Zusammenhänge zwischen der Exitnode und der Client-IP knüpfen. Das Verhalten von Google als Suchmaschinenbetreiber wird vom Autor scharf kritisiert und stellt Google als seriösen Anbieter in Frage.

### 5.3 Ausblick

Für zukünftige Erweiterungen bietet das Gerät noch viel Potential. Einige Vorschläge zur Erweiterung:

- Mit der vom Hersteller ausgelieferten als auch mit der in dem Umfang dieser Arbeit erweiterten Firmware hat das Netcenter eine Kapazitätsgrenze von 500 GB. Dies betrifft die Summe des eingebauten, als auch der via USB angeschlossenen Massenspeicher. Es soll einigen Forenteilnehmern gelungen sein, durch ändern des Variablentypes zur Verwaltung der Kapazität im Kernel von signed integer auf unsigned integer die maximal mögliche Kapazität zu verdoppeln. Durch tiefgreifende Änderungen kann diese Grenze unter Umständen auch noch weiter verschoben werden.
- Auf dem Gerät sind ab Hersteller bereits Dienste für eine PPPoE Einwahl vorhanden. Es ist möglich, das Gerät als Ersatz für einen DSL-Router zu verwenden.
- Einige Forennutzer haben auf dem Gerät einen Media Streaming Server installiert,

um die auf der Festplatte gespeicherten Mediendateien auf netzwerkfähigen Mediaplayern wiedergeben zu können. Da sich ein solcher nicht im Besitz des Autors findet, wurde diese Funktion bisher hier nicht implementiert.

- Durch Kompilieren entsprechender Kernelmodule wäre es möglich, zusätzliche Hardware an das Gerät anzuschließen. Denkbar wäre ein Scanner, um diesen mittels SANE und SaneTwain netzwerkfähig und unter Linux und Windows-Clients verfügbar zu machen. Eine zusätzliche USB-Netzwerkkarte böte die Möglichkeit, das Gerät als promiscuous Firewall einzusetzen. Ein WLAN-USB-Stick würde die Möglichkeit eröffnen, das Gerät als WLAN-Access-Point, WLAN-Sniffer oder WLAN-Bridge zu nutzen. Eine angeschlossene Webcam würde das Gerät zu einem netzwerkfähigen Videoüberwachungssystem machen. Günstige Netzwerk-Videoüberwachungskameras liegen preislich zwei Größenordnungen höher.
- Ein passendes Kernelmodul für iptables ließe das Netcenter als Einwahlknoten und Masquerading-Dienst für SSH Verbindungen und somit als sicheren Hafen zur Einwahl aus einem unsicheren Netz fungieren.
- Mit den passenden Bibliotheken ließe sich Truecrypt auf dem Gerät installieren und so der gesamte Festplatteninhalt verschlüsseln. Die Passworteingabe ließe sich über das schon existierende https-Interface realisieren. Die Installation von Truecrypt auf der MIPSel Architektur soll ebenfalls bei einigen Anwendern erfolgreich verlaufen sein, allerdings sank die Schreib- und Lese-Performance auf 250 kb/sek.
- Optional kann der Versuch unternommen werden, OpenWRT auf das Gerät zu portieren und somit den Kernel gegen eine gänzlich freie Alternative auszutauschen.[\[46\]](#) Die in dieser Arbeit kompilierten Anwendungen wären mit dieser Firmware weiterhin nutzbar.
- Viele Dienste laufen bisher unter dem root-Benutzer. Um die Sicherheit im Falle einer Attacke auf einen Dienst zu erhöhen, ist es sinnvoll, jeden Dienst im Kontext eines eingeschränkten Benutzers laufen zu lassen.

## A Softwarekonfiguration des Netcenters

### A.1 xinetd

Da das Netcenter über lediglich 32 MB RAM verfügt, stieg die Nutzung des Arbeitsspeichers mit der Installation diverser Dienste so stark an, dass die Disk-Spindown-Zeit aufgrund der Swap-Vorgänge nie erreicht wurde. Die Festplatte lief daher ununterbrochen. Das ist energietechnisch nicht optimal und, obwohl das Gerät ohne Lüfter auskommt, auch unter Lautstärkeaspekten insbesondere in der Nacht von Nachteil. Ein weiteres Argument, das gegen das dauerhaft eingeschaltete, integrierte Festplattenlaufwerk spricht, ist die Tatsache, dass es sich um ein Consumer-Gerät handelt. Diese werden in der Regel nicht für den dauerhaften Einsatz konstruiert. Gegenüber der nächtlichen Standby-Schaltung ist beim "Dauerlauf" daher von einer niedrigeren Lebenserwartung auszugehen.

Daher wurde versucht, den von den Diensten verwendeten Arbeitsspeicher auf ein Minimum zu reduzieren. Den größten Erfolg versprach die Verwendung von Xinetd. Xinetd lauscht im Netzwerk nach Anfragen und startet die zugehörigen Dienste bei Bedarf. In der Konfigurationsdatei von Xinetd werden Dienste eingetragen, an deren Netzwerkports der Xinet-Dämon lauscht. Erreicht das Netcenter eine Anfrage auf dem Netzwerkport, erstellt der Xinetd einen Fork, d.h. einen Kindsprozess und startet den in der Konfiguration eingetragenen Dienst. Den ein- und ausgehenden Datenstrom leitet Xinetd über Unix-Streams an diesen Dienst weiter.[36] Durch die geänderte Kommunikation des Dienstes über einen Unix-Stream statt über die TCP/UDP Verbindung muss dieser eine Unterstützung für Xinetd mitbringen. Dadurch, dass der entsprechende Dienst mit der ankommenden Verbindung erst gestartet wird, erhöht sich zudem die Antwortzeit des Dienstes beträchtlich. Auch das sollte bei einer Fehlersuche berücksichtigt werden, führte in Bezug auf die durchgeführten Arbeiten während dieser Thesis allerdings zu keinen Problemen. Viele, aber lange nicht alle Dienste bringen Unterstützung für Xinetd mit:

- Dropbear SSH
- Telnetd
- Vsftpd

Ohne Xinetd-Unterstützung kamen die folgenden Dienste einher:

- Samba
- Lighttpd
- Bit Torrent Protocol Dämon (btpd)
- Privoxy
- TOR

Die mächtigsten Dienste, und damit die Dienste, die am meisten Arbeitsspeicher konsumieren, brachten keine Xinetd-Unterstützung mit. Sonderfälle bilden Telnetd und Btpd. Telnetd brachte in der vorinstallierten Busybox-Fassung des Netcenters keinen Xinetd-Support mit. In der nachträglich über ipkg installierten Form war der Telnetd nicht integriert. Erst die manuelle Kompilation der aktuellsten Fassung wurde mit Telnetd und Xinetd-Support erstellt. Der Kompilations- und Installationsvorgang wird in Kapitel 3.4 behandelt. Btpd, der Bit Torrent Protocol Dämon brachte ebenfalls keinen Xinetd-Support mit. Stattdessen wird der Dienst über eine intelligente Skriptsteuerung beendet, wenn kein Torrent in der Warteschlange steht. Auf den Sonderfall Btpd wird im Kapitel A.10 eingegangen. Für Dnsmasq ist die Nutzung von Xinetd nicht gewünscht. Er gehört zu den Diensten mit ohnehin geringem Ressourcenanspruch. Er ist stets im Einsatz, solange einer der Clients eingeschaltet ist und ist auf eine niedrige Latenzzeit angewiesen, die durch den Einsatz von Xinetd negativ beeinflusst würde. Weitere Informationen über die Funktionen von Xinetd finden sich in[36].

Listing 27: Xinetd-Konfiguration

---

```
1 defaults
2 {
3     instances      = 60
4     log_on_success = HOST PID
5     log_on_failure = HOST
6     cps            = 25 30
7 }
8
9 service ftp
10 {
11     disable      = no
12     wait         = no
13     protocol     = tcp
14     user         = root
15     socket_type  = stream
16     only_from    = 0.0.0.0
17     server       = /sbin/vsftpd
18     server_args  = /etc/vsftpd.conf
19 }
20
21 service ssh
22 {
23     disable      = no
24     wait         = no
25     protocol     = tcp
26     user         = root
27     socket_type  = stream
28     only_from    = 0.0.0.0
29     server       = /opt/bin/dropbear
30     server_args  = -i -b /opt/etc/banner_ssh
31 }
32
33 service telnet
34 {
35     disable      = no
36     wait         = no
37     protocol     = tcp
38     user         = root
39     socket_type  = stream
40     only_from    = 0.0.0.0
41     server       = /opt/bin/telnetd
42     server_args  = -i -l /bin/login
43 }
```

---

Die Xinetd.conf unterteilt sich in einen allgemeinen Teil und einen Abschnitt pro verwendetem Dienst. Im allgemeinen Teil befinden sich die folgenden Optionen:

**instances** Gibt die Anzahl von Instanzen an, die Xinetd unterstützt. Dies dient als Schutz, um Denial of Service Angriffe (DoS / DDoS) zu begrenzen.

**cps** Dieser Wert gibt die maximale Anzahl von Verbindungsaufbauten pro Sekunde an. In diesem Falle werden 25 Verbindungsaufbauten erlaubt, wobei der Wert nach 30 Sekunden zurückgesetzt wird.

Im Anschluss befindet sich ein Abschnitt je Service, für den Xinetd Anfragen entgegen nimmt:

**disable = no** aktiviert Xinetd für diesen Dienst.

**wait = no** Sagt aus, dass dieser Dienst multithreaded ist

**protocol** Gibt das verwendete Protokoll des Dienstes an. Weitere Informationen entnimmt Xinetd der /etc/protocols .

**user** Der User, unter welchem der Dienst läuft.

**socket\_type** Gibt die Art des Unix-Sockets an, mit dem Xinetd mit der Anwendung kommuniziert. stream wird üblicherweise bei TCP Netzwerkverbindungen genutzt (verbindungsorientiert), dgram wird bei UDP-Verbindungen genutzt (verbindungslos).

**only\_from** Setzt einen IP-Filter. In diesem Fall ist der Zugriff von überall erlaubt.

**server** Gibt den Pfad zur Executablen des Servers an.

**sever\_args** Gibt die Argumente an, die an den Server übergeben werden.[\[68\]](#)

Folgende Server-Argumente wurden den jeweiligen Diensten übergeben:

**ftp** Konfigurationsdatei

**ssh** Banner-File für den Login-Banner sowie die Option -i für "Interactive", entspricht "nutze Xinetd"

**telnetd** "Interactive" sowie die Loginshell /bin/login zur Loginverifizierung.

Die Portnummern, an den Xinetd für jeden Service lauscht, entnimmt dieser aus der Datei `/etc/services`. Diese Datei listet alle Services mitsamt der zugehörigen Portnummer auf. Daher ist die korrekte Schreibweise des Services wichtig, sie ist kein freier Bezeichner.

Die Dateien `/etc/services` und `/etc/protocols`, auf die Xinetd sowie auch der Vorgänger dieses Services namens `inetd` zurückgreifen, sind im Programm hartkodiert, müssen daher im Verzeichnis `/etc/` liegen. Da der `/etc`-Ordner im Flash des Netcenters liegt und nicht beschreibbar ist, muss vor Nutzung des Xinetd die Firmware entpackt und mit den obigen Dateien erneut zusammengesetzt werden. Nachdem die erweiterte Firmware hochgeladen wurde, startet auch Xinetd. Dieser kann ebenfalls in die `rc.start` eingetragen werden:

```
xinetd -f /opt/etc/xinetd.conf
```

Gibt es nach dem Firmwareupdate stets Probleme mit Xinetd, kann mit dem Debug-Parameter `-d` nach einer Fehlerursache gesucht werden.

## A.2 Firmware entpacken

Zum Entpacken der Firmware bot sich eine Anleitung des Open-Netcenter Forum an[37]. Die Firmware besteht aus 2 Teilen, welche mit einem Tool namens `trx` zu einer Firmware-Datei gepackt wurden. Dieses stammt von den freigegebenen Quellen der Toolchain der Linksys WRT-Router. Eine Wiki-Seite findet sich unter[38], die Downloadquelle für die Toolchain mitsamt den Tools findet sich unter[43].

Der erste Teil der Firmware ist ein mit `gzip` gepackter Kernel namens `piggy.gz`. Der zweite Teil enthält das Root-Filesystem auf Basis von `CramFS`. `CramFS` steht für `Compressed Read Only Memory File System`<sup>6</sup>, es ist ein komprimierendes Dateisystem, das einen direkten Zugriff ohne vorheriges Entpacken in den Arbeitsspeicher erlaubt. Das Dateisystem arbeitet mit 4kb Blöcken, die Kompression erfolgt nicht blockübergreifend. Der Kompressionsgrad ist daher eher bescheiden, sodass es von neueren Dateisystemen

---

<sup>6</sup>Warum dann `CramFS` und nicht `CromFS` bleibt wohl ein Rätsel

wie SquashFS [40] abgelöst wurde, das allerdings von dem in der Firmware integrierten Kernel nicht unterstützt wird[39].

Nun gilt es, die bestehende Firmware zu entpacken. Dazu wird mit dem Unixtool “dd” für “Data Definition” <sup>7</sup> zuerst die piggy.gz und im Anschluss das CramFS aus der Firmware-Datei herauskopiert.

Dazu wird die Firmware, die als Grundlage dienen soll, in einem Hexeditor geöffnet. In diesem Fall wurde die Firmware des Nutzers EPIAS verwendet, die bereits einige Modifikationen beinhaltet, dazu gehört unter anderem ein Telnetzugang. Als Hexeditor kam der Linux Editor Bless[41] zum Einsatz. Der Inhalt der folgenden Beschreibung wurde der oben erwähnten Anleitung des Open-Netcenter Forums entnommen.

Es wird nach einem Hexstring “1F 8B 08 08” gesucht, der in der Nähe des Dateianfangs steht und ein gz-File kennzeichnet, in diesem Falle die piggy.gz.

---

<sup>7</sup> auch bekannt unter “disk dump”

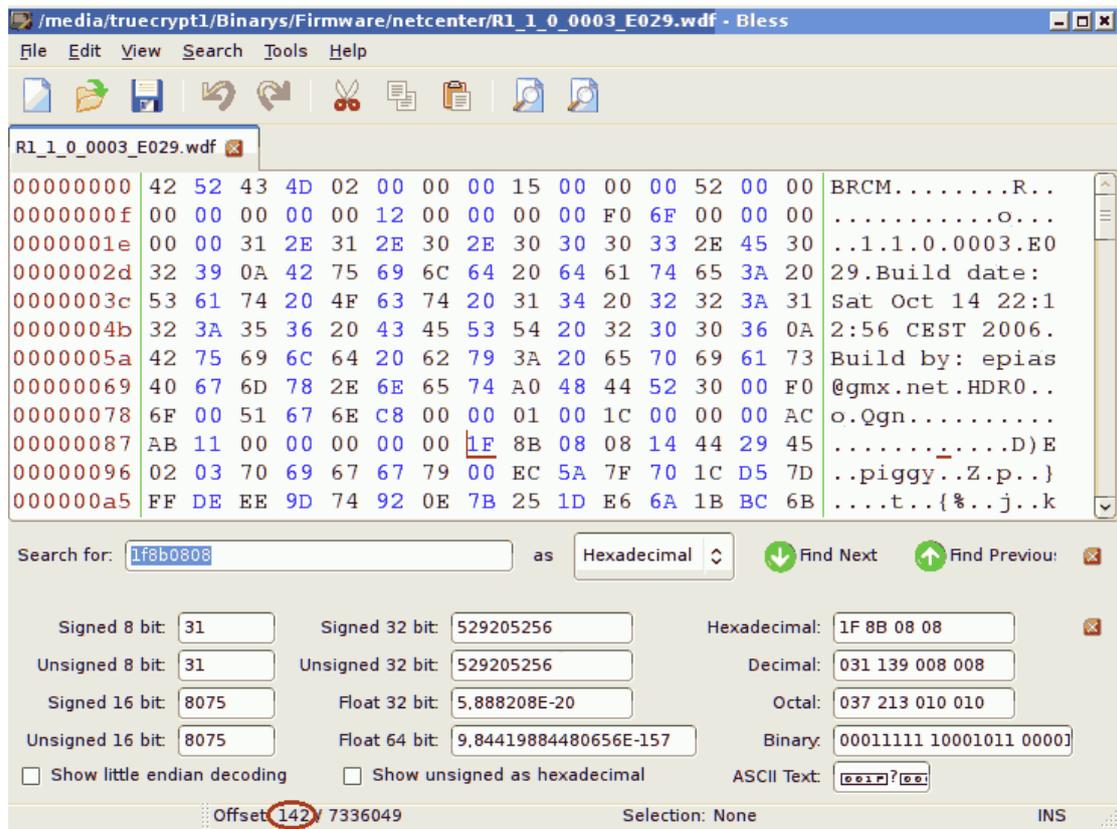


Abbildung 19: Bless: Ablesen des Offset im unteren Bildschirmbereich nach Stringsuche

Wurde der String gefunden, ist im unteren Bereich des Programmes der Offset des ersten Zeichens des Strings abzulesen. Benötigt wird der dezimale Offset. Durch Klicken auf den Offset wechselt die Anzeige zwischen oktal (beginnend mit "0"), hexadezimal (beginnend mit "0x") und dezimal (beginnend mit "1..9"). Dieser Wert wird als "skip"-Parameter für den späteren Entpackungsvorgang mittels dd benötigt, daher wird er notiert.

Als nächstes wird nach dem hexadezimalen String "45 3D CD 28" gesucht, dieser befindet sich etwa am Ende des ersten Drittels der Firmware. Auf den String folgend liest sich in der ASCII-Darstellung der Text "compressed". Man notiert wieder den dezimalen Offset der Anfangsposition des obigen Strings. Er gibt den Beginn des CramFS-Dateisystems an.

Nun wird die Firmware in einzelne Teile zerlegt. Dazu wird dd zweimal ausgeführt, mit

jedem der notierten Werte als skip-Parameter einmal.

```
dd if=firmware.wdf of=piggy.gz bs=1 skip=NUMBER  
dd if=firmware.wdf of=root.cramfs bs=1 skip=NUMBER
```

Der Vorgang dauert aufgrund der Blockgröße von 1 auch auf einem modernen Rechner circa eine Minute, da jedes Byte einzeln kopiert wird und somit die Anzahl der Speicherzugriffe groß ist. Das Resultat sind zwei Dateien. Eine beinhaltet das CramFS-Dateisystem, die andere den gepackten Kernel namens “piggy.gz” mit dem gesamten CramFS-Dateisystem als unerwünschten Anhang. Entfernt wird dieser unerwünschte Anhang durch entpacken und anschließendes packen des gz-Archives. Gzip erkennt das Ende eines Archives automatisch und beendet die Dekompression. Das Packen der Datei erfolgt im Anschluss mit maximaler gzip-Kompression:

```
gunzip piggy.gz; gzip -9 piggy
```

Die erstellte Datei muss exakt 1158029 Bytes groß sein. Die Meldung, dass beim Entpacken der Datei das Ende abgeschnitten wurde, ist das erwartete Resultat der Prozedur. Falls eine Fehlermeldung über ein nicht gültiges gzip-Format erscheint, ist beim Extrahieren der piggy.gz etwas schiefgegangen, in diesem Fall ist der Offset im Hex-Editor noch einmal zu kontrollieren.

Nun wird das CramFS-Dateisystem als Loopback-Device gemountet:

```
sudo mkdir /mnt/cramfs  
sudo mount -o loop,ro ./root.cramfs /mnt/cramfs
```

Die Dateien werden von dort in einen Ordner auf der lokalen Festplatte kopiert. Es ist wichtig, dass das Ziel auf einem Unix-Datenträger liegt, welches symbolische Unix-Verknüpfungen unterstützt. NTFS oder FAT eignen sich nicht als Ziellaufwerk. NTFS unterstützt zwar ebenfalls symbolische Verknüpfungen, scheitert allerdings an denen von Unix.

```
cp -Rp /mnt/cramfs ./root
```

Einige leere Ordner werden bei diesem Prozess nicht kopiert, sie müssen von Hand angelegt werden. Im Anschluss wird das CramFS-Laufwerk wieder aufgehängt.

```
cd root
mkdir foreign_shares proc mnt shares tmp usr/share/empty
sudo umount /mnt/cramfs
```

### A.3 Firmware verändern

In dem root-Dateisystem müssen nun die gewünschten Änderungen für Xinetd durchgeführt werden. Dazu werden die Dateien `/etc/services` und `/etc/protocols` von der eigenen Linuxinstallation in das Rootverzeichnis der Firmware hineinkopiert. Da sich die Services sehr selten ändern, wurde die Entscheidung getroffen, eine aktuelle Datei von einem Desktop-Linux in das CramFS-Dateisystem zu integrieren, statt mittels Softlink auf einen beschreibbaren Teil der Netcenter-Festplatte zu verweisen.

```
cp /etc/protocols ./root/etc/
cp /etc/services ./root/etc/
```

Ebenso wurde die Datei `/etc/shells` erstellt. Diese Datei wird von vielen Programmen als Liste gültiger System-Shells abgefragt. Nur Benutzer, welche eine Shell in der `/etc/passwd` stehen haben, die auch in der `/etc/shells` enthalten ist, werden als gültige Benutzer anerkannt und haben Zugriff auf das System. Als gültige Shells haben sich auf dem Netcenter die folgenden Shells als nützlich herausgestellt:

- `/bin/sh`
- `/bin/bash` oder `/opt/bin/bash`
- `/bin/false`

Die Shell `/bin/false` wird für User genutzt, denen kein Telnetzugriff, jedoch FTP-Zugang gestattet wird. Diese können sich mit `/bin/false` als Login-Shell in der `/etc/passwd` nicht einloggen, werden hingegen korrekt gegen den FTP-Server authentisiert und auto-

risiert.

Da einige Skripte des Netcenters für die Bash geschrieben wurden, wurde die Bash-Executable - wie im Eintrag für die `/etc/shells` zu erkennen - in den Ordner `/bin/` des CramFS kopiert. So wird gewährleistet, dass die im Hintergrund laufenden Skripte keine Festplattenzugriffe durchführen und so die Festplatte daran hindern, in den Standbymodus zu wechseln - oder die Festplatte aus diesem erwecken.<sup>8</sup>

Die Programme `Vsftpd` und `Dnsmasq` wurden durch neuere Versionen ersetzt. Als neue Version für `Dnsmasq` bot sich eine selbst kompilierte Version an. Der Kompilationsvorgang wird in Kapitel B.1 beschrieben. Zudem ist es eine Überlegung wert, weitere häufig benötigte Programme mit in das Flash-Filesystem aufzunehmen. Unter Umständen können einige nicht benötigte Dateien aus dem Flash herausgelöscht werden, um Platz zu schaffen. Das Webinterface des integrierten Webservers lies sich auf die integrierte Festplatte kopieren und dort aus dem Flash via Softlink verlinken. Auf der anderen Seite wurden der `Btpd` und die zugehörigen Skripte in den Flash-Speicher integriert.

Die Busybox-Executable wurde wegen inkompatibler "inssmod"-Funktion nicht durch eine neue Version ersetzt, sondern lediglich durch die neue Version ergänzt. Siehe hierzu auch das Kapitel über die Kompilierung der Busybox 3.4.

#### A.4 Firmware packen und aufspielen

Das Packen der Firmware ist mit der Toolchain des WRT54GS der Firma Linksys möglich. Verwendet wurde die Toolchain `WRT54GS_v4.70.6`, welche der Hersteller Linksys zum Herunterladen anbietet[43]. Zwei Programme des Archivs werden benötigt: Das erste, `mkcramfs`, dient zum Erstellen eines neuen CramFS-Dateisystems aus dem modifizierten System auf der lokalen Festplatte:

```
./WRT54GS_4_70_6_0526_US/release/src/linux/linux/scripts/cramfs/mkcramfs root  
root.cramfs
```

---

<sup>8</sup>Da die Bash mit ca. 1,1 MB ähnlich viel Platz benötigt wie die Busybox mit 1,2 MB, wurde die Busybox Sh-Shell in einer späteren Änderung mit Bash-Erweiterung kompiliert und die Bash-Executable durch die Busybox Sh-Shell ersetzt.

Das zweite Tool dient dazu, aus der `piggy.gz` und der `root.cramfs` eine neue Firmware zu generieren:

```
./WRT54GS_4_70_6_0526-US/release/tools/trx -o myfirmware piggy.gz root.cramfs
```

Dabei ist darauf zu achten, dass die erzeugte Firmware nicht größer als die Kapazität des Flashimages wird. Die Größe der fertigen Firmware ist nicht genau vorhersagbar, da sich verschiedene Dateien unterschiedlich stark komprimieren lassen. Textdateien und Binärdateien lassen sich recht gut komprimieren, bereits komprimierte Bilder hingegen nicht. Softlinks benötigen keinen eigenen 4kb Block und verbrauchen daher nahezu keinen Speicherplatz.

Das Netcenter unterstützt Flash-Images mit Dateigrößen knapp unter 8.000.000 Bytes. Der Upload der Firmware geschieht über das integrierte Webinterface. Wurde dieses zu Gunsten eines eigenen HTTP-Server beendet, kann es über Konsolenzugriff erneut gestartet werden. Hierzu wird eine Verbindung zu dem Gerät via Telnet oder SSH hergestellt. Sollte bereits ein eigenes Webinterface auf Port 80 lauschen, beendet man dieses mit

```
killall lighttpd  
killall php.ld
```

Alternativ kann der Listener-Port des vom Hersteller mitgelieferten Webinterfaces geändert werden. In diesem Fall wurde letzteres durchgeführt.

```
nvrw web_configuration_port=81  
nvrw commit  
httpd
```

Nun kann über das auf Port 81 auffindbare Webinterface die neue Firmware hochgeladen werden. Bleibt die Telnet- oder SSH-Sitzung während dieser Zeit geöffnet, können eventuelle Fehlermeldungen über die Kommandozeile mitverfolgt werden. Diese dort aufgeführten Meldungen geben besseren Aufschluss über die Fehlerursache als das Webinterface.

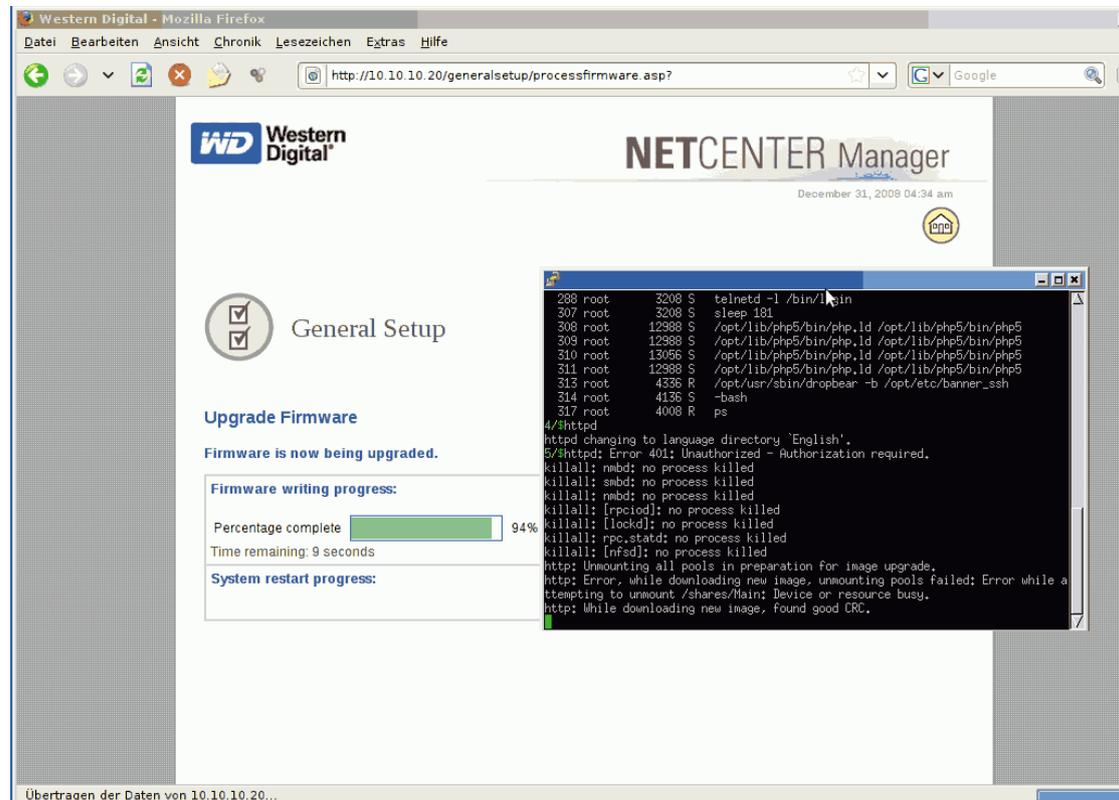


Abbildung 20: Upload der Firmware über das Webinterface

Wurde eine Firmware geflasht, die sich im Nachhinein als nicht lauffähig herausstellt, so besteht die Möglichkeit, diese nach einem Hardreset des Gerätes über den Bootloader durch eine funktionsfähige Firmware zu ersetzen. Unter diesem Aspekt kann das Gerät auch mit Firmwareänderungen nicht beschädigt werden. Der Recoverymechanismus funktioniert über einen TFTP-Client, der nach einem Hardreset ein Firmwareimage im Netzwerk anfragt. Um die Firmware aufzuspielen gibt es ein Tool namens “nasload.exe”, welches auf der CD des Mitbewerberproduktes “Maxtor Shared Storage” beiliegt und unter [69] herunterladbar ist, aber auch der CD zu dieser Thesis beiliegt. Ein Recoveryvorgang lässt sich mit diesem Tool allerdings nur unter Windows bewerkstelligen. Hierzu wird die IP-Adresse des Client-PCs auf die feste Adresse 192.168.1.10 eingestellt, das Tool mittels

```
nasload.exe /U /P /f myfirmware.wdf
```

aufgerufen und im Anschluss der Reset-Button 10 sek lang gedrückt. Nun wird nachgefragt, ob das Image hochgeladen werden soll. Dies bestätigt man durch drücken der Taste “y”. Nach Beendigung des Hochladevorganges dauert es bis zu 10 min, bis das Netcenter nach einem Neustart wieder erreichbar ist. Nach dem ersten Startvorgang müssen Grundeinstellungen, beispielsweise die IP-Adresse, wieder gesetzt werden.

Listing 28: Flashen der Firmware über Nasload.exe

---

```
1 C:\>nasload.exe /f R1_1_0_0003_E029.wdf /U /P
2
3 Attempting to download firmware to local subnet.
4 Use control-c to exit.
5
6 Remote machine WD-NetCenter (at address 192.168.1.1) has requested
7 a firmware download.
8 Do you wish to download new firmware? [y/n]:y
9 Downloading R1_1_0_0003_E029.wdf...
10 Übertragung erfolgreich: 7336050 Bytes in 19 Sekunde(n), 386107 Bytes/s
11 Transfer to WD-NetCenter successful.
```

---

## A.5 FTP-Server Vsftpd

Für den Vsftpd wird neben dem Executable, welches sich bereits im Flash-Speicher befindet, noch die Konfiguration benötigt. Auch hier ist die Beispiel-Konfiguration durch viele Kommentare und Text erläutert. Aus Platzgründen sind hier nur die relevanten Parameter ergänzt durch Kommentare abgebildet. Nähere Informationen sowie eine leere Beispielkonfiguration findet sich unter [70]. Ein besonderes Augenmerk gilt den Parametern “listen” und “check\_shell”. Durch den Einsatz von Xinetd muss “listen” auf “NO” gesetzt werden. “check\_shell” kann dank der Interaktion der /etc/shells auf “yes” gesetzt werden.

Listing 29: Vsftpd-Konfiguration

---

```
1 #Anonymer Zugang wird nicht erlaubt
2 anonymous_enable=NO
3
4 #Lokale User dürfen sich mit Schreibrechten anmelden
5 local_enable=YES
6 write_enable=YES
7
8 #Umask gibt Dateirechte hochgeladener Dateien an
9 local_umask=022
10
11 dirmessage_enable=YES
12
13 #Transfer wird geloggt
14 xferlog_enable=YES
15
16 #Setzt FTP-Data Port auf Port 20
17 connect_from_port_20=YES
18
19 #FTP-Welcome Message
20 ftpd_banner>Welcome to the WD-NetCenter FTP service.
21
22 #Erlaubt Login nur, wenn gültige Shell aus /etc/shells des Users in der /etc/
    passwd steht
23 check_shell=YES
24
25 #NO wenn xinetd genutzt wird, YES wenn direkt auf Port 21 gelauscht wird
26 listen=NO
27
28 #Change-Root Umgebungseinstellungen
29 secure_chroot_dir=/
30 chroot_local_user=YES
31 passwd_chroot_enable=YES
32
33 #Zeiteinstellungen
34 use_localtime=YES
```

---

## A.6 SSH Zugriff - Dropbear

Der folgende Abschnitt wurde durch Neukompilation von Dropbear im Kapitel [B.2](#) obsolet. Er befindet lediglich aufgrund einiger Erkenntnisse über die Arbeitsweise von Dropbear in dieser Thesis.

Für den sicheren Shellzugriff kommt Dropbear als ssh-Server zum Einsatz. Verwendet

wurde eine Version, die via ipkg verfügbar ist. Diese wurde mit in das Flash kopiert und wird mittels Xinetd gestartet. Die Startparameter sind daher im Abschnitt über die Xinetd-Konfiguration [A.1](#) zu finden.

Beim Einloggen via SSH nimmt das Warten auf den Login-Prompt teilweise sehr viel Zeit in Anspruch. Dropbear nutzt als Quelle für Zufallszahlen in der über ipkg verfügbaren Version 0.48 das Unix-Device `/dev/random`, dieses wiederum generiert die Zufallszahlen aus Aktivitäten, die auf dem Netcenter stattfinden. Befinden sich alle Prozesse des Netcenters im Idle-Zustand, werden von `/dev/random` keine oder nur ungenügend Zufallszahlen generiert, um einen Login-Prompt anzuzeigen oder die Autorisierung durchzuführen.

Mit dem nachfolgenden Skript wurde die Geschwindigkeit des `/dev/random` und des `/dev/urandom` verglichen.

Listing 30: Geschwindigkeitsvergleichen zwischen `/dev/urandom` und `/dev/random`

```
1 $cat /dev/urandom > /tmp/test & sleep 10; killall cat
2 $ls -lh /tmp/test
3 -rw-r--r--  1 root  root  2.5M Jan 20 14:52 /tmp/test
4
5 $cat /dev/random > /tmp/test & sleep 10; killall cat
6 $ls -lh /tmp/test
7 -rw-r--r--  1 root  root  129 Jan 20 14:52 /tmp/test
```

---

Nach 10 Sekunden ergibt sich, wie oben ersichtlich, der folgende Durchsatz:

**`/dev/random`** 2,5 Megabyte = 250 kb/sek

**`/dev/urandom`** 129 Byte = 13 Byte/sek

Um die Anmeldung auch bei geringer Aktivität stets durchführen zu können, gibt es zwei Möglichkeiten:

1. Aktivitäten auf dem Netcenter erzeugen, indem beispielsweise das Webinterface parallel zum Login-Vorgang abgerufen wird.
2. Den Dropbear Server auf das `/dev/urandom` Device statt des `/dev/random` Device verweisen.

Die Entscheidung fiel hier auf den letzteren Vorgang. Das geht zwar mit verminderter Sicherheit einher, jedoch ist zweifelhaft, ob das /dev/random-Devices höhere Sicherheit bietet, wenn die Zufallszahlen dieses Devices durch immer den gleichen Vorgang - das Abrufen des Webinterfaces - erzeugt werden. Wird wegen der geringen Geschwindigkeit aus Bequemlichkeit stattdessen auf Telnet ausgewichen, muss hier nicht näher erläutert werden, warum /dev/urandom doch als Alternative in Betracht gezogen wurde.

Um Dropbear nicht erneut mit geänderten Einstellungen kompilieren zu müssen, wurde auf einen Hexeditor-Hack ausgewichen. Das erneute Kompilieren wird als Future-Todo in Aussicht gestellt. Für den Hack wird Dropbear in dem vom Autor favorisierten Hexeditor Bless geöffnet, der String “/dev/random” gesucht und durch “/dev/urandom” ersetzt. Da es sich bei dem Programm um ein Executable mit Einsprungadressen handelt, darf das Programm keinesfalls um ein Byte “länger” werden, daher muss der nachfolgende String um den zusätzlichen Buchstaben gekürzt werden. Dies geschah durch das Ändern von “error” in “err.”. Achtung: Bei dem in “err.” verwendete Punkt handelt es sich um den ASCII-“Punkt” (0x2E), bei dem in Bless als Punkt dargestellten Zeichen zwischen “urandom” und “err.” handelt es sich um einen Platzhalter für die ASCII-Doppelnul (0x00), die das Ende des vorhergehenden Strings anzeigt.

Nach dem Speichern kann die Datei auf das Netcenter zurückkopiert oder alternativ in das Flash integriert werden.

## A.7 Bashrc

Über die Bashrc ist es möglich, das Aussehen und Verhalten der Shell zu beeinflussen. Auf dem Netcenter wird diese Datei genutzt, um den Prompt PS1, einige weitere Variablen, als auch eigene Alias-Befehle zu implementieren. Die Variablen werden exportiert, sodass sie in jeder Subshell vorhanden sind.

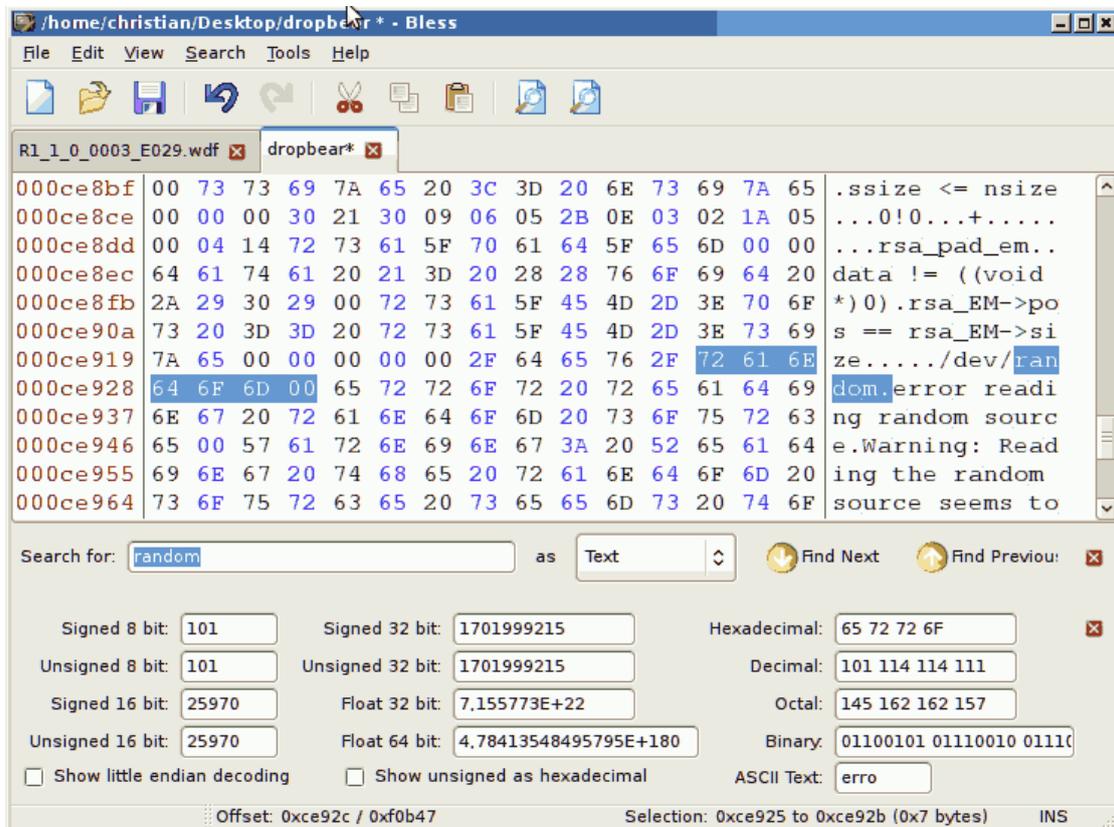


Abbildung 21: Stringsuche nach /dev/random

### Listing 31: /etc/profile

```
1 PS1="\[\033[36m\]\#\[\033[33m\]\w\[\033[32m\]\$\[\033[0m\]"
2 PATH=/sbin:/bin:/usr/sbin:/usr/bin:/opt/sbin:/opt/bin
3 LD_LIBRARY_PATH=/opt/lib:${LD_LIBRARY_PATH}
4 TZ=CET-1CEST
5
6 alias ll='ls -lh'
7 alias dir='ls -lh'
8 alias cd.='cd .'
9 alias mk='mkdir'
10 alias rd='rm -r'
11 alias bin='cd /opt/bin'
12 alias etc='cd /opt/etc'
13 alias opt='cd /opt'
14 alias ipkg='ipkg-cl -f /opt/etc/ipkg.conf'
15 alias apt-get='ipkg-cl -f /opt/etc/i.conf'
16
17
18 export PS1 PATH LD_LIBRARY_PATH TZ
```

```
| 2A 29 30 29 00 72 73 61 5F 45 4D 2D 3E 70 6F | *)U).rsa_EM->po  
| 73 20 3D 3D 20 72 73 61 5F 45 4D 2D 3E 73 69 | s == rsa_EM->si  
| 7A 65 00 00 00 00 00 2F 64 65 76 2F 75 72 61 | ze...../dev/ura  
| 6E 64 6F 6D 00 65 72 72 2E 20 72 65 61 64 69 | ndom.err. readi  
| 6E 67 20 72 61 6E 64 6F 6D 20 73 6F 75 72 63 | ng random sourc
```

Abbildung 22: /dev/random durch /dev/urandom ersetzen und Verkürzen des nachfolgenden Strings

Die /etc/profile wird ausschließlich von Login-Shells eingelesen, nicht jedoch von aufgerufenen “interactive” Subshells wie z.B. dem Programm “screen”. Da die Variablen über die Export-Funktionen auch den Subshells zugänglich gemacht werden, betrifft dies nur die Alias-Befehle. Diese sind zwar praktisch, aber nicht unbedingt erforderlich.

Die Reihenfolge, in welcher die Bash als Login-Shell Start-Dateien durchsucht, ist folgende:

- /etc/profile
- ~/.bash\_profile
- ~/.bash\_login
- ~/.profile

Erste und letztere werden dabei auch von nicht-Bash-Shells durchsucht. Als interactive-Shell wird von der Bash ausschließlich die folgende Datei durchsucht:

- ~/.bashrc

Mit dem in der /etc/profile gewählten, interaktiven Prompt kommen einige Shells nicht zurecht. Der Prompt sieht nur unschön aus, richtet aber keinen Schaden an. Da andere Shells außer die Bash ohnehin nur für Skripte verwendet werden, ist dieser “Bug” nicht relevant.

## A.8 IP-Updateskript

Eine weiterhin nützliche Funktion ist die ständige Erreichbarkeit des Netcenters über das Internet. Die nach außen sichtbare IP-Adresse eines Internet-Service-Providers ist bei einem DSL-Anschluss in der Regel nur bis zur nächsten Einwahl via PPPoE gültig. In der Regel erfolgt daher durch die providerseitige Zwangstrennung nach 24 Stunden ein Wechsel der IP-Adresse. Um einen Rechner an einer DSL-Verbindung dennoch erreichen zu können, gibt es dynamische IP-Adressdienste. Diese haben sich darauf spezialisiert, Subdomains ihrer eigenen Domain zu vergeben und die im DNS hinterlegte IP-Adresse via http-Request aktualisieren zu lassen. Viele DSL-Router sind in der Lage, diese http-Request zu den größten DynDNS-Provider absetzen zu können und somit auch bei einer variablen DSL-IP-Adresse die Erreichbarkeit unter einem immergleichen Namen zu ermöglichen. Einige dieser Provider bieten diesen Dienst für Privatkunden kostenlos an, einer dieser Dienste ist "dyndns.org" [71]. Da der DSL-Router des Autors nicht in der Lage ist, einen der kostenlosen Dienste zu aktualisieren, bietet es sich an, die Aktualisierung vom Netcenter durchführen zu lassen. Gleichzeitig kann auf der Festplatte ein Log der vergebenen IP-Adressen geschrieben werden.

Der IP-Check sollte in Abständen laufen, die klein genug sind, um eine stetige Verfügbarkeit zu gewähren und groß genug sind, um nicht unnötig Ressourcen auf dem eigenen Gerät und Serverlast beim abgefragten Dienst zu erzeugen. Als Zeitabschnitt haben sich hier 3 min bewährt, die eine Wiedererreichbarkeit statistisch nach 90 Sekunden ermöglichen. Weiterhin wurde eine Funktion implementiert, die dafür sorgt, dass die Internetverbindung um 5 Uhr nachts getrennt wird, um eine providerseitige Zwangstrennung über Tage zu verhindern. Die Trennung um 5 Uhr nachts wird dabei nur durchgeführt, wenn gerade kein Download via Torrentdienst durchgeführt wird.<sup>9</sup>

Eine Alternative gegenüber einem Skript mit Endlosschleife stellt der Aufruf via Cron-Dämon dar. Cron ist ein Dämon, der jede Minute aufwacht und in einer Konfigurationsdatei namens Crontab nach Programmen ausschaut, die ausgeführt werden müssen. Zwei Gründe sprechen gegen die Ausführung mittels Cron: Zum einen ist es

---

<sup>9</sup>Der vom Autor verwendete DSL-Provider Arcor hat - wahrscheinlich aufgrund der Vorratsdatenspeicherung - die 24-stündige Zwangstrennung Ende Januar 2009 abgeschafft. Das Log des Netcenters sowie diverse Internetforen bestätigen dies, eine offizielle Stellungnahme von Seiten des ISP liegen nicht vor. Die nächtliche Verbindungstrennung zur Verschiebung der Zwangstrennung in die späte Nacht wurde daher nachträglich entfernt.

bisher nicht gelungen, Cron auf dem Netcenter zum Laufen zu bringen. Zwar lies sich der Dämon starten, allerdings führte er keinerlei Programme aus der crontab aus, obwohl diese entsprechend der Manpage [72] konfiguriert war. Andere Nutzer des Netcenter und des Maxtor Shared Storage klagten über ähnliche Probleme. Der andere Grund ist die geänderte Status-Verwaltung des Internet-Zuganges und der letzten IP-Adresse bei Starten über Cron. Daher wäre das Skript für Cron von Grund auf neu zu schreiben, ohne das sich Verbesserungen hieraus ergäben.

Eine weitere Funktion des Skriptes stellt das Setzen der Leuchtdiode am Netcenter dar. Das Netcenter besitzt zwei unmittelbar nebeneinander positionierte Leuchtdioden, eine in amberfarben und eine blaue. Leuchten beide, sieht der Betrachter eine violette Diode. Während der Zustand der amberfarbenen Diode frei gewählt werden kann, leuchtet die blaue Diode nach dem Start des Netcenters ununterbrochen. Zwar kann sie analog zur amberfarbenen Diode über eine Device-Datei verändert werden, der Status springt jedoch innerhalb einer Sekunde zurück auf "an". Es scheint, als setzt ein weiterer Dienst den Status dieser Diode. Da die Leuchtintensität mit der Festplattenaktivität schwankt, scheint ein Kernelmodul hierfür zuständig.

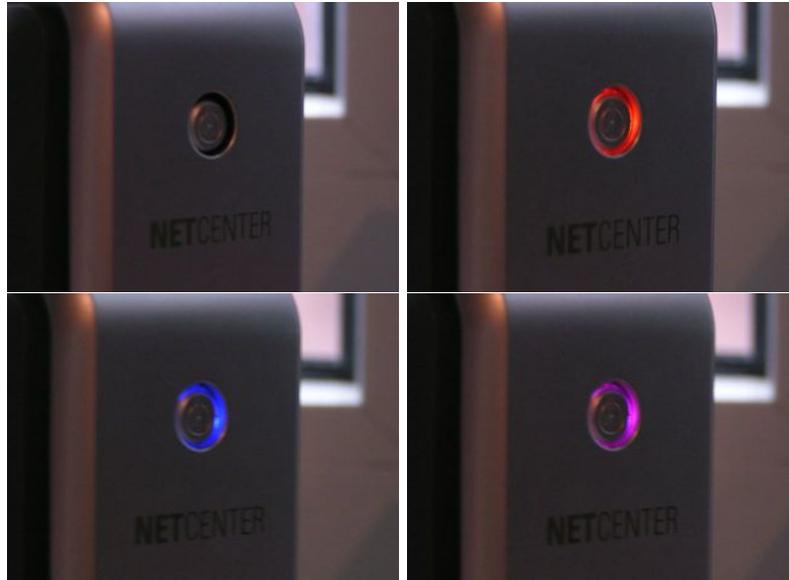


Abbildung 23: Zustände der Statusleuchte an der Gerätefront

Die amberfarbene Diode wird über das Device “/proc/miscio/gpio\_4” angesprochen, gesetzt wird es über ein echo zusammen mit der Ausgabeumleitung:

```
echo 1 > /proc/miscio/gpio_4
```

Dabei kann die amberfarbene Diode die folgenden Zustände annehmen. Hinter den Zuständen sind die im ipupdate-Skript verwendeten Funktionen notiert.

- 0** ein = Internetverbindung aktiv, Torrentdienst aktiv
- 1** aus = Internetverbindung aktiv
- 2** blinkend = Internetverbindung inaktiv, keine Aussage über den Torrentdienst

Die Anzahl der aktiven Torrentdienste wird über einen Befehl bestimmt, der den Prozessbaum über “ps” erfragt, mittels grep nach “torrent” sucht und im Anschluss den eigenen grep-Suchprozess herausfiltert. Die zurückgelieferten Zeilen werden mittels “wc -l” gezählt. Ist dieser Wert “0”, wurde kein Torrent-Prozess gefunden.

Die weiteren Funktionsweisen können denn inline-Kommentaren des Skriptes entnommen werden.

---

Listing 32: ipupdate.sh

---

```
1 #!/bin/bash
2
3 #Parameter-Settings
4 state=running
5 TIMEOUT=180
6 logfile=/opt/etc/iphistory.log
7 hostname=example.dyndns.org
8 username=example
9 password=password
10
11 #Hole letzte IP und letzte Aktualisierung aus der Log-File
12 IPalt=$(sed -e '$!d' -e 's|.*IP=||' ${logfile})
13
14 #Lese Torrentstatus aus und setze Statusleuchte
15 if [[ "'ps | grep torrent | wc -l'" -eq "1" ]]; then
16
17   #Kein Torrentprozess aktiv, setze Statusleuchte auf "aus"
18   echo 1 > /proc/miscio/gpio_4
```

```
19
20 #Torrentprozess aktiv, setze Statusleuchte auf "ein"
21 else
22 echo 0 > /proc/miscio/gpio_4
23 fi
24
25 #Endlosschleife
26 while [[ 1 ]] #Endless loop.
27 do
28
29 #Aktuelle IP abfragen
30 IP=$(wget -q -O - http://checkip.dyndns.org/ | /bin/sed -e 's|.*Address: ||; s
    |</body>.*||')
31
32 #Falls IP sich geändert hat...
33 if [[ "$IPalt" != "$IP" ]]; then
34     timestamp='date +%Y-%m-%d~%R'
35
36     #...und neue IP ist 'leer'...
37     if [[ "$IP" == "" ]]; then
38
39         #...setze Statusleuchte -> blinken, prüfe alle 10 Sekunden auf neue IP...
40         echo 2 > /proc/miscio/gpio_4
41         TIMEOUT=10
42
43         #... prüfe ob der DSL-Router noch lebt
44         if [[ "'ping -c 1 10.10.10.10 | sed -e '2!d'" == "" ]]; then
45
46             #...wenn nicht, kann man ohnehin nichts tun, vermerke dies in der Logfile
47             if [[ "$state" != "death" ]]; then
48                 echo $timestamp keine Internetverbindung, Router tot >> ${logfile}
49                 state=death
50             fi
51         else
52             #...wenn ja, neustarten (bisher wg. Nebenwirkung nicht implementiert)
53             if [[ "$state" != "alive" ]]; then
54                 echo $timestamp keine Internetverbindung, Router lebt >> ${logfile}
55                 state=alive
56             fi
57         fi
58     fi
59
60     # Wenn neue IP eine gültige IP ist, schreibe IP in Logfile
61     if [[ "$IP" != "" ]]; then
62         IPalt=$IP
63         echo $timestamp IP=$IP >> ${logfile}
64         state=running
65         TIMEOUT=180
66
67         #...sende die neue IP an DynDNS
68         wget -q http://${username}:${password}@members.dyndns.org/nic/update?hostname=
```

```
        ${hostname} &
69
70     #Aufgrund des Logwrites ist Platte aktiv -> führe sonstige Dienste aus
71
72     #Überschreibe passwd mit /opt/etc/passwd (USB-Bug)
73     /opt/bin/usbreset.sh &
74     fi
75 fi
76
77 #Reconnect um 05:00 Uhr, wenn keine Torrents laufen
78 #Andernfalls lösche alle fertigen Torrents und beende ggf. den Dienst
79
80 if [[ "$state" == "running" ]]; then
81     if [[ "'ps | grep torrent | wc -l'" -eq "1" ]]; then
82         if [[ "'date +%R'" =~ "04:56" ]]; then sleep 230; /opt/bin/reconnect.sh; fi
83         if [[ "'date +%R'" =~ "04:57" ]]; then sleep 170; /opt/bin/reconnect.sh; fi
84         if [[ "'date +%R'" =~ "04:58" ]]; then sleep 110; /opt/bin/reconnect.sh; fi
85         if [[ "'date +%R'" =~ "04:59" ]]; then sleep 50; /opt/bin/reconnect.sh; fi
86     else
87         btkill christian
88     fi
89
90     #Status-Laempchen setzen
91     #Setze auf "an", wenn Torrents laufen, ansonsten setze es auf "aus"
92     #Nur gültig -> Internetverbindung auf "running"
93     #"blinken" = "Kein Internet" -> höhere Priorität als Torrent-Status
94     if [[ "'ps | grep torrent | grep -v grep | wc -l'" -eq "0" ]]; then
95         echo 1 > /proc/miscio/gpio_4
96     else
97         echo 0 > /proc/miscio/gpio_4
98     fi
99 fi
100
101 sleep $TIMEOUT
102 done
```

---

## A.9 SSLwrap

Nachdem der verschlüsselte Zugang über die Kommandozeile mittels ssh realisiert wurde, fehlt der verschlüsselte Zugang über das Webinterface. Da das aus den Paketquellen bezogene Lighttpd keine SSL-Unterstützung mitbringt, wurde SSL via SSLwrap nachgerüstet. SSLwrap ist ein Service, der einen bestehenden Dienst um SSL-Funktionalität erweitert, indem er von außen verschlüsselte Anfragen entgegen nimmt und diese per Loopback-IP-Device an einen lokalen Dienst weiterleitet. Nicht alle Protokolle lassen

sich “wrappen”, ohne Unterstützung kommen beispielsweise ftp und sämtliche Dienste, die auf das UDP-Protokoll aufsetzen.

SSLwrap wurde über die Paketquellen installiert mittels

```
ipkg-cl -f /opt/etc/ipkg.conf install SSLwrap
```

Damit SSLwrap die Anfragen SSL-verschlüsseln kann, wird zusätzlich ein Zertifikat benötigt. Dieses wurde mittels Openssl erstellt. Die Erstellung erfolgte aus Geschwindigkeits- und Sicherheitsgründen auf dem Linux-Hostsystem. Es wird angenommen, dass Sicherheitsrisiken quelloffener Verschlüsselungssoftware mit ihrer Einsatzhäufigkeit abnehmen und daher Sicherheitsrisiken wie das Debian-SSL-Debakel [73] auf Embedded-Devices unter Umständen nicht entdeckt werden. Zumal nicht nachvollzogen werden kann, wer Openssl für das Netcenter kompiliert hat.

Das Zertifikat muss zusätzlich von einer Zertifizierungsstelle zertifiziert werden. Im Falle von ausschließlich selbst genutzten Diensten ist der Einsatz eines kommerziellen Zertifikates, das ab etwa 50 Euro erhältlich, nicht wirtschaftlich sinnvoll. Stattdessen kann entweder ein kostenloses Zertifikat des auf Web-of-Trust basierenden CA-Cert [74] genutzt werden, oder man nutzt ein selbst signiertes Zertifikat. Dieses liefert zwar in aktuellen Browserversionen beim ersten Aufruf eine Fehlermeldung zurück, nach einem Zertifikateimport ist es jedoch nutzbar und sicherheitstechnisch unbedenklich, denn sowohl der Dienstebetreiber als auch der Zertifikateaussteller sind bekannt. Daher kommt dieses im Falle de Netcenters zum Einsatz.

Das Erstellen des Zertifikates geschieht in zwei Schritten. Im ersten Schritt wird ein Zertifikat für die eigene Certification Authority erstellt. Mit diesem Stammzertifikat wird dann später das eigene Webserverzertifikat signiert.

```
openssl genrsa -out ca.key 1024  
openssl req -new -key ca.key -out ca.csr  
openssl x509 -days 1095 -signkey ca.key -in ca.csr -req -out ca.crt
```

Die erste Zeile generiert ein RSA-Schlüsselpaar, die zweite Zeile bittet uns, die nötigen,

persönlichen Daten der eigenen Certification Authority einzugeben und die dritte erstellt daraus ein X509-Zertifikat.

Nun folgt die Erstellung des Webserverzertifikates, welches im Anschluss von der eben kreierten Certification Authority signiert wird.

```
openssl genrsa -out server.key 1024  
openssl req -new -key server.key -out server.csr  
echo -ne '01' > ca.srl  
openssl x509 -days 730 -CA ca.crt -CAkey ca.key -in server.csr -req -out server.crt
```

Hier generiert die erste Zeile ebenfalls ein RSA-Schlüsselpaar, diesmal handelt es sich um das Webserverzertifikat. Die zweite Zeile bittet zur Eingabe der persönlichen Zertifikatinformationen, die dritte erstellt eine Datei mit einer Seriennummer, die in diesem Falle "01" beträgt und die vierte Zeile erstellt aus den vorangegangenen Dateien das von der oben erstellten Certification Authority signierte Zertifikat. Die Dateien server.crt und server.key werden auf das Netcenter kopiert und die Berechtigungen wie oben beschrieben auf nur-Lesen des SSLwrap-Nutzers gesetzt.

Kritisch ist hingegen, dass das Zertifikat nur unverschlüsselt auf der Festplatte abgelegt werden kann, da SSLwrap über Xinetd gestartet wird und eine manuelle Eingabe des Passwortes daher nicht möglich ist. Da ohnehin nur private Dienste auf der Festplatte zur Verfügung gestellt werden, ist ein kommerzielles Interesse an einem Identitätsdiebstahl unwahrscheinlich und - aufgrund des Einsatzes eines selbstsignierten Zertifikates - nicht von großem Schaden. SSLwrap dient hier nur als Verbindungsverschlüsselung, nicht als Serveridentifikation. Als mindeste Sicherheitsvorkehrung sollte dennoch ein separater Benutzer für SSLwrap angelegt werden und das Zertifikat ausschließlich für diesen Nutzer lesbar sein. Weiterhin sollte sich das Zertifikat nicht in einem Ordner befinden, der von einem anderen Dienst im Netz freigegeben ist. Im Rahmen dieser Anleitung wird nicht auf das Anlegen eines separaten Nutzers und die Übertragung der Rechte auf diesen eingegangen.

Der folgende Startparameter dient dazu, SSLwrap für Lighttpd im Kontext des separaten Users "sslwrap" aufzurufen. Das Ausführen in einer neuen Shell ist

nötig, um den Dienst über die sudo-Syntax als Hintergrundprozess starten zu können.

```
sudo -u SSLwrap sh SSLwrap -accept 443 -port 80 -cert /opt/etc/server.crt -key /opt/etc/server.key&
```

Wird der Dienst dauerhaft genutzt empfiehlt sich ein Eintrag in der Xinetd.conf:

Listing 33: xinetd-Eintrag für SSLwrap

---

```
1 service https
2 {
3   disable      = no
4   wait         = no
5   protocol     = tcp
6   user         = SSLwrap
7   socket_type  = stream
8   only_from    = 0.0.0.0
9   server       = /opt/bin/sslwrap
10  server_args   = -cert /opt/etc/server.crt -key /opt/etc/server.key -port 80
11 }
```

---

## A.10 Bit Torrent Protocol Daemon

Bei dem “Bit Torrent Protocol Daemon”, kurz Btpd, handelt es sich um einen Bittorrent-Client. Im Gegensatz zu Clients wie ctorrent beherrscht dieser Client den gleichzeitigen Download mehrerer Torrents parallel und benötigt dazu lediglich einen eingehenden TCP-Port, was die Konfiguration des in fast jedem Heimnetzwerk vorhandenen DSL-Router vereinfacht. Mit Btpd ist die Möglichkeit gegeben, Torrents direkt über das Netcenter herunterzuladen, statt die entsprechende Aufgabe über einen Desktop-PC zu tätigen. So sind größere Downloads auch mit ausgeschaltetem Desktop-Rechner möglich. Um eine einfache Bedienung zu gewährleisten, wurde ein Webinterface in PHP programmiert. Der Quellcode wurde dem Internetforum der Maxtor-Shared-Storage-Community[12] zur Verfügung gestellt. Dieser findet sich ebenfalls auf der beiliegenden CD im Ordner wwwroot.

Um Mehrbenutzerzugang für Btpd zu ermöglichen, wurde auf mehrere parallele Instanzen von Btpd zurückgegriffen. Damit das Benutzermanagement mit einer möglichst ein-

fachen Syntax auskommt, wurden einige Skripte auf dem Netcenter implementiert. Diese sind ebenfalls im PHP-Webinterface implementiert. Eine Beschreibung des Webinterface ist nicht Bestandteil dieser Thesis.

Im ersten Schritt müssen für jeden Nutzer zwei Ordner angelegt werden. Der erste Ordner befindet sich in einem von Samba freigegebenen Verzeichnis. Dieser Ordner wird hier mit “/opt/christian/torrents/” angegeben. Hier werden die “.torrent”-Dateien vom Nutzer abgelegt, die heruntergeladen werden sollen. In diesem Ordner speichert Btpd auch die fertigen Download-Dateien.

Den zweiten Ordner benötigt der Btpd für temporäre Download- und Prozess-ID-Informationen. Dieser Ordner wird in einem nicht für den User sichtbaren Teil auf der Festplatte angelegt. In diesem Falle “/var/opt/torrent/christian”.

```
mkdir -p /opt/christian/torrents/ /var/opt/torrent/christian
```

Der Parameter “-p” legt nicht vorhandene, übergeordnete Verzeichnisse an.

Btpd starten: “btstart”

Listing 34: Bittorrent: btstart

---

```
1 #!/bin/sh
2 if [[ "$1" == "christian" ]]; then btpd -d /opt/var/torrent/$1 --bw-out 20 -p
   60000 --max-peers 300 & fi
```

---

Für jeden zusätzlichen Benutzer wird diesem Skript eine Zeile hinzugefügt, der Pfad angepasst und der Port erhöht. Wichtig ist an dieser Stelle, das Uploadlimit und die maximal zulässigen Verbindungen zu beachten. Übersteigt das Uploadlimit aller Nutzer die verfügbare Bandbreite des Internetzugangs, so ist unter Umständen die Internetverbindung vollständig ausgelastet. Ähnliches gilt bei zu vielen maximalen Verbindungen: Einige DSL-Router lassen sich mit zu vielen offenen Verbindungen zum Absturz bringen.

Hinzufügen von Downloads: “btadd”

Listing 35: Bittorrent: btadd

---

```
1 #!/bin/bash
2 cd /shares/Main/$1/torrents/
3 btstart $1
4 echo 0 > /proc/miscio/gpio_4
5 sleep 1
6 /opt/bin/find /shares/Main/$1/torrents/ -name '*.torrent' -exec /bin/btcli -d /opt
  /var/torrent/${1}/ add -d . {} \;
```

---

Der Befehl “btadd” sucht alle Dateien mit der Endung “.torrent” aus dem Verzeichnis des jeweiligen Benutzers und fügt sie dem Btpd hinzu. Dieses Skript macht dabei Gebrauch von der Executable “btcli”, welche die Benutzerschnittstelle zum Btpd darstellt und mit diesem über Unix-Streams kommuniziert. “btadd” erwartet als ersten Kommandozeilenparameter den Namen des Benutzers, den es direkt als Pfadangabe zur Kommunikation mit dem Btpd nutzt. Wird ein ungültiger Benutzer angegeben, gibt es eine Fehlermeldung, dass Btpd nicht gefunden wurde.

Dieses Vorgehen ist unter Sicherheitsaspekten ein unsauberes, da keine Verifikation des eingegebenen Benutzers erfolgt, sondern der übergebene String direkt verarbeitet wird. Eine Stringverifikation ist unter dem Sicherheits- und Manipulationsaspekt ein unbedingtes Muss, öffnet man den Zugang zum System einem breiteren Anwenderkreis. Die Weboberfläche verwendet eine solche Stringverifikation für alle dort getätigten Eingaben.

Vereinfachung der lokalen Userverwaltung: “bt”

Listing 36: Bittorrent: bt

---

```
1 #!/bin/bash
2 name=$1
3 /bin/btcli -d /opt/var/${name}torrent $2 $3 $4 $5 $6
```

---

Dieses Skript vereinfacht die lokale Statusabfrage über btcli. Es erwartet als Übergabeparameter den Benutzername an erster Stelle und den btcli-Befehl im Anschluss. Übliche Anwendungen zur Anzeige des (detaillieren) Downloadstatus, der Liste der Torrents mit Namen oder dem Löschen eines Torrents sind wie folgt:

*bt christian stat*  
*bt christian stat -i*  
*bt christian list*  
*bt christian del NUMBER*

Alle fertigen Torrents löschen: Bit Torrent delete all “btda”:

---

Listing 37: Bittorrent: btda

---

```
1 #!/bin/bash
2 bt $1 del `bt $1 list | grep 100.0% | sed -e 's|S\..*||' -e 's|.* ||'`
```

---

Dieses Skript verwendet reguläre Ausdrücke, indem es in dem detaillierten Status nach der Zeichenfolge “100%” sucht, die zugehörige Torrentnummer ausliest und diese an den Befehl “bt username del ” übergibt.

Torrentdienst beenden: “btkill”:

---

Listing 38: Bittorrent: btkill

---

```
1 #!/bin/bash
2 # 100% Torrents entfernen
3 btda $1
4 sleep 5
5 # Schauen, ob Bittorrent des Users $1 nichts herunterlädt ("nan" in der Liste
   aller Downloads)
6 process=$(bt $1 stat | sed -e '$!d' -e 's|%.*||' -e 's|^ *||')
7 if [[ "nan" == "$process" ]]; then
8   # Killen der ProzessID
9   bt $1 kill
10 fi
```

---

Gerade bei mehreren Benutzern ist es sinnvoll, den Torrentdienst aus Ressourcegründen zu beenden, falls kein Download mehr stattfindet. Daher wird das Skript zum Entfernen fertiger Downloads gestartet und nach einer kurzen Pause die letzte Zeile des Btpd-Status nach dem Wort “nan” durchsucht. Dieser taucht nur auf, wenn kein Download mehr aktiv ist. In diesem Fall wird der Dienst ordnungsgemäß beendet.

Das automatische Löschen fertiger Downloads widerspricht dem Prinzip von Bittorrent, fertige Downloads noch eine Zeit lang an andere Nutzer zu verteilen. Insofern sollte das

btkill-Skript nicht automatisch alle 3 min über das ipupdate-Skript gestartet werden, sondern beispielsweise lediglich um 05:00 Uhr nachts. Gleiches gilt selbstredend auch für das Skript “btlda” zum Löschen aller fertigen Torrents, auf welches “btkill” zurückgreift.

## A.11 Weitere Anwendungen

Weitere Anwendungen kurz angeschnitten, die ebenfalls auf dem Netcenter erfolgreich laufen:

**Curl** ist ein Programm zum Download von Dateien via http, ähnlich wget.

**microperl** ist ein funktionserleichterter Perl-Interpreter. Er kann beispielsweise zum parsen erweiterter regulärer Ausdrücke verwendet werden.

**BitchX** ist ein IRC-Client, der über die Kommandozeile bedient wird.

**Midnight Commander** ist ein Dateiverwalter ähnlich dem Nortoncommander für DOS.

**qEmacs** qEmacs ist ein Editor mit einer Fülle an Zusatzfunktionen. Gegenüber seines großen Bruders Emacs ist er um einige Funktionen erleichtert.

## A.12 Startup-Skript rc.start

Der Vollständigkeit halber das Startup-Skript rc.start:

Listing 39: Startskript rc.start

---

```
1 #Routen und Variablen setzen + exportieren
2 route add -net 0.0.0.0 gw 10.10.10.10 netmask 0.0.0.0
3
4 export PATH=/sbin:/bin:/usr/sbin:/usr/bin:/opt/bin
5 export LD_LIBRARY_PATH=/lib:/usr/lib:/opt/lib
6 export HOSTNAME=Netcenter
7 export TZ=CET-1CEST
8
9 ## Network Settings
10 # set network mode
11 #ifconfig eth0 media 100baseT media opt full-duplex
12 #ifconfig eth0 media 100baseT media opt half-duplex
13 #ifconfig eth0 media 10baseT media opt full-duplex
14 #ifconfig eth0 media 10baseT media opt half-duplex
15 ifconfig eth0 mtu 1492
16
17 ## Prozesse töten und NVRAM Parameter setzen
18 #Let httpd die
19 nvram set let_httpd_die=enabled
20 killall httpd
21 killall telnetd
22
23 # timeout harddisk inaktiv
24 #nvram unset initial_disk_spin_down_setting
25
26 # timeout harddisk aktiv 600 seconds
27 nvram set initial_disk_spin_down_setting=3600
28
29 # Web-Interface Port aendern
30 #nvram set web_configuration_port=1000
31
32 # Last-Power-State
33 nvram unset kernel_boot_wait_gpio
34
35 # NVRAM speichern
36 nvram commit
37
38 ## Pw-Dateien gegen Standard-Dateien austauschen
39 cp /opt/etc/resolv.conf /tmp/
40 cp /opt/etc/passwd /tmp/
41 cp /opt/etc/shadow /tmp/
42 cp /opt/etc/hosts /tmp/
43
44 # Namensserver töten
45 ps | grep wdns | cut -d " " -f 3 | xargs kill
```

```
46 ps | grep nmbd | cut -d " " -f 3 | xargs kill
47
48 # Samba beenden
49 ps | grep smbd | cut -d " " -f 3 | xargs kill
50
51 ## Programme starten
52 # Torrent Server starten
53 btstart christian
54
55 # DnsMasq starten
56 /opt/etc/init.d/S56dnsmasq &
57
58 # xinetd starten
59 /opt/sbin/xinetd &
60
61 # Webserver starten
62 /opt/bin/lighttpd/sbin/lighttpd -f /opt/bin/lighttpd/lighttpd.conf &
63
64 # NFS-starten
65 exportfs -a
66
67 # Samba beenden und mit eigener Config neustarten
68 /usr/local/samba/sbin/smbd -D -s /opt/etc/smb.conf
69
70 # Tor und Privoxy starten
71 /opt/bin/tor -f /opt/etc/torrc
72 /opt/sbin/privoxy /opt/etc/privoxy/config &
73
74 # NTP-Update
75 ntpclient -d -s -i 1 -h ptbtime1.ptb.de &
76
77 # DynDNSupdate
78 cp /opt/bin/ipupdate.sh /tmp
79 /tmp/ipupdate.sh &
```

---

## B Kompilation von Dnsmasq, Dropbear und Bash

### B.1 Dnsmasq kompilieren

Um dem Leser die Problematik der verschiedenen Compiler-Optionen näher zu bringen, wird in diesem Kapitel auf das Kompilieren von Dnsmasq, der Bash sowie Dropbear eingegangen. Im ersten Schritt wird Dnsmasq heruntergeladen und entpackt[42]. Grundlegende Compileroptionen für Freetz werden gesetzt mittels:

```
export PATH=/opt/toolchain/target/bin:$PATH
export CC=/opt/toolchain/target/bin/mipsel-linux-gcc
export CPPFLAGS=-I/opt/toolchain/target/include
export CPP=/opt/toolchain/target/bin/mipsel-linux-cpp
export LD_LIBRARY_PATH=/opt/toolchain/target/lib
export CROSS_COMPILE=mipsel-linux-uclibc-
export ARCH=mipsel-linux-uclibc
```

Nun wird versucht, Dnsmasq mittels *make* zu kompilieren. Der folgende Fehler tritt auf:

Listing 40: Dnsmasq-Compile: IPv6-Fehlermeldung

---

```
1 /opt/toolchain/target/bin/mipsel-linux-gcc -L/opt/toolchain/target/lib -o dnsmasq
   cache.o rfc1035.o util.o option.o forward.o network.o dnsmasq.o dhcp.o lease.
   o rfc2131.o netlink.o dbus.o bpf.o helper.o tftp.o log.o
2 option.o: In function 'one_opt':
3 option.c:(.text+0x42fc): undefined reference to 'in6addr_any'
4 option.c:(.text+0x45f4): undefined reference to 'in6addr_any'
5 network.o: In function 'reload_servers':
6 network.c:(.text+0x6d0): undefined reference to 'in6addr_any'
7 network.o: In function 'allocate_sfd':
8 network.c:(.text+0xabc0): undefined reference to 'in6addr_any'
9 network.o: In function 'pre_allocate_sfds':
10 network.c:(.text+0x1194): undefined reference to 'in6addr_any'
11 network.o:network.c:(.text+0x132c): more undefined references to 'in6addr_any'
    follow
12 collect2: ld returned 1 exit status
13 make[1]: *** [dnsmasq] Fehler 1
14 make[1]: Verlasse Verzeichnis '/home/christian/Desktop/dnsmasq-2.47/src'
15 make: *** [dnsmasq] Fehler 2
```

---

Der Fehler zeigt, dass Funktionseinsprünge, die IPv6 betreffen, nicht gefunden werden. Da das Netcenter ohnehin keinen IPv6-Support mitbringt, wird im nächsten Schritt versucht, Dnsmasq ohne IPv6-Support zu kompilieren. Es wird dem Make-Befehl die Option "COPTS=-DNO\_IPV6" mitgegeben. Die Kompilation läuft durch, allerdings handelt es sich bei dem Kompilat um ein dynamische gelinktes Executable, welches ohne die Bibliotheken auf dem Netcenter nicht startet.

```
make COPTS=-DNO_IPV6
```

---

Listing 41: Dnsmasq-Compile dynamisch gelinkt ohne IPv6

---

```
1 /opt/toolchain/target/bin/mipsel-linux-gcc -L/opt/toolchain/target/lib -o dnsmasq
   cache.o rfc1035.o util.o option.o forward.o network.o dnsmasq.o dhcp.o lease.
   o rfc2131.o netlink.o dbus.o bpf.o helper.o tftp.o log.o
2 make[1]: Verlasse Verzeichnis '/home/christian/Desktop/dnsmasq-2.47/src'
3 ~/Desktop/dnsmasq-2.47$ file src/dnsmasq
4 src/dnsmasq: ELF 32-bit LSB executable, MIPS, MIPS32 version 1 (SYSV), dynamically
   linked (uses shared libs), not stripped
```

---

---

Listing 42: Dnsmasq dynamisch gelinkt: Fehlermeldung

---

```
1 2/$/tmp/dnsmasq
2 -bash: /tmp/dnsmasq: No such file or directory
```

---

Es wird nun versucht, weitere Funktionen zum Beschränken der Binary-Größe auszuklammern und statisch zu linken. Dabei darf nicht vergessen werden, dass "-static" eine Option für den Linker, nicht für den Compiler ist. Übergibt man den "-static"-Parameter dem Compiler, gibt es keine Fehlermeldung, das Linken geschieht jedoch in diesem Falle dennoch dynamisch.

Korrekt lautet die verwendete Kommandozeile ohne IPv6-Unterstützung, ohne Internationalisation-Unterstützung, ohne DBus-Unterstützung, statisch gelinkt und im Anschluss gestript:

```
make COPTS="-DNO_IPV6 -DNO_I18N -DNO_DBUS" LDFLAGS=-static
mipsel-linux-strip src/dnsmasq
```

## B.2 Dropbear kompilieren

Dropbear besitzt eine recht ausführliche Readme-Datei. Nach Download von [48], entpacken und setzen der üblichen Umgebungsparameter für den Freetz-Compiler (siehe vorhergehendes Kapitel Dnsmasq) wird kompiliert mittels:

```
LDFLAGS=-Wl,-gc-sections  
CFLAGS="-ffunction-sections -fdata-sections"  
make PROGRAMS=dropbear MULTI=1 STATIC=1
```

Die ersten beiden Zeilen stellen Optimierungen für den gcc in Hinsicht der Binary-Größe dar. Sie wurden der Readme entnommen. Der Parameter "static=1" gibt an, dass statisch, also ohne Abhängigkeit von Bibliotheken des Netcenters kompiliert wird, "Multi=1" gibt an, dass alle Funktionen von Dropbear in einem einzigen Binary vereint werden und somit redundanter Code im Executable vermieden wird. Im Anschluss werden - ähnlich zur Busybox - Softlinks auf die einzelnen Funktionen gesetzt, d.h. auch das Dropbear-Executable erkennt anhand des aufgerufenen Dateinamens, wie es sich verhalten muss:

```
rm -f dropbear  
ln -s dropbearmulti dropbear  
rm -f dbclientmake  
ln -s dropbearmulti dbclient  
rm -f dropbearkey  
ln -s dropbearmulti dropbearkey  
rm -f dropbearconvert  
ln -s dropbearmulti dropbearconvert
```

Die neukompilierte Dropbearversion erwartet - da wir entsprechende Parameter nicht gesetzt haben - die rsa- und dsa-keyfiles unter einem anderen Pfad als die ursprünglich verwendete Version. Daher müssen die aufgerufenen Kommandozeilenparameter oder entsprechend die Server\_Args in der xinetd.conf um folgenden String erweitert werden:

```
-d /opt/etc/dropbear/dropbear_dsa_host_key -r /opt/etc/dropbear/dropbear_rsa_host_key
```

Eine der Neuerungen ab Version 0.50 ist die Verwendung von `/dev/urandom` statt `/dev/random` [47], sodass der im Kapitel A.6 beschriebene Hack als obsolet betrachtet werden kann.

### B.3 Bash kompilieren

Auch bei der Kompilation der Bash werden nach dem Download von [49] und entpacken des Archives im ersten Schritt die Umgebungsvariablen wie im Kapitel B.1 gesetzt. Die Kompilation der Bash nutzt im Gegensatz zu Dropbear und Dnsmasq ein `./configure`-Skript. Diesem muss via `“-host=mipsel-linux”`-Parameter das Zielsystem mitgeteilt werden. Weiterhin wird dem Linker ein `“-static”` übergeben, sodass der Kommandozeilenaufruf wie folgt lautet:

```
./configure -host=mipsel-linux LDFLAGS=-static  
make  
mipsel-linux-strip bash
```

## Glossar

**Auto-Negotiation** ist eine Methode, die Netzwerkgeräte beherrschen, um die Geschwindigkeit und den Duplexmodus der Gegenstelle zu detektieren. Dabei werden spezielle Signale verwendet, die nicht in der IEEE 802.3 spezifiziert sind. Unterstützt ein Gerät Auto-Negotiation nicht, werden ankommende Signale nicht beantwortet, die Gegenstelle nimmt in diesem Fall Halb-Duplex an. Halb-Duplex kann entweder senden oder empfangen, nicht beides gleichzeitig. Voll-Duplex bedeutet, dass gleichzeitig gesendet und empfangen werden kann, das Netz ist daher kollisionsfrei.

**Bibliotheken** stellen in einem Betriebssystem Funktionen zur Verfügung, die von mehreren Programmen (gleichzeitig) genutzt werden.

**Bridging** stellt die Kopplung zweier Netzwerktechnologien dar (beispielsweise Ethernet mit Tokenring oder Ethernet mit Wireless Ethernet). Der Begriff wird im WLAN-Umfeld ebenfalls genutzt, wenn zwei Ethernet-Netze via WLAN miteinander gekoppelt werden.

**Busybox** Die Busybox ist ein Linux-Binary, welches speziell für Embedded-Devices entwickelt wurde und die wichtigsten Unix-Kommandozeilenbefehle mit abgespeckter Parameterunterstützung speicherplatzsparsam vereint.

**Caching** bezeichnet das Zwischenspeichern von Informationen. Im Falle des Internetproxys werden von mehreren Clients aufgerufene Webseiten nur einmal aus dem Internet geladen und den weiteren Clients die zwischengespeicherte Version serviert.

**Carriage-Return** Carriage-Return (CR) ist neben dem Line-Feed (LF) eines der Zeilenumbruchsymbole. CR bedeutet im eigentlichen Sinne, den Cursor innerhalb der gleichen Zeile an die Anfangsposition zu setzen. LF hingegen ist das Setzen des Cursors in die nächste Zeile. Während Windows als Zeilenumbruchsymbold ein CR+LF fordert, nutzt Apple lediglich ein CR und Linux das LF. Unter Linux ist ein CR insofern störend, als es die aktuelle Zeile mit nachfolgenden Symbolen überschreibt.

**Dämon** Ein Dämon ist ein dauerhaft aktives Programm, unter Windows auch als Dienst bezeichnet. Dieser nimmt in irgend einer Form Daten entgegen oder beantwortet Anfragen. Dienste können beispielsweise Server für die Folgenden Protokolle sein: FTP, HTTP, DNS

**DBus** ist ein Softwaresystem für Interprozesskommunikation.

**(Distributed) Denial of Service Angriff** Als Denial of Service (DoS, zu Deutsch etwa: Dienstverweigerung) bezeichnet man einen Angriff auf einen Host (Server) oder sonstigen Rechner in einem Datennetz mit dem Ziel, einen oder mehrere seiner Dienste arbeitsunfähig zu machen. In der Regel geschieht dies durch Überlastung. Erfolgt der Angriff koordiniert von einer größeren Anzahl anderer Systeme aus, so spricht man von Verteilter Dienstblockade bzw. DDoS (Distributed Denial of Service). (Quelle: Wikipedia)

**DHCP** Dynamic Host Configuration Protokoll. Ein Netzwerkprotokoll zur zentralen und konfigurationsfreien Vergabe von IP-Adressen. Mittels zusätzlicher Optionen ist die Vergabe nicht auf IP-Adressen beschränkt, sondern kann auf Namen oder Domains und zusätzliche Netzwerkparameter ausgedehnt werden.

**Disk-Spindown-Zeit** Eine manuell festgelegte Zeit der Inaktivität, die verstreichen muss, bis die interne Festplatte abgeschaltet wird.

**DNS-Relay** Ein DNS-Relay nimmt DNS-Anfragen entgegen und löst sie mithilfe externer DNS-Server auf. Im Gegensatz zu einem DNS-Server verfügt es über keine eigene Zone, d.h. es ist vollständig auf den externen DNS-Server zur Auflösung angewiesen. Die Auflösung kann - je nach Relay - rekursiv oder iterativ erfolgen. DSL-Router verfügen meist über ein solches Relay. Der Nutzen solcher Relay ist einzig die Zusammenfassung von Standardgateway und DNS-Server im Rahmen der manuellen Konfiguration.

**Embedded Device** Ein Embedded Device, zu Deutsch “eingebettetes System”, ist ein elektronisches Gerät, welches in eine Umgebung eingepasst wurde und dort Messung, Regelung und Steuerung übernimmt. Die Implementierung der Geräte erfolgt abhängig von der Preisklasse auf leistungsarmer und energiesparsamer Standardhardware in Software, häufig mit Windows CE oder Linux als Betriebssystem, oder in Hardware (Haussteuerung, Automobiltechnik).

**Executable** Eine ausführbare Binärdatei.

**Firmware** bezeichnet das Betriebssystem von Embedded Devices. Meist wird die Firmware auf einem EEPROM- oder Flash-Medium abgespeichert. Sie ist in der Regel austauschbar, im Gegensatz zu Betriebssystemen von Computern jedoch nicht während der Nutzung beschreibbar oder änderbar.

**Full-Duplex** Voll-Duplex, siehe Auto-Negotiation.

**ISP** Internet-Service-Provider. Ein Anbieter von Internetanschlüssen über Modem, ISDN, DSL, Kabel oder Satellit.

**Layer2-Switches** Ethernetswitches, die ausschließlich auf OSI Layer 2 arbeiten. Sie beschränken sich auf das Weiterleiten von Ethernetframes und besitzen keinerlei Routing-Funktionalität auf OSI Layer 3.

**LPR** steht für Line Printer Daemon protocol / Line Printer Remote protocol (LPD, LPR). Es handelt sich um Druckerspooles und Netzwerkdruckserver, die ursprünglich unter Unix eingesetzt wurden.

**Masquerading** bezeichnet das Verstecken von mehreren Client-Rechnern im Netz hinter einer einzigen, nach außen sichtbaren Adresse. Auch als Port Address Translation (PAT) bezeichnet. Dieses wiederum wird im Volksmund auch als NAT (Network AT) bezeichnet.

**Mime-Type** Multipurpose Internet Mail Extensions (MIME) ist ein Kodierstandard, der die Struktur und den Aufbau von E-Mails und anderen Internetnachrichten festlegt. (Quelle: Wikipedia)

**MIPSel** MIPSel steht für die Little Endian-Version des MIPS-Prozessors. MIPS ist ein Mikroprozessor basierend auf dem RISC-Konzept und steht für “Microprocessor without interlocked pipeline stages”. Ziel des Prozessordesigns ist es, die Hardwarearchitektur durch In-Order-Execution - der Prozessor kann die Befehlsabfolge nicht abändern - schlank zu halten und die Befehlsabfolge schon über den Compiler zu optimieren.

**MTU** steht für Maximum Transfer Unit und gibt die maximale Paketgröße der Nutzdaten eines Ethernetframes an. Dieser ist im Standard auf 1500 Byte festgesetzt. Führt der Weg zum Ziel nicht ausschließlich über Ethernet, sondern über ein Netz, deren Paketgröße kleiner als 1500 Byte ist, so bietet es sich unter Umständen an, die maximale Paketgröße auch im Ethernet zu reduzieren, um Fragmentierung zu vermeiden. Eine DSL-Verbindung besitzt aufgrund des zusätzlichen PPP-Headers von 8 Bytes eine MTU von 1492 Bytes.

**Multithreaded** bezeichnet die Fähigkeit, zwei Prozesse simultan abarbeiten zu können.

**NAS** steht für Network Attached Storage und bezeichnet ein per Netzwerk angebundenes Speicher, häufig in Form von per Netzwerk erreichbarer Festplattensysteme, die ihren Speicherplatz über Protokolle wie NFS oder SMB verfügbar machen.

**Netzwerkstack** Ein Netzwerkstack ist ein Stück Software, welches das Senden und Empfangen von Informationen über ein Netzwerk ermöglicht.

**NFS** steht für Network File System. Es ist ein Netzwerkdateisystem unter Linux.

**Paketmanager** Paketmanager verwalten Installationsarchive hinsichtlich Installation und Deinstallation für Linux-Distributionen und kennen die Abhängigkeiten verschiedener Pakete. Bekannte Paketmanager sind der “Red-Hat Packet Manager” (RPM) und der “Debian Packet Manager” (dpkg).

**Pipelining** In dem in dieser Thesis verwendeten Kontext ist Pipelining das Zusammenfassen von mehreren HTTP-Anfragen innerhalb einer HTTP-Verbindung. Durch Pipelining wird die Geschwindigkeit erhöht, da der Verbindungsaufbau für jede “Pipeline” nur einmal stattfindet.

**PPPoE** Point to Point Protocol over Ethernet. Eine PPP-Dial-Up Verbindung, die auf Ethernet aufsetzt. PPPoE ist das Standardprotokoll in Deutschland, welches zur Authentifizierung einer DSL-Verbindung genutzt wird.

**promiscuous** stammt von dem Wort promiskuitiv = wahllos. Nach dem Ethernetstandard konfigurierte Netzwerkkarten nehmen nur Frames entgegen, die an ihre eigene MAC-Adresse (Hardwareadresse einer Netzwerkkarte) oder an die Broadcastadresse adressiert sind. Eine Netzwerkkarte im "promiscuous"-Modus nimmt jedes Datenpaket entgegen. Dieser Modus ist Voraussetzung für Paketsniffer oder transparente Firewalls.

**PXE-Server** Das Preboot eXecution Environment ist ein Verfahren, um Computern einen netzwerkbasierten Bootvorgang zu ermöglichen, der von client-seitig verfügbarem Massenspeicher und insbesondere Betriebssystemen unabhängig ist. (Quelle: Wikipedia)

**Regex** In der Informatik ist ein Regulärer Ausdruck (engl. regular expression, Abk. RegExp oder Regex) eine Zeichenkette, die der Beschreibung von Mengen beziehungsweise Untermengen von Zeichenketten mit Hilfe bestimmter syntaktischer Regeln dient. (Quelle: Wikipedia)

**Routing** bezeichnet das Leiten von paketvermittelten Datenströmen durch vermaschte Netze.

**signed / unsigned integer** Integer bezeichnet eine Darstellungsform von Ganzzahlen für Computer. Die Ganzzahlen können je nach Deklaration nur im positiven, oder sowohl im positiven als auch negativen Bereich liegen. Im ersten Fall gibt es kein Vorzeichen (unsigned), im zweiten Fall wird ein Bit für das Vorzeichen der Zahl verwendet (signed).

**SMB/Samba** SMB steht für Server Message Block und ist ein proprietäres Protokoll der Firma Microsoft zum Bereitstellen von Netzlaufwerken. Es wurde unter dem Namen Samba via Reverse-Engineering auf Linux portiert.

**Socket** Ein Socket ist eine Bezeichnung für die IP-Adresse (OSI Layer 3) zusammen mit dem TCP/UDP-Port (OSI Layer 4).

**Socks-Proxy** Socks steht für Socket und bezeichnet einen Server, der Proxy-Funktionalität (Weiterleitungsfunktionalität) unabhängig vom eingesetzten OSI Layer 5 Protokoll ermöglicht.

**Unix-Streams** Streams stellt eine Möglichkeit der Prozesskommunikation unter Unix/Linux dar. Es ermöglicht Vollduplex-fähigen, bidirektionalen Zeichenaustausch (Charakter-Device).

**URL** Als Uniform Resource Locator (URL, dt. "einheitlicher Quellenanzeiger") bezeichnet man eine Unterart von Uniform Resource Identifiern (URIs). URLs identifizieren und lokalisieren eine Ressource über das verwendete Netzwerkprotokoll (beispielsweise http oder ftp) und den Ort (engl. location) der Ressource in Computernetzwerken. (Quelle: Wikipedia)

**Web of Trust** Netz des Vertrauens bzw. Web of Trust (WOT) ist in der Kryptologie die Idee, die Echtheit von digitalen Schlüsseln durch ein Netz von gegenseitigen Bestätigungen (Signaturen) zu sichern. Es stellt eine dezentrale Alternative zum hierarchischen PKI-System (Private Key Infrastructure) dar. (Quelle: Wikipedia)

**WINS** Windows Naming Service ist ein proprietäres Microsoftprotokoll zur Namensauflösung, die unter dem Betriebssystem Windows zum Einsatz kommt.

**WLAN-Sniffer** Ein (WLAN)-Sniffer speichert alle durch ein Netzwerk geleiteten Pakete. Neben dem Einsatz zur Netzwerküberwachung und -fehlerbehebung dient er zum Ausspionieren von Informationen.

**WoL** Wake on LAN ermöglicht, einen an das Netzwerk angeschlossenen Rechner via Datenpaket ("Magic Paket") einzuschalten.

**Wrapper** Ein Wrapper ist ein Programm mit Übersetzungsfunktion. Ein Wrapper kommuniziert mit einem Programm und gibt die Daten in einem anderen Format / Protokoll aus. Beispiele sind Torify, welches den Netzwerkverkehr eines Programmes über das TOR-Netzwerk leitet, oder SSL Wrap, welches eine unverschlüsselte HTTP-Verbindung in eine verschlüsselte HTTPS-Verbindung transformiert.

## Literatur

- [1] Heise: Von Datenschutz und Schäuble-Katalog: Terrorbekämpfung, TK-Überwachung, Online-Durchsuchung  
<http://www.heise.de/ct/hintergrund/meldung/95584>
- [2] Heise: BKA-Präsident bekräftigt Forderung nach Mautdatennutzung und Online-Durchsuchungen  
<http://www.heise.de/newsticker/BKA-Praesident-bekraeftigt-Forderung-nach-Mautdatennutzung-und-Online-Durchsuchungen-/meldung/97774>
- [3] Heise: Bundesregierung beschließt Auskunftsanspruch gegen Provider  
<http://www.heise.de/newsticker/Bundesregierung-beschliesst-Auskunftsanspruch-gegen-Provider-/meldung/84214>
- [4] Heise: Identifikationsnummer für alle Bürger kommt ab Juli  
<http://www.heise.de/newsticker/Identifikationsnummer-fuer-alle-Buerger-kommt-ab-Juli-/meldung/90890>
- [5] IETF: WPAD Implementierung  
<http://tools.ietf.org/html/draft-cooper-webi-wpad-00>
- [6] Net-Center Community <http://www.brennwertheizung24.de/wdncforum/>
- [7] The Onion Router - Webseite <http://www.torproject.org>
- [8] The Onion Router - Signaturverifikation  
<https://wiki.torproject.org/noreply/TheOnionRouter/VerifyingSignatures>
- [9] Opensource Portal Sourceforge <http://sourceforge.net/>
- [10] Suchmaschine Google <http://www.google.de/>
- [11] DD-WRT Wiki über DNSmasq  
[http://www.dd-wrt.com/wiki/index.php/DNSMasq\\_als\\_DHCP\\_Server](http://www.dd-wrt.com/wiki/index.php/DNSMasq_als_DHCP_Server)

- [12] Internetforum für alternative Firmware des Maxtor Storage Center  
<http://openmss.org/>
  
- [13] Heise: Schwache Krypto Schluessel unter Debian  
<http://www.heise.de/security/Schwache-Krypto-Schluessel-unter-Debian-Ubuntu-und-Co-/news/meldung/107808>
  
- [14] Bugtrack für Mozillabug 356831: Proxy autodiscovery doesn't check DHCP (option 252) [https://bugzilla.mozilla.org/show\\_bug.cgi?id=356831](https://bugzilla.mozilla.org/show_bug.cgi?id=356831)
  
- [15] Heise: Apple steigt um auf Intel Architektur  
<http://www.heise.de/newsticker/WWDC-Apple-steigt-um-auf-Intel-Architektur-4-Update-/meldung/60335>
  
- [16] Bernd Leitenberger: Die Gegensätze der Rechnerarchitekturen  
<http://www.bernd-leitenberger.de/cisc-risc.shtml>
  
- [17] Heise: Sicherheitslücke im Browser: Extra-Patch für Internet Explorer  
<http://www.heise.de/newsticker/Extra-Patch-fuer-Internet-Explorer-/meldung/120552>
  
- [18] Writing Effective Proxy PAC Files  
<http://jurnow.home.comcast.net/jurnow/WritingEffectivePACFiles.html>
  
- [19] Der falsche Klick <http://www.tagesspiegel.de/politik/deutschland/BKA-Datenschutz;art122,2390884>
  
- [20] Sperren von Kinderpornoseiten  
<http://www.heise.de/newsticker/Familienministerin-Provider-machen-mit-beim-Sperren-von-Kinderporno-/meldung/121769>
  
- [21] Todesstrafe für unliebsame Internetnutzer <http://info.kopp-verlag.de/news/china-todesstrafe-fuer-unliebsame-internetnutzer.html>
  
- [22] LG Hamburg: Mitstörerhaftung bei offenem WLAN  
<http://www.advocatus.de/heng/weblog.php?id=P1383>

- [23] Startseite des Wubi-Projektes <http://wubi-installer.org/>
  
- [24] Western Digital Toolchain für das Netcenter  
<http://support.wdc.com/product/download.asp?groupid=409&sid=41&lang=de>
  
- [25] Dynamic Host Configuration Protocol <http://tools.ietf.org/html/rfc2131>
  
- [26] DOMAIN NAMES - CONCEPTS AND FACILITIES  
<http://tools.ietf.org/html/rfc1034>
  
- [27] DOMAIN NAMES - IMPLEMENTATION AND SPECIFICATION  
<http://tools.ietf.org/html/rfc1035>
  
- [28] Uni-München: Reguläre Sprachen, reguläre Ausdrücke  
<http://www.lrz-muenchen.de/services/schulung/unterlagen/regul/>
  
- [29] DHCP Options and BOOTP Vendor Extensions <http://tools.ietf.org/html/rfc2132>
  
- [30] IANA: DHCP-Optionen  
<http://www.iana.org/assignments/bootp-dhcp-parameters/>
  
- [31] Lighttpd, PHP und MySQL auf dem Netcenter installieren  
<http://www.openmss.org/forum/viewtopic.php?f=3&t=424>
  
- [32] Sourceforge Suche nach httpd <http://sourceforge.net/search/?words=httpd>
  
- [33] Lighttpd vs. thttpd  
<http://www.damonparker.org/blog/2005/04/10/lighttpd-vs-thttpd/>
  
- [34] Lighttpd vs. thttpd Update  
<http://damonparker.org/blog/2005/04/14/lighttpd-vs-thttpd-part-2/>
  
- [35] Benchmark Lighttpd vs thttpd  
<http://damonparker.org/html/thttpd-vs-lighttpd.html>
  
- [36] FAQ des xinet-Dienstes <http://www.xinetd.org/faq.html>

- [37] How To: Firmware des Netcenter extrahieren und neu packen  
<http://www.brennwertheizung24.de/netcenter/seiten/files/Howto-Filesystem-Netcenter.pdf>
- [38] Programming the Linksys WRT54GS Wireless Broadband Router  
[http://sarwiki.informatik.hu-berlin.de/Programming\\_the\\_Linksys\\_WRT54GS\\_Wireless\\_Broadband\\_Router](http://sarwiki.informatik.hu-berlin.de/Programming_the_Linksys_WRT54GS_Wireless_Broadband_Router)
- [39] Sourceforge Projekt CramFS <http://sourceforge.net/projects/cramfs/>
- [40] Sourceforge Projekt zum CramFS Nachfolger SquashFS  
<http://squashfs.sourceforge.net/>
- [41] Seite des Hex-Editor Bless für Linux <http://home.gna.org/bless/>
- [42] Seite des DHCP/DNS/TFTP-Server Dnsmasq  
<http://www.thekelleys.org.uk/dnsmasq/doc.html>
- [43] Linksys Toolchain Download  
[http://www-ca.linksys.com/servlet/Satellite?c=L\\_Download\\_C2&childpage=CAen%2FLayout&cid=1153781216392&packedargs=sku%3D1175239204445&pagename=Linksys%2FCommon%2FVisitorWrapper&lid=1639212181B08](http://www-ca.linksys.com/servlet/Satellite?c=L_Download_C2&childpage=CAen%2FLayout&cid=1153781216392&packedargs=sku%3D1175239204445&pagename=Linksys%2FCommon%2FVisitorWrapper&lid=1639212181B08)
- [44] Webseite des Firefox-Werbefilter-Plugin Adblock Plus  
<https://addons.mozilla.org/de/firefox/addon/1865>
- [45] Internetportal des Webforums FluxBB <http://fluxbb.org/>
- [46] Open-WRT Projekt: Status Firmwareentwicklung Netcenter  
<http://wiki.openwrt.org/OpenWrtDocs/Hardware/WD/NetCenter>
- [47] Release-Notes von Dropbear <http://matt.ucc.asn.au/dropbear/CHANGES>
- [48] Download-Seite für Dropbear <http://matt.ucc.asn.au/dropbear>
- [49] Downloadseite der Bash <http://ftp.gnu.org/gnu/bash/>

- [50] Anleitung zum Kompilieren der Busybox  
[http://www.wehavemorefun.de/fritzbox/index.php/Busybox\\_ersetzen](http://www.wehavemorefun.de/fritzbox/index.php/Busybox_ersetzen)
- [51] Webpräsenz der uclibc-Toolchain <http://www.uclibc.org/download.html>
- [52] Webpräsenz der Busybox <http://www.busybox.net/>
- [53] Sourceforge Downloads von Privoxy  
[https://sourceforge.net/project/showfiles.php?group\\_id=11118](https://sourceforge.net/project/showfiles.php?group_id=11118)
- [54] “Anonym im Netz” von “Jens Kubicziel”, erschienen im synopsis Verlag, 1. Auflage ISBN: 978-3-937514-42-0
- [55] 25c3 / 2977 Videomitschnitt: Roger Dingledine: Security and anonymity vulnerabilities in TOR [http://mirror.core.im/video/25c3/video\\_h264.iPod/25c3-2977-en-security\\_and\\_anonymity\\_vulnerabilities\\_in\\_tor.ipod.m4v](http://mirror.core.im/video/25c3/video_h264.iPod/25c3-2977-en-security_and_anonymity_vulnerabilities_in_tor.ipod.m4v)
- [56] Webseite des Proxyservers Polipo <http://www.pps.jussieu.fr/~jch/software/polipo>
- [57] Anleitung zu Privoxy von Hauptentwickler Fabian Keil  
<http://www.fabiankeil.de/privoxy-anleitung>
- [58] Compiler, Assembler, Linker and Loader: A brief story  
<http://www.tenouk.com/ModuleW.html>
- [59] Selfhtml-Forum: Beschreibung des Configure-Skriptes  
<http://forum.de.selfhtml.org/archiv/2007/2/t145389>
- [60] Downloadseite für die Freetz-Toolchain <http://trac.freetz.org/downloads>
- [61] Anleitung zur Erstellung der Freetz-Toolchain  
[http://www.wehavemorefun.de/fritzbox/index.php/Cross-Compile\\_Toolchain](http://www.wehavemorefun.de/fritzbox/index.php/Cross-Compile_Toolchain)
- [62] Busyboxhomepage: Getting Started  
[http://www.busybox.net/FAQ.html#getting\\_started](http://www.busybox.net/FAQ.html#getting_started)

- [63] Heise: KeeLoq Schlüssel geknackt <http://www.heise.de/autos/Forscherteam-hat-KeeLoq-Schluesel-mit-50-Rechnern-in-zwei-Tagen-geknackt-/artikel/s/4362>
  
- [64] Heise: Mifare RFID-Verschlüsselung geknackt  
<http://www.heise.de/security/Letzte-Details-der-Mifare-RFID-Verschlueselung-veroeffentlicht-/news/meldung/117074>
  
- [65] Impressum mit PGP-Key des Privoxy-Entwicklers Fabian Keil  
<http://www.fabiankeil.de/autor.html>
  
- [66] Seite von Fabian Keil über die Privoxy Konfigurationsdateien  
<http://www.fabiankeil.de/privoxy-anleitung/#konfigurationsdateien>
  
- [67] Heise: Portal, um die eigene, aktuelle IP-Adresse einzusehen  
<http://www.heise.de/ip>
  
- [68] Red Hat Enterprise Linux 4 Referenzhandbuch: TCP Wrapper und xinetd  
<http://web.mit.edu/rhel-doc/4/RH-DOCS/rhel-rg-de-4/s1-tcpwrappers-xinetd-config.html>
  
- [69] Maxtor Shared Storage: Tool Download Page  
[http://www.seagate.com/www/de-de/support/downloads/shared\\_storage\\_plus/](http://www.seagate.com/www/de-de/support/downloads/shared_storage_plus/)
  
- [70] Debian Howto: Configure VSFTPD  
[http://archiv.debianhowto.de/de/vsftpd/c\\_vsftpd.html](http://archiv.debianhowto.de/de/vsftpd/c_vsftpd.html)
  
- [71] Dynamischer DNS-Provider dyndns.org <http://www.dyndns.org>
  
- [72] Manpage Crontab <http://unixhelp.ed.ac.uk/CGI/man-cgi?crontab>
  
- [73] Heise: Konsequenzen des OpenSSL-Debakels  
<http://www.heise.de/newsticker/Konsequenzen-des-OpenSSL-Debakels-/meldung/107921>
  
- [74] Ca-Cert: Free SSL Certificates <http://www.cacert.org>

- [75] Heise: Kinderporno-Sperren im internationalen Vergleich  
<http://www.heise.de/newsticker/Kinderporno-Sperren-im-internationalen-Vergleich-/meldung/133295>
- [76] Homepage des Recoveryprogramm Testdisk  
<http://www.cgsecurity.org/wiki/TestDisk>
- [77] Intel Math Kernel Library  
<http://www.intel.com/cd/software/products/asm-na/eng/307757.htm>
- [78] Pro-Linux: Der Gnu-Privacy Guard <http://www.pro-linux.de/berichte/gnupg.html>

## Index

- Bash, Compile, 124
- Bashrc, 104–106
- Bibl., statisch vs. dynamisch, 61
- Bibliotheken, 58–64
- Bibliotheken, Freetz-, 49–51
- Bibliotheken, ldd.sh, 62–64
- Bit Torrent Protocol Daemon, 114–118
  
- Compiling, ./configure, 46–47
- Compiling, Bibliotheken, 45–46
- Compiling, Funktionsweise, 42–47
- Compiling, Make, 46, 51
- Cross-C., Bash, 124
- Cross-C., Busybox, 48, 53–56
- Cross-C., Dnsmasq, 121–122
- Cross-C., Dropbear, 122–124
- Cross-C., Fehlerbehebung, 56–59, 74–78
- Cross-C., Prime, 42–45, 51–53
- Cross-C., Privoxy, 69–74
- Cross-C., Toolchain, 40–42
- Cross-C., Toolchain, Freetz-, 42, 48–51
- Cross-C., Toolchain, hnd-, 41–42, 48
- Cross-C., Toolchain, uclibc-, 42
- Cross-C., TOR, 65–69
- Cross-C., vs. Nativ-, 40
- Cross-Compiling, 8
  
- Dienste, Zusammenspiel der, 39
- Disk-Spindown-Zeit, 20
- Dnsmasq, 21
- Dnsmasq, Compile, 121–122
- Dropbear, Compile, 122–124
- Dropbear, Hack, 102–104
  
- Fazit, Ausblick, 87–88
- Fazit, Zielerreichung, 85–86
- Firmware, Manipulation der, 20, 93–101
- Firmware, Recoverymodus, 99–101
  
- Geschwindigkeitsmessung, 80–83
  
- http-Webinterface, 18
- http-Webinterface, Lighttpd, 23–25
  
- Internet, Zugriff aus dem, 17–18
- IP-Updateskript, 106–111
- IPKG Paketmanager, 15–16
  
- Netcenter, HDD-Recovery, 62
- Netcenter, Kernel, 15
- Netcenter, sonstige Software, 118
- Netcenter, technische Daten, 13–15
- Netzwerkumgebung, 13
  
- Privoxy, Compiling, 69–74
- Privoxy, Funktionsweise, 37–38
- Privoxy, Konfiguration, 71–74
- Proxy Auto-Config, 25–26
  
- Random-Device, Test des, 103
  
- Samba, 18–19
- Speicher sparen, 20–21, 23
- SSLwrap, 111–114
- SSLwrap, Zertifikaterstellung, 112–113
- Startupfile, rc.start, 17–21, 119–120
- Statusleuchte, Setzen der, 108–109
  
- TOR, Compiling, 65–69
- TOR, Funktionsweise, 32–34
- TOR, Konfiguration, 67–69
- TOR, Schwächen, 34–37

TOR-Button, [78–79](#)

Vsftpd, Konfiguration, [101–102](#)

Web Proxy Autodiscovery P., [26–29](#)

Xinetd, [89–93](#)

Zensur, Google, [82–83](#), [86–87](#)

Zwangproxy, [29–31](#)

Hiermit versichere ich, dass die vorliegende Arbeit von mir selbstständig und ohne unerlaubte Hilfe angefertigt worden ist, insbesondere dass ich alle Stellen, die wörtlich oder annähernd wörtlich aus Veröffentlichungen entnommen sind, durch Zitate als solche gekennzeichnet habe. Ich versichere auch, dass die von mir eingereichte schriftliche Version mit der digitalen Version übereinstimmt. Weiterhin erkläre ich, dass die Arbeit in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegen hat. Ich erkläre mich damit einverstanden, dass die Arbeit der Öffentlichkeit zugänglich gemacht wird. Ich erkläre mich damit einverstanden, dass die Digitalversion dieser Arbeit zwecks Plagiatsprüfung auf die Server externer Anbieter hoch geladen werden darf. Die Plagiatsprüfung stellt keine Zurverfügungstellung für die Öffentlichkeit dar.

Dormagen, 23.02.2009  
(Ort, Datum)

(Eigenhändige Unterschrift)