

Paket PROXY

Version 4.0.0-trunk-x86-r60800

Frank Meyer Das fli4l-Team
E-Mail: frank@fli4l.de E-Mail: team@fli4l.de

20. Dezember 2022

Inhaltsverzeichnis

1	Dokumentation des Paketes PROXY	3
1.1	PROXY - Verschiedene Proxy-Server	3
1.1.1	OPT_PRIVOX - Ein Werbung-filternder HTTP-Proxy	3
1.1.2	OPT_TOR - Ein anonymes Kommunikationssystem für das Internet . .	6
1.1.3	OPT_SS5 - Ein Socks4/5 Proxy	7
1.1.4	OPT_TRANSPROXY (EXPERIMENTELL) - Transparenter HTTP-Proxy	8
1.1.5	OPT_SIPROXD (experimentell) – Proxy für Session Initiation Protocol	8
1.1.6	OPT_IGMPPROXY - Proxy für Internet Group Management Protocol	10
1.1.7	OPT_STUNNEL - Tunneln von Verbindungen über SSL/TLS	17
	Abbildungsverzeichnis	24
	Tabellenverzeichnis	25
	Index	26

1 Dokumentation des Paketes PROXY

1.1 PROXY - Verschiedene Proxy-Server

1.1.1 OPT_PRIVOXY - Ein Werbung-filternder HTTP-Proxy

Privoxy wird auf der offiziellen Privoxy-Homepage (<http://www.privoxy.org/>) als "Privacy Enhancing Proxy" (= "privatsphärenerweiternder Proxy") beworben. Als sichtbarer und erwünschter Nebeneffekt ersetzt Privoxy Werbe-Banner und -Popups durch leere Bilder, verhindert das Speichern von ungewollten Cookies (kleine Datenpakete, mit denen eine Website einen bestimmten Surfer wiedererkennen kann) und verhindert die Anzeige von sogenannten Web-Bugs (das sind 1x1 Pixel große Bilder, die dazu benutzt werden, um das Surfverhalten von Benutzern auszuspähen).

Privoxy kann, während es läuft, ganz einfach über ein Webinterface konfiguriert und (de)aktiviert werden. Dieses Webinterface findet sich unter <http://config.privoxy.org/> oder <http://p.p/>.

Privoxy ist eine konsequente Weiterentwicklung des Internet Junkbusters, der bis Version 2.1.0 in diesem Paket (<http://www.junkbuster.com/>) enthalten war. Die wichtigste Neuerung ist, dass alle Regeln für die Filterung in der zentralen Datei `default.action` definiert werden. Diese befindet sich bei fli4l im Verzeichnis `/etc/privoxy`. Der große Vorteil an dieser Methode ist, dass sich neue Versionen dieser Datei separat von <http://sourceforge.net/projects/ijbswa/files/> herunterladen lassen. So kann jeder fli4l-Benutzer die Datei selbst auf dem neusten Stand halten, ohne auf Updates von fli4l angewiesen zu sein. (Momentan befindet sich die Version 1.8 dieser Datei im Paket.)

Eine so
getätigte
Konfi-
guration
überlebt
keinen
Neu-
start... (tobi)

PRIVOXY_MENU Fügt dem httpd-Menü einen Privoxy-Abschnitt hinzu.

PRIVOXY_N Gibt die Anzahl der Privoxy-Instanzen an, die gestartet werden sollen.

PRIVOXY_x_LISTEN Hier werden die IP-Adressen oder symbolischen Namen inklusive der Portnummer der Interfaces angegeben, auf denen Privoxy auf Verbindungen von Clients horchen soll. Es ist eine gute Idee, hier nur die Adressen der Interfaces anzugeben, denen man vertraut, da alle Rechner vollen Zugriff auf Privoxy haben (und auf den eventuell aktivierten Konfigurations-Editor). In der Regel ist die Vorgabe `IP_NET_1_IPADDR:8118` sinnvoll.

Auf hier angegebenen Adressen lauscht Privoxy und bietet seine Dienste an. 8118 ist der Standard-Port. Die Angabe hier muss man dann bei der Proxy-Konfiguration des jeweils verwendeten Internet-Browsers benutzen. Weitere Informationen zur Konfiguration von Internet Explorer und Netscape Navigator auf

<http://www.privoxy.org/>

Als Proxy beim jeweiligen Browser muss der fli4l-Rechner angegeben werden, also das, was man bei `HOSTNAME='fli4l'` angegeben hat bzw. dessen IP (z.B. 192.168.6.1), die

Genaue
URL

man bei `HOST_x_IP='192.168.6.1'` angegeben hat. Zusammen mit dieser Port-Angabe hier hat man dann alle nötigen Daten, um seinen Webbrowser für die Nutzung von Privoxy zu konfigurieren.

PRIVOXY_x_ALLOW_N Gibt die Anzahl der Listeneinträge an.

PRIVOXY_x_ALLOW_x Die Liste der Netze und/oder IP-Adressen für die der Paketfilter geöffnet wird. Sinnvoll ist hier auch wieder die Vorgabe `IP_NET_1`.

PRIVOXY_x_ACTIONDIR Diese Variable gibt den Ort an, an dem die Privoxy-Regelsätze (die Dateien *default.action* und *user.action*) auf dem Router liegen sollen. Der angegebene Pfad wird relativ zum Wurzelverzeichnis ausgewertet. Diese Variable kann für zwei Dinge verwendet werden:

Verlagern der Standardregeln auf permanenten Speicher Gibt man als Verzeichnis einen Ort ausserhalb der Ram-Disk an, werden die Standardregeln beim erstmaligen Booten dorthin kopiert und dann von diesem Ort aus genutzt. Änderungen an diesen Regelsätzen überleben dann ein Reboot des Routers. Zu beachten ist, dass nach einem Update des Privoxy-Paketes diese Regeln immer noch Verwendung finden und evtl. mit dem aktuellen Paket kommende neuere Regelsätze ignoriert werden.

Verwenden eigener Regelsätze fl4l gestattet das Überschreiben der Standardregeln mit nutzerspezifischen Regeln. Dazu legt man im *config*-Verzeichnis ein Unterverzeichnis an (z.B. *etc/my_privoxy*; es darf nicht *etc/privoxy* heissen) und legt dort die eigenen Regeln ab.

Das Setzen dieser Variable ist optional.

PRIVOXY_x_HTTP_PROXY Möchte man zusätzlich zu Privoxy einen weiteren HTTP Proxy verwenden, der dann z.B. auch Webseiten zwischenspeichert, so kann man den hier angeben. Privoxy bedient sich dann dieses Proxys. So kann man die Vorteile mehrerer Proxys nutzen. Ein Eintrag könnte so aussehen:

```
PRIVOXY_1_HTTP_PROXY='mein.provider.de:8000'
```

Die Angabe ist optional.

PRIVOXY_x SOCKS_PROXY Möchte man zusätzlich zu Privoxy einen weiteren SOCKS Proxy verwenden, kann man den hier angeben. Um die Privatsphäre weiter zu erhöhen kann der Datenverkehr vom Privoxy beispielsweise durch das Tor Netzwerk geschickt werden. Für weitere Details zu Tor lesen Sie in der [Tor Dokumentation](#) (Seite 6) weiter. Ein Eintrag für die Nutzung von Tor könnte so aussehen:

```
PRIVOXY_x SOCKS_PROXY='127.0.0.1:9050'
```

Die Angabe ist optional.

PRIVOXY_x_TOGGLE Diese Option aktiviert die Möglichkeit, den Proxy über das Webinterface ein- bzw. auszuschalten. Wird Privoxy ausgeschaltet, wirkt er als einfacher Forwarding-Proxy und ändert den Inhalt der übertragenen Seiten in keinsten Weise. Es

ist zu beachten, dass diese Einstellung für ALLE Benutzer des Proxys gilt, d.h. wenn ein Benutzer Privoxy abschaltet, ist Privoxy auch für die anderen Nutzer nur noch ein Weiterleitungs-Proxy.

PRIVOXY_x_CONFIG Diese Option ermöglicht den Benutzern des Proxys die interaktive Bearbeitung der Konfiguration über das Privoxy-Webinterface. Für weitere Details bitte ich auch hier, die Privoxy-Dokumentation zu konsultieren.

PRIVOXY_x_LOGDIR Mit dieser Option kann ein Logverzeichnis für Privoxy angegeben werden. Dies kann z.B. dann sinnvoll sein, wenn Website-Zugriffe der Benutzer geloggt werden sollen. Wird hier nichts angegeben (Standard), dann loggt nur die wichtigsten Meldungen auf die Konsole und ignoriert **PRIVOXY_LOGLEVEL**.

Hier kann auch 'auto' eingetragen werden, was den Log-Pfad auf das System-Verzeichnis für persistente Daten verlegt. Bitte darauf achten, daß in diesem Fall **FLI4L_UUID** korrekt konfiguriert wird, da mit großen Datenmengen zu rechnen ist und sonst /boot oder gar die Ram-Disk gefüllt wird.

PRIVOXY_x_LOGLEVEL Diese Option gibt an, was Privoxy in die Logdatei schreiben soll. Folgende Werte sind möglich, sie können durch Leerstelle getrennt angegeben werden, man kann sie aber auch addieren.

Wert	Was wird geloggt?
1	Jeden Request (GET/POST/CONNECT) ausgeben.
2	Status jeder Verbindung ausgeben
4	I/O-Status anzeigen
8	Header-Parsing anzeigen
16	Alle Daten loggen
32	Force-Feature debuggen
64	reguläre Ausdrücke debuggen
128	schnelle Weiterleitungen debuggen
256	GIF De-Animation debuggen
512	Common Log Format (zur Logfile-Analyse)
1024	Popup-Kill-Funktion debuggen
2048	Zugriffe auf den eingebauten Webserver loggen
4096	Startmeldungen und Warnungen
8192	Nicht-fatale Fehler

Um eine Logdatei im Common Logfile Format zu erstellen, sollte NUR der Wert 512 angegeben werden, da sonst die Logdatei durch andere Meldungen "verschmutzt" wird und somit nicht mehr problemlos ausgewertet werden kann.

Privoxy bietet sehr viele Konfigurationsmöglichkeiten. Diese können aber aus verständlichen Gründen nicht alle durch die Konfigurations-Datei von fli4l abgedeckt werden. Sehr viele dieser Optionen können im Webinterface von Privoxy eingestellt werden. Genauere Infos über den Aufbau dieser Dateien gibt es auf der Privoxy-Homepage. Die Konfigurationsdateien von Privoxy liegen unter <fli4l-Verzeichnis>/opt/etc/privoxy/. Es handelt sich hierbei um die Original-Dateien aus dem Privoxy-Paket, allerdings wurden, um Platz zu sparen, alle Kommentare entfernt.

1.1.2 OPT_TOR - Ein anonymes Kommunikationssystem für das Internet

Tor ist ein Werkzeug für eine Vielzahl von Organisationen und Menschen, die ihren Schutz und ihre Sicherheit im Internet verbessern wollen. Die Nutzung von Tor hilft Ihnen, das Browsen und Veröffentlichen im Web, Instantmessaging, IRC, SSH und anderen TCP basierende Anwendungen zu anonymisieren. Weiterhin bietet Tor eine Plattform auf der Softwareentwickler neue Anwendungen schaffen können die zu mehr Anonymität, Sicherheit und zum Schutz der Privatsphäre beitragen.

<https://www.torproject.org/index.html.de>

TOR_LISTEN_N

TOR_LISTEN_x Hier werden die IP-Adressen oder symbolischen Namen inklusive der Portnummer der Interfaces angegeben, auf denen Tor auf Verbindungen von Clients horchen soll. Es ist eine gute Idee, hier nur die Adressen der Interfaces anzugeben, denen man vertraut, da alle Rechner vollen Zugriff auf Tor haben (und auf den eventuell aktivierten Konfigurations-Editor). In der Regel ist die Vorgabe `IP_NET_1_IPADDR:9050` sinnvoll.

Auf hier angegebenen Adressen lauscht Tor und bietet seine Dienste an. 9050 ist der Standard-Port. Die Angabe hier muss man dann bei der Proxy-Konfiguration des jeweils verwendeten Programms benutzen.

Als Proxy beim jeweiligen Programm muss der fli4l-Rechner angegeben werden, also das, was man bei `HOSTNAME='fli4l'` angegeben hat bzw. dessen IP (z.B 192.168.6.1), die man bei `HOST_x_IP='192.168.6.1'` angegeben hat. Zusammen mit dieser Port-Angabe hier hat man dann alle nötigen Daten, um sein Programm für die Nutzung von Tor zu konfigurieren.

TOR_ALLOW_N Gibt die Anzahl der Listeneinträge an.

TOR_ALLOW_x Die Liste der Netze und/oder IP-Adressen für die der Paketfilter geöffnet wird. Sinnvoll ist hier auch wieder die Vorgabe `IP_NET_1`.

TOR_CONTROL_PORT Hier kann angegeben werden auf welchem TCP Port Tor einen Kontrollzugang über das Tor Control Protocol öffnen soll. Die Angabe ist optional. Wird nichts angegeben wird diese Funktion deaktiviert.

TOR_CONTROL_PASSWORD Hier kann ein Passwort für den Kontrollzugang angegeben werden.

TOR_DATA_DIR Diese Angabe ist optional. Wird nichts angegeben, wird der Standardordner `/etc/tor` verwendet

TOR_HTTP_PROXY Soll Tor die Anfragen an einen HTTP-Proxy weiterleiten, kann man den hier angeben. Tor bedient sich dann dieses Proxys. So kann man die Vorteile mehrerer Proxys nutzen. Ein Eintrag könnte so aussehen:

```
TOR_HTTP_PROXY='mein.provider.de:8000'
```

Die Angabe ist optional.

TOR_HTTP_PROXY_AUTH Eine eventuell notwendige Authentifizierung für den Proxy kann hier in der Form Benutzername:Passwort eingetragen werden.

TOR_HTTPS_PROXY Hier kann ein HTTPS-Proxy eingetragen werden. Siehe dazu auch [TOR_HTTP_PROXY](#).

TOR_HTTPS_PROXY_AUTH Siehe dazu [TOR_HTTP_PROXY_AUTH](#).

TOR_LOGLEVEL Diese Option gibt an, was Tor in die Logdatei schreiben soll. Folgende Werte sind möglich: debug, info, notice, warn oder err. Die Werte debug und info sollten aus Sicherheitsgründen möglichst nicht verwendet werden.

TOR_LOGFILE Falls Tor statt ins syslog in eine Datei loggen soll, kann diese hier angegeben werden.

Hier kann auch 'auto' eingetragen werden, was den Log-Pfad auf das System-Verzeichnis für persistente Daten verlegt. Bitte darauf achten, daß in diesem Fall FLI4L_UUID korrekt konfiguriert wird, da mit großen Datenmengen zu rechnen ist und sonst /boot oder gar die Ram-Disk gefüllt wird.

1.1.3 OPT_SS5 - Ein Socks4/5 Proxy

Für einige Programme wird ein Socks-Proxy benötigt. SS5 stellt diese Funktionalität bereit. <http://ss5.sourceforge.net/>

SS5_LISTEN_N

SS5_LISTEN_x Hier werden die IP-Adressen oder symbolischen Namen inklusive der Portnummer der Interfaces angegeben, auf denen SS5 auf Verbindungen von Clients horchen soll. Es ist eine gute Idee, hier nur die Adressen der Interfaces anzugeben, denen man vertraut, da alle Rechner vollen Zugriff auf SS5 haben (und auf den eventuell aktivierten Konfigurations-Editor). In der Regel ist die Vorgabe IP_NET_1_IPADDR:8050 sinnvoll.

Auf hier angegebenen Adressen lauscht SS5 und bietet seine Dienste an. 8050 ist der Standard-Port. Die Angabe hier muss man dann bei der Proxy-Konfiguration des jeweils verwendeten Programms benutzen.

Als Proxy beim jeweiligen Programm muss der fli4l-Rechner angegeben werden, also das, was man bei HOSTNAME='fli4l' angegeben hat bzw. dessen IP (z.B. 192.168.6.1), die man bei HOST_x_IP='192.168.6.1' angegeben hat. Zusammen mit dieser Port-Angabe hier hat man dann alle nötigen Daten, um sein Programm für die Nutzung von SS5 zu konfigurieren.

SS5_ALLOW_N Gibt die Anzahl der Listeneinträge an.

SS5_ALLOW_x Die Liste der Netze und/oder IP-Adressen für die der Paketfilter geöffnet wird. Sinnvoll ist hier auch wieder die Vorgabe IP_NET_1.

1.1.4 OPT_TRANSProxy (EXPERIMENTELL) - Transparenter HTTP-Proxy

Transproxy ist ein „transparenter“ Proxy, also ein Programm, dass es ermöglicht, alle HTTP-Abfragen, die über den Router laufen, abzufangen und an einen normalen HTTP-Proxy, z.B. Privoxy, weiterzuleiten. Um diese Funktionalität zu erreichen, muss der Paketfilter HTTP-Anfragen, die eigentlich ins Internet gehen sollen, an Transproxy weiterreichen, welcher diese weiter aufbereitet und an den anderen HTTP-Proxy weitergibt. iptables bietet zur Unterstützung dieser Funktion die Aktion „REDIRECT“:

```
PF_PREROUTING_1='tpr: http IP_NET_1 REDIRECT:8081'
```

Diese Regel würde alle HTTP-Pakete aus dem ersten definierten Netz (normalerweise das interne LAN) an Transproxy auf Port 8081 weiterleiten.

TRANSProxy_LISTEN_N

TRANSProxy_LISTEN_x Hier werden die IP-Adressen oder symbolischen Namen inklusive der Portnummer der Interfaces angegeben, auf denen Transproxy auf Verbindungen von Clients horchen soll. Hier müssen alle Interfaces angegeben werden, für die im Paketfilter Pakete auf Transproxy umgelenkt werden. Mit der Vorgabeeinstellung `any:8081` hört Transproxy auf allen Interfaces.

TRANSProxy_TARGET_IP

TRANSProxy_TARGET_PORT Mit diesen Optionen wird festgelegt, an welchen Dienst eingehende HTTP-Anfragen umgeleitet werden. Dies kann ein beliebiger Standard-HTTP-Proxy (Squid, Privoxy, Apache, etc.) auf einem beliebigen anderen Rechner (oder auch auf fl4l selbst) sein. Hier ist darauf zu achten, dass der Proxy sich nicht im Bereich der durch den Paketfilter umgeleiteten HTTP-Anfragen befindet, da sonst eine Schleife entsteht.

TRANSProxy_ALLOW_N

TRANSProxy_ALLOW_x Die Liste der Netze und/oder IP-Adressen für die der Paketfilter geöffnet wird. Dies sollte die gleichen Netze abdecken, die auch im Paketfilter umgeleitet werden. Werden hier keine Bereiche angegeben, müssen die Angaben von Hand in der Paketfilter-Konfiguration vorgenommen werden.

1.1.5 OPT_SIPROXD (experimentell) – Proxy für Session Initiation Protocol

Möchte man mehrere SIP-Anwendungen (egal ob Ekiga, x-lite oder Hardware SIP-Telefone) hinter einem Router betreiben, so kann es vorkommen, dass Netzwerk-Ports speziell weitergeleitet werden müssen, da sonst die Verbindungen nicht so funktionieren, wie sie sollten.

Um dies zu vermeiden, kann man einen speziellen SIP Proxy-Server nutzen. Es werden derzeit (fl4l V4.0.0) mehrere solche Proxys evaluiert. Sollte jemand über eine Empfehlung verfügen so darf er sich gerne an das Team wenden!

Konfiguration

SIPROXD_N Gibt die Anzahl der Siproxd-Instanzen an, die gestartet werden sollen. Mehrere Instanzen können nötig sein, wenn man SIP-Clients aus verschiedenen hinter dem fli4l-Router liegenden Subnetzen benutzen will.

Standard-Einstellung: `SIPROXD_N='0'`

Beispiel: `SIPROXD_N='2'`

SIPROXD_x_DEV_IN Hier ist die Netzwerk-Schnittstelle zum lokalen Netz bzw. LAN hin anzugeben, vom dem aus sich die VoIP-Clients mit dem SIP-Server verbinden. Erlaubt sind sowohl Namen von Netzwerkinterfaces (inklusive VLAN-Schnittstellen) als auch fli4l-Variablen wie bspw. `IP_NET_1_DEV`.

Beispiel: `SIPROXD_1_DEV_IN='br0'`

SIPROXD_x_DEV_OUT Die Netzwerkschnittstelle zum Internet hin, über die der SIP-Server erreicht wird, also beispielsweise das selbe Interface wie der default circuit.

Beispiel: `SIPROXD_1_DEV_OUT='eth0.23'`

SIPROXD_x_TRANSPARENT Ist diese Option gesetzt, werden dem Paketfilter automatisch Regeln hinzugefügt, die den Betrieb als transparenten Proxy erlauben. Das heißt die SIP-Client Software oder das VoIP-Gerät müssen nicht explizit die Konfiguration eines Proxy zulassen, sondern funktionieren direkt so, als wären sie ohne NAT mit dem SIP-Server verbunden. Ermöglicht beispielsweise die Verwendung der FRITZ!App Fon, wenn der fli4l-Router hinter einer FRITZ!Box betrieben wird.

Beispiel: `SIPROXD_1_TRANSPARENT='yes'`

Diese Variable ist optional. Ist sie nicht gesetzt, bleibt die Funktion deaktiviert.

SIPROXD_x_ALLOW_REG_N Anzahl der lokalen IP-Netze, aus denen Client-Registrierung erlaubt werden soll.

Beispiel: `SIPROXD_1_ALLOW_REG_N='2'`

SIPROXD_x_ALLOW_REG_x Ein lokales Netz, aus dem der Zugriff auf den Proxy für VoIP-Clients erlaubt wird. Im Gegensatz zu `SIPROXD_x_DEV_IN` werden hier IP-Netze angegeben, die an das lokale Interface gebunden sind bzw. über dieses erreicht werden können. Erlaubt sind neben direkten Angaben wie `192.168.42.0/24` auch Variablen der Form `IP_NET_1`.

Beispiel: `SIPROXD_1_ALLOW_REG_1='IP_NET_1'`

SIPROXD_x_SIP_PORT Port auf dem der SIP-Proxy auf eingehende SIP-Nachrichten lauscht. Der Standard-Port muss hier nur geändert werden, wenn mehr als eine Instanz des siproxd betrieben wird.

Beispiel: `SIPROXD_1_SIP_PORT='5060'`

SIPROXD_x_RTP_PORT_MIN Unteres Ende des Port-Bereichs, der für eingehende RTP-Stream-Verbindungen benutzt wird.

Beispiel: `SIPROXD_1_RTP_PORT_MIN='7070'`

SIPROXD_x_RTP_PORT_MAX Oberes Ende des Port-Bereichs für eingehenden RTP-Verkehr.

Beispiel: `SIPROXD_1_RTP_PORT_MAX='7089'`

SIPROXD_x_USER_N Ist dieser Wert größer null, müssen sich alle Nutzer gegenüber dem siproxd mit Nutzernamen und Passwort authentifizieren.

Beispiel: `SIPROXD_1_USER_N='1'`

Diese Variable ist optional. Ist sie nicht gesetzt, wird der Wert '0' angenommen.

SIPROXD_x_USER_x_NAME Nutzernamen des jeweiligen Nutzerkontos, das sich am siproxd authentifizieren muss.

Beispiel: `SIPROXD_1_USER_1_NAME='alice'`

Diese Variable ist optional.

SIPROXD_x_USER_x_PASS Passwort des jeweiligen Nutzerkontos, das sich am siproxd authentifizieren muss.

Beispiel: `SIPROXD_1_USER_1_PASSWD='batteryhorsestaple'`

Diese Variable ist optional.

Beispiel

```
OPT_SIPROXD='yes'
```

```
...
```

1.1.6 OPT_IGMPProxy - Proxy für Internet Group Management Protocol

Die Deutsche Telekom AG bietet mittlerweile seit einigen Jahren VDSL25/50 (Bandbreite: 25/50 MBit/s) zusammen mit Entertain-Paketen an. Damit besteht die Möglichkeit, Fernsehen über Internet (IPTV) zu empfangen.

Die Verteilung von IPTV erfolgt als Multicast, d.h. von einem Punkt zu einer (geschlossenen) Gruppe. Zur Organisation von Multicast-Gruppen ist das Netzwerkprotokoll IGMP (Internet Group Management Protocol) notwendig. IGMP (<http://de.wikipedia.org/wiki/IGMP>) bietet die Möglichkeit, dynamisch Multicast-Gruppen zu verwalten. Die Verwaltung findet nicht in der Sende-Station statt, sondern in den Routern, an denen Empfänger einer Multicast-Gruppe direkt angeschlossen sind. IGMP bietet Funktionen, mit denen eine Station einem Router mitteilt, dass sie Multicast-IP-Pakete einer bestimmten Multicast-Gruppe empfangen will.

Die mitgelieferten Speedport-Router (derzeit W700V/W701V/W722) unterstützen IGMP. Wer fli4l für IPTV statt Speedport-Router nutzen will, benötigt einen IGMP-Proxy (<http://sourceforge.net/projects/igmpproxy/>) auf dem fli4l-Router. OPT_IGMPProxy ist ein IGMP-Proxy für fli4l.

Diese Dokumentation zum OPT_IGMP Paket beschreibt die Konfiguration von fli4l, um VDSL und IPTV mit der mitgelieferten Set-Top-Box (STB) X300T/X301T bzw. MR-303 hinter einem fli4l-Router zu betreiben. In dieser Beschreibung erfolgt die Installation von IPTV über eine zusätzliche Netzwerkkarte.

Voraussetzung

Die Deutsche Telekom hat VDSL als VLAN eingeführt. In der Einführungsphase (Startnetz) wurde nur ein VLAN-Tag (ID7) verwendet, über den der gesamte Traffic floss. Nach der Umstellung (Zielnetz) auf zwei VLAN-Tags (ID7, ID8) bleibt der Internet Traffic auf ID7 und der neue ID8 wird ausschließlich für den IPTV Multicast-Traffic verwendet. Die Umstellung des VDSL Betriebs auf das Zielnetz (zwei VLAN Tags ID7/ID8) ist nach derzeitigem Stand größtenteils abgeschlossen.

Hardware (neben Set-Top-Box und VDSL-Modem):

- HW für fli4l: Für VDSL 25/50 sollte es besser kein 486er mehr sein. Falls es zu Bild-/Tonstörungen kommt kann das daran liegen, dass die eingesetzte HW zu wenig Leistung hat.
- High-End-NICs (Beispiele: 3Com, Intel Pro100). Realtek-Chipsätze stellen eher das Low-End-Spektrum dar.

Software:

- Paket: `advanced_networking`
- Paket: `dhcp_client` (für Zielnetz und Verwendung von ID8)

Die Anpassung der Konfigurationsdateien (`base.txt`, `dsl.txt`, `advanced_networking.txt`, `dhcp_client.txt`, `dns_dhcp.txt`) wird im Folgenden beschrieben.

Hardware Setup

Die Empfehlung für den Speedport-Router, die IPTV STB ohne weitere Netzwerk-Elemente direkt an den Router anzuschließen, gilt natürlich auch für den fli4l-Router. Falls dennoch Netzwerk-Knoten (Hub, Switch, Bridge, Gateway, Router) zwischen IPTV Box und Router geschaltet werden, sollten diese multicastfähig sein, um Störungen zu vermeiden.

Im Heimnetz werden i.d.R. keine Switches verwendet, die virtuelle Netze (VLAN) voneinander trennen, um den restlichen Verkehr (ID7) vom IPTV Multicast-Traffic (ID8) zu entlasten.

Deshalb wird hier als HW-Konfiguration eine separate NIC (Network Interface Card = LAN- bzw. Ethernet-Karte) im fli4l verwendet, um die Set-Top-Box (STB) direkt mit dem fli4l zu verbinden und das restliche Heimnetz vom Multicast-Traffic zu entlasten und alle o.g. Probleme auszuschließen. Wer die 'Single'-NIC-Methode bevorzugt sollte selbst wissen was er tut (das wird hier nicht weiter beschrieben).

Anbei ein Diagramm, wie im genannten Beispiel der fli4l-Router vom Standard-Router zum Router mit 3 NIC's migriert wird:

- Standard-Konfiguration:
 - `eth0` wird als NIC für das interne home/office LAN in `base.txt` eingetragen
 - In `dsl.txt` wird als DSL-Interface `eth1` angegeben
- Erweiterte Konfiguration mit zusätzlicher IPTV NIC:
 - Nach Einbau der zusätzlichen NIC in den fli4l-Router wird diese in `base.txt` als `eth2` eingetragen.

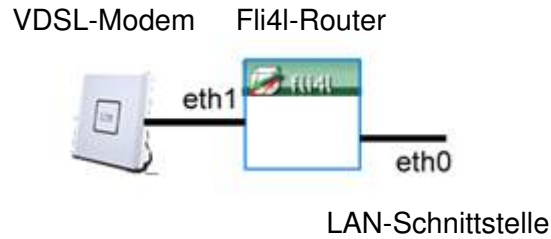


Abbildung 1.1: Fli4l in der Standardkonfiguration

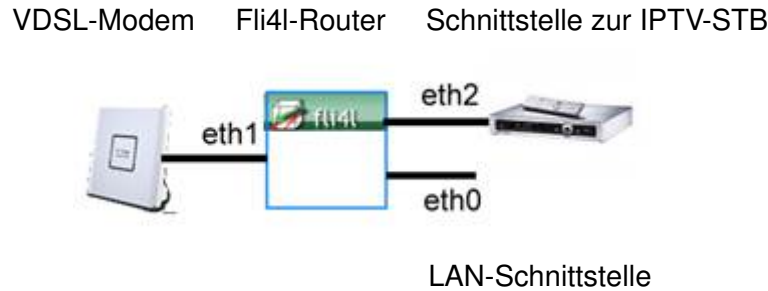


Abbildung 1.2: Fli4l in der IPTV-Konfiguration

VLAN-Konfiguration

Eines vorweg: OPT_IGMP ist nicht auf VLAN angewiesen. Vielmehr wird VLAN derzeit von der Deutschen Telekom für VDSL verwendet und muss dafür vom Router unterstützt werden. Ob VLAN für den Internet-Betrieb auch bei anderen Providern (Arcor, Alice, etc...) benötigt wird, ist uns nicht bekannt.

Um VDSL25/50 von T-Home für den Internet-Betrieb zum Laufen zu bringen, muss die NIC zum VDSL-Modem zwingend als VLAN-Interface konfiguriert werden - siehe auch (<http://www.fli4l.de/fileadmin/doc/deutsch/html/fli4l-3.4.0/node30.html>).

Ein Hinweis für alle, die nur das 'normale DSL' der Telekom, also ADSL, ADSL2, ADSL2+ haben: VLAN wird nur von VDSL benötigt, nicht aber vom 'normalen DSL'. Die VLAN-Konfiguration wird deshalb mit dem 'normalen DSL' nicht funktionieren.

Bei dem Vorhandensein von zwei VLAN-Tags (Zielnetz, siehe oben) wird der traffic wie folgt aufgeteilt:

- VLAN ID7: Internet-Traffic
- VLAN ID8: IPTV Multicast-Traffic

Damit läuft Internet-Traffic unabhängig vom IPTV-Traffic. Wesentlicher Unterschied ist, dass für VLAN ID7 eine PPPoE-Einwahl erforderlich ist. VLAN ID8 wird über einen DHCP-Server ohne Einwahl zur Verfügung gestellt. In dieser Zielarchitektur gibt es keine Zwangstrennung nach 24h mehr.

Für VLAN ist folgende Konfiguration erforderlich (NICs wie im Abschnitt Hardware-Setup angegeben):

advanced__networking.txt

```
VLAN_DEV_N='2'
VLAN_DEV_1_DEV='eth1'      # interface of VDSL-Modem; example: eth1
                           # In unserem Beispiel geht 'eth1' zum VDSL-Modem
VLAN_DEV_1_VID='7'         # ID7 to support VLAN for internet
VLAN_DEV_2_DEV='eth1'      # interface of VDSL-Modem; example: eth1
                           # In unserem Beispiel geht 'eth1' zum VDSL-Modem
VLAN_DEV_2_VID='8'         # ID8 to support VLAN for IPTV
```

Die Virtual-NIC eth1.7 muss in die DSL-Konfiguration eingetragen werden:

dsl.txt

```
PPPOE_ETH='eth1.7'        # eth<nummer der karte zum vdsl-modem>.7'
                           # Bsp 'eth1.7'
```

Für die Virtual-NIC eth1.8 benötigen wir einen dhcp_client, da VLAN ID8 über einen DHCP-Server ohne Einwahl zur Verfügung gestellt wird.

dhcp_client

```
OPT_DHCP_CLIENT='yes'
DHCP_CLIENT_TYPE='dhcpcd'
DHCP_CLIENT_INTERFACES='IP_NET_3_DEV' # listen on interface eth1.8
DHCP_CLIENT_USEPEERDNS='no'
DHCP_CLIENT_HOSTNAME=''
```

Seit Fli4l V3.3 kann für das Interface nicht mehr **eth1.8** angegeben werden, sondern es muss der Eintrag **IP_NET_x_DEV** verwendet werden, der für das Interface in base.txt definiert wurde; hier **IP_NET_3_DEV**.

Optional:

Falls die verwendete NIC mit der MTU-Größe Probleme hat, muss der MTU-Wert über den Parameter **DEV_MTU** angepasst werden. Im Test zeigten Intel Pro/100 (e100) und auch eine 3-Com-Karte keine Probleme, andere User berichten, dass bei der 3Com '3c59x' der MTU-Wert auf 1496 angepasst werden muss.

```
DEV_MTU_1=''              # Adjust MTU size of NIC on VDSL-Modem
                           # Example: DEV_MTU_1='eth1 1496'
```

Jetzt sind noch die Konfigurationsdateien base.txt und dns_dhcp.txt anzupassen, wie im nächsten Kapitel beschrieben.

Konfiguration einer zusätzlichen NIC für IPTV

In base.txt und dns_dhcp.txt muss die Konfiguration für VLAN und für die zweite NIC angepasst werden.

Zweite NIC für IPTV eintragen:

```
NET_DRV_N='2'
NET_DRV_1='via-rhine'      # 1. NIC für als LAN-Schnittstelle
NET_DRV_2='3c59x'         # 2. NIC - hier 3Com für IPTV SetTopBox
```

Jetzt muss der Adressraum für die zweite NIC festgelegt werden. Hier wird fürs LAN 192.168.2.0/24 und für die zweite NIC 192.168.3.0/24 verwendet. Außerdem werden Einträge für die Virtual-NICs eth1.7 und eth1.8 benötigt:

```
IP_NET_N='4'
IP_NET_1='192.168.2.1/24'      # home/office LAN
IP_NET_1_DEV='eth0'
IP_NET_2='192.168.3.1/24'      # iptv LAN
IP_NET_2_DEV='eth2'
IP_NET_3='dhcp'               # dhcp client - IP ueber dhclient
IP_NET_3_DEV='eth1.8'
IP_NET_3_MAC='00:40:63:da:cf:32' # neue MAC/nicht MAC von eth1
IP_NET_4='dhcp'               # eth1.7 zum modem
IP_NET_4_DEV='eth1.7'
IP_NET_4_MAC='00:40:63:da:cf:33' # neue MAC/nicht MAC von eth1
```

Wichtig ist auch die Änderung die MAC-Adressen für eth1.7 und eth1.8, welche nicht mit eth1 übereinstimmen dürfen, da sonst – abhängig vom VDSL-Net der DTAG – ggf. Störungen nach der Zwangstrennung auftreten können.

Für die neue NIC muss der Zugriff auf das Internet natürlich genauso funktionieren, wie für die erste NIC. Dazu sind weitere Einstellungen notwendig:

```
PF_INPUT_1='IP_NET_1 ACCEPT'
PF_INPUT_2='IP_NET_2 ACCEPT'
PF_INPUT_3='any 224.0.0.0/4 ACCEPT'
[...]
PF_FORWARD_3='any 224.0.0.0/4 ACCEPT'
PF_FORWARD_5='IP_NET_1 ACCEPT'
PF_FORWARD_6='IP_NET_2 ACCEPT'
[...]
PF_POSTROUTING_1='IP_NET_1 MASQUERADE'
PF_POSTROUTING_2='IP_NET_2 MASQUERADE'
```

Damit später auch eine dynamische DHCP-Adressierung an der neuen IPTV-NIC klappt und die SetTop-Box mit einem Namen angesprochen werden kann, sind noch folgende Einstellungen in dns_dhcp.txt erforderlich:

```
HOST_10_NAME='igmp'
HOST_10_IP4='192.168.3.1'
HOST_11_NAME='iptv'
HOST_11_IP4='192.168.3.4'
HOST_11_MAC='00:D0:E0:93:49:34'      # MAC Adr T-Home X300T
[...]
DHCP_RANGE_2_NET='IP_NET_2'
```

```
DNSDHCP_RANGE_2_START='192.168.3.10'  
DNSDHCP_RANGE_2_END='192.168.3.20'  
DNSDHCP_RANGE_2_DNS_SERVER1=''  
DNSDHCP_RANGE_2_DNS_SERVER2=''  
DNSDHCP_RANGE_2_NTP_SERVER=''  
DNSDHCP_RANGE_2_GATEWAY=''
```

Am Besten ist es, nach der Konfiguration der neuen NIC, diese erst einmal an einen PC zu hängen um zu sehen ob man über die neue NIC auch ins Internet kommt. Ist der Test erfolgreich, sollte die neue zweite NIC richtig konfiguriert sein.

IGMP-Funktion

Beim Booten des fli4l-Routers werden die Parameter der config-Datei proxy.txt in die Konfigurationsdatei /etc/igmpproxy.conf geschrieben, welche beim Start des Programms igmpproxy eingelesen werden.

Entgegen den früheren opt_igmp Versionen, wird der IGMP-Proxy jetzt einmalig beim Booten des Routers gestartet und läuft dann solange eine physikale Verbindung zum Internet besteht. Der IGMP-Proxy wird weder durch die 24h Zwangstrennung, noch durch ein manuelles trennen und verbinden des Internet-Traffics beeinflusst.

IGMP-Konfiguration

OPT_IGMPPROXY Mit 'yes' wird das IGMP Proxy Paket aktiviert. Die Einstellung 'no' deaktiviert das Paket komplett.

IGMPPROXY_DEBUG Mit 'yes' können Meldungen des IGMP Proxies ins syslog ausgegeben werden.

IGMPPROXY_DEBUG2 Mit 'yes' kann das Loglevel des IGMP Proxies erhöht werden.

IGMPPROXY_QUICKLEAVE_ON Mit Quickleave kann die Last im Upstream-Link gesenkt werden. Falls der Parameter mittels 'yes' eingeschaltet wird, führt dies dazu, dass der Multicast nach einem Kanalwechsel schneller abbestellt und so die Last im Downstream gesenkt wird, indem sich der IGMP-Proxy wie ein Receiver verhält.

Gibt es 2 STBen und sehen diese dasselbe Programm, dann kann es (mit Quickleave = yes) passieren, dass beim Umschalten des Programms von einer STB bei der zweiten STB das Programm unterbrochen wird. Beim Einsatz von nur einer STB kann Quickleave gefahrlos eingeschaltet werden (yes).

```
IGMPPROXY_QUICKLEAVE_ON='yes'      # Quickleave-Modus einschalten  
                                   # yes or no; Default: yes
```

IGMPPROXY_UPLOAD_DEV Für den IPTV-Betrieb benötigt der IGMP-Proxy ein Upstream- und ein Downstream-Interface. Das Upstream-Interface ist die Schnittstelle mit der NIC, an der das VDSL-Modem hängt. Diese sollte i.d.R. immer gleich bleiben.

Mit der Trennung von IPTV auf ID8 muss natürlich auch in der Konfiguration für den IGMP-Proxy eth1.8 statt bisher ppp0 eingetragen werden, womit die Umstellung Startnetz (nur ID7) auf das Zielnetz (mit ID7/8) komplett ist.

```
IGMPPROXY_UPLOAD_DEV='eth1.8'      # Upstream Interface; Default: ppp0
                                     # eth1.8 für T-Home/VDSL mit id7/id8
```

IGMPPROXY_DOWNLOAD_DEV Die Schnittstelle des Downstream-Interfaces (NIC zur IPTV SetTop-Box) ist hier abhängig von der HW-Konfiguration einzutragen. Für fli4l mit zweiter NIC – wie in diesem Dokument beschrieben – ist eth2 das Interface zur SetTop-Box.

```
IGMPPROXY_DOWNLOAD_DEV='eth2'      # Downstream Interface
```

IGMPPROXY_ALT_N Mit diesem Parameter wird die Anzahl der Adressbereiche für Multicast Streams festgelegt.

IGMPPROXY_ALT_x_NET Mit dem Parameter IGMPPROXY_ALT_x_NET werden Adressbereiche für Multicast-Traffic festgelegt, welche Ihren Ursprung außerhalb des Heim-Netzwerks haben, sowie der lokale Adressbereich, an dem die STB hängt.

```
IGMPPROXY_ALT_N='3'                # Anzahl der Multicast Sourcen
IGMPPROXY_ALT_1_NET='239.35.0.0/16' # IPTV streams - immer benoetigt
IGMPPROXY_ALT_2_NET='217.0.119.0/24' # Erforderlich fuer T-Home
IGMPPROXY_ALT_3_NET='193.158.34.0/23' # Erforderlich fuer T-Home
                                     # vor Mai 2013 '193.158.35.0/24'
# IGMPPROXY_ALT_4_NET='192.168.3.0/24' # Adressraum IPTV SetTop-Box/nicht
                                     # mehr notwendig
```

IGMPPROXY_WLIST_N Mit diesem Parameter wird die Anzahl der Whitelists für die IGMP Reports festgelegt.

IGMPPROXY_WLIST_x_NET :

Bei IGMPv3 können alle Adressen in einem Report zusammengefasst werden (<http://grinch.itg-em.de/entertain/artikel/zielnetzarchitektur-und-igmpproxy/>). Dieser wird dann komplett ignoriert, was dazu führt, dass der IGMP Querier irgendwann sämtlichen Multicasttraffic abschaltet, da er denkt, dieser würde nicht mehr benötigt. Um dies zu verhindern wurde die Konfiguration von whitelists erlaubt. Nur Multicastgruppen in dieser Liste werden dann auch auf WAN-Seite angefordert.

```
IGMPPROXY_WLIST_N='1'              # Anzahl der Multicast Sourcen
IGMPPROXY_WLIST_1_NET='239.35.0.0/16' # IPTV streams - immer benoetigt
                                     # siehe oben
```

Änderungen in anderen Config-Dateien

Seit Revision 32955 ist es nicht mehr notwendig, die Firewall für den IGMP Proxy und die Multicast Streams anzupassen, wenn in der base.txt die Standardregeln für die Firewall aktiviert sind (PF_INPUT_ACCEPT_DEF='yes' und PF_FORWARD_ACCEPT_DEF='yes'). Das Startskript fügt dann automatisch die notwendigen Regeln ein, wenn OPT_IGMPPROXY='yes' gesetzt ist.

Im Detail handelt es sich um zwei Regeln, die in der INPUT Kette eingetragen werden, damit die eingehenden Nachrichten den IGMP Proxy erreichen:


```
Chain INPUT (policy DROP 0 packets, 0 bytes)
pkts bytes target prot opt in out source destination
[...]
0 0 ACCEPT all -- * * 0.0.0.0/0 224.0.0.1 \
/* automatically added for IGMP Proxy */

0 0 ACCEPT all -- * * 0.0.0.0/0 224.0.0.22 \
/* automatically added for IGMP Proxy */
[...]
```

Und außerdem eine Regel in der FORWARD Kette, die erlaubt, dass die eingehenden Multicast Streams auch zum Media Receiver weitergeleitet werden:

```
Chain FORWARD (policy DROP 0 packets, 0 bytes)
pkts bytes target prot opt in out source destination
[...]
0 0 ACCEPT all -- * * 0.0.0.0/0 239.35.0.0/16 \
/* automatically added for IPTV streams */
[...]
```

Sofern man die Standardregeln nicht aktiviert sind, müssen manuell mindestens die folgenden Regeln eingefügt werden.

```
PF_INPUT_x='any 224.0.0.1/32 ACCEPT'
PF_INPUT_x='any 224.0.0.22/32 ACCEPT'
[...]
PF_FORWARD_x='any 239.35.0.0/16 ACCEPT'
```

Hinweis: Gegenüber früheren Versionen der Dokumentation wurden die Regeln auf die tatsächlich notwendigen Netze eingeschränkt. Sofern das IPTV nicht laufen sollte, nehmen wir gerne Hinweise auf zusätzlich notwendige Netze entgegen.

WICHTIG! Ab Ende Mai 2013 hat die Telekom für Entertain neue (classless) Routen eingeführt (<http://www.onlinekosten.de/forum/showthread.php?t=116415&page=38>). Der Grund dafür scheint wohl zu sein, das jetzt mehr als 256 Sender bzw. Adressen benutzt werden. Damit sendet der DHCP-Server dhcp-routen, die nicht mehr im bisherigen Subnetz enthalten sind. Solange die Telekom nicht ihr iptv-server-subnetz (193.158.34.0/23) wechselt, kann eine statische route fuer das vlan8-interface anlegt werden, um diese Änderung zu berücksichtigen, da sonst multicast nicht mehr funktioniert.

Lösung: In base.txt muss ein zusätzliche Route angegeben werden.

```
IP_ROUTE_N='1'
IP_ROUTE_1='193.158.34.0/23 eth1.8'
```

1.1.7 OPT_STUNNEL - Tunneln von Verbindungen über SSL/TLS

Das Programm "stunnel" erlaubt es, ansonsten unverschlüsselte Verbindungen in einem verschlüsselten SSL/TLS-Tunnel zu kapseln. Dies ermöglicht sicheren Datenaustausch über sonst unsichere Klartext-Protokolle. Auf Grund der Möglichkeiten des SSL/TLS-Protokolls sind verschiedene Formen der Client/Server-Validierung möglich.

Konfiguration

OPT_STUNNEL Diese Variable aktiviert die Unterstützung für SSL/TLS-Tunnel.

Standard-Einstellung: `OPT_STUNNEL='no'`

Beispiel: `OPT_STUNNEL='yes'`

STUNNEL_DEBUG Mit dieser Variable kann eingestellt werden, wie sehr “stunnel” seine Arbeitsweise protokolliert. Mögliche Einstellungen umfassen “yes” (alles wird protokolliert), “no” (Warnungen und Fehler werden protokolliert) oder ein Wert zwischen null und sieben, der die maximale Schwere der zu protokollierenden Meldungen angibt, wobei null für allerdingendste Meldungen und sieben für Debug-Meldungen steht. Die Einstellung “yes” entspricht der maximalen Schwere sieben, die Einstellung “no” entspricht der maximalen Schwere vier.

Standard-Einstellung: `STUNNEL_DEBUG='no'`

Beispiel 1: `STUNNEL_DEBUG='yes'`

Beispiel 2: `STUNNEL_DEBUG='5'`

STUNNEL_N Diese Variable konfiguriert die Anzahl der Tunnel-Instanzen. Jede Tunnel-Instanz “horcht” auf einem Netzwerkport “A”, verbindet sich bei einer eingehenden Verbindung mit einem anderen Netzwerkport “B” (der durchaus auch auf einer ganz anderen Maschine liegen kann) und leitet jeglichen Datenverkehr von “A” nach “B” weiter. Ob die Daten, die bei “A” ankommen, via SSL/TLS verschlüsselt sind, von “stunnel” entschlüsselt und dann nach “B” unverschlüsselt weitergeleitet werden oder umgekehrt, entscheidet sich durch die Variable `STUNNEL_x_CLIENT` (Seite 18).

Standard-Einstellung: `STUNNEL_N='0'`

Beispiel: `STUNNEL_N='2'`

STUNNEL_x_NAME Der Name des jeweiligen Tunnels. Er muss unter allen konfigurierten Tunneln eindeutig sein.

Beispiel: `STUNNEL_1_NAME='imond'`

STUNNEL_x_CLIENT Über diese Variable kann eingestellt werden, welche Teile der Kommunikation via SSL/TLS verschlüsselt werden. Es gibt zwei Möglichkeiten:

- *Client-Modus:* Der Tunnel erwartet von außen unverschlüsselte Daten und schickt diese verschlüsselt an das andere Ende des Tunnels. Dies entspricht der Einstellung `STUNNEL_x_CLIENT='yes'`.
- *Server-Modus:* Der Tunnel erwartet von außen via SSL/TLS verschlüsselte Daten und schickt diese entschlüsselt an das andere Ende des Tunnels. Dies entspricht der Einstellung `STUNNEL_x_CLIENT='no'`.

Tunnel im Client-Modus eignen sich also vor allem für Verbindungen, die “nach außen”, also z. B. ins (ungeschützte) Internet gehen, da die Daten vor dem Verlassen des lokalen Netzwerks verschlüsselt werden. Dafür muss die Gegenstelle aber natürlich auch einen Server anbieten, der via SSL/TLS verschlüsselte Daten erwartet. Beispielsweise kann so

ein E-Mail-Client im LAN, der nur unverschlüsseltes POP3 “spricht”, einen POP3-over-SSL-Dienst im Internet nutzen.¹

Tunnel im Server-Modus eignen sich umgekehrt für Verbindungen, die “von außen”, also z. B. aus dem (ungeschützten) Internet kommen, bei denen die Daten verschlüsselt ankommen. Wenn der eigentliche Dienst auf Server-Seite jedoch kein SSL/TLS versteht, müssen die Daten vorher entsprechend entschlüsselt werden. Beispielsweise kann so der Zugriff auf die Web-GUI des fli4l über via SSL/TLS verschlüsseltes HTTP (HTTPS) erfolgen, indem man auf dem fli4l einen Tunnel konfiguriert, der via SSL/TLS verschlüsselte HTTP-Daten auf Port 443 empfängt, diese entschlüsselt und dann an den lokalen Web-Server `mini_httpd`, der auf Port 80 horcht, weiterleitet.

Die Konfigurationen für diese Anwendungsfälle werden weiter hinten vorgestellt.

Beispiel: `STUNNEL_1_CLIENT='yes'`

STUNNEL_x_ACCEPT Hiermit wird festgelegt, auf welchem Port (und an welcher Adresse) der Tunnel auf eingehende Verbindungen “lauscht”. Es gibt prinzipiell zwei Möglichkeiten:

- Der Tunnel soll an *allen* Adressen (auf allen Schnittstellen) lauschen. Hierfür muss “any” verwendet werden.
- Der Tunnel soll nur an bestimmten Adressen lauschen. Dies wird mit Hilfe einer entsprechenden Referenz auf das konfigurierte IP-Subnetz eingestellt, beispielsweise `IP_NET_1_IPADDR` (für IPv4) oder `IPV6_NET_2_IPADDR` (für IPv6).

Des Weiteren *muss* hinter die Adresse der Port stehen, wobei der Port von der Adresse mit Hilfe eines Doppelpunktes (“:”) abgetrennt ist.

Beispiel 1: `STUNNEL_1_ACCEPT='any:443'`

Beispiel 2: `STUNNEL_1_ACCEPT='IP_NET_1_IPADDR:443'`

Beispiel 3: `STUNNEL_1_ACCEPT='IPV6_NET_2_IPADDR:443'`

Zu bedenken ist, dass die Verwendung von `IP_NET_x_IPADDR` bzw. `IPV6_NET_x_IPADDR` das Layer-3-Protokoll (IPv4 oder IPv6) festlegt; diese Wahl *muss* zu den Belegungen der Variablen `STUNNEL_x_ACCEPT_IPV4` und `STUNNEL_x_ACCEPT_IPV6` passen. Sie können also nicht IPv6 für den Tunnel mit Hilfe von `STUNNEL_1_ACCEPT_IPV6='no'` deaktivieren und dann mit Hilfe von `STUNNEL_1_ACCEPT='IPV6_NET_2_IPADDR:443'` an einer IPv6-Adresse lauschen; dies gilt analog für die umgekehrte Konstellation (`STUNNEL_1_ACCEPT_IPV4='no'` und die Verwendung von `IP_NET_x_IPADDR`). Des Weiteren hängt die Bedeutung von “any” von den aktivierten Layer-3-Protokollen (IPv4 oder IPv6) ab: Es wird natürlich nur auf Adressen gelauscht, die zu den via `STUNNEL_x_ACCEPT_IPV4` und `STUNNEL_x_ACCEPT_IPV6` aktivierten Layer-3-Protokollen gehören.

STUNNEL_x_ACCEPT_IPV4 Mit dieser Variable wird eingestellt, ob das IPv4-Protokoll für *eingehende* Verbindungen des Tunnels genutzt werden soll. Typischerweise ist dies der Fall, und diese Variable sollte den Wert “yes” enthalten. Die Belegung mit “no” stellt sicher, dass der Tunnel nur eingehende IPv6-Verbindungen akzeptiert. Dies erfordert jedoch eine valide IPv6-Konfiguration (siehe hierzu die Dokumentation des `ipv6`-Paketes).

Standard-Einstellung: `STUNNEL_x_ACCEPT_IPV4='yes'`

¹vgl. <http://de.wikipedia.org/wiki/POP3S>

Beispiel: STUNNEL_1_ACCEPT_IPV4='no'

STUNNEL_x_ACCEPT_IPV6 Analog zu STUNNEL_x_ACCEPT_IPV4 wird mit dieser Variable eingestellt, ob das IPv6-Protokoll für eingehende Verbindungen des Tunnels genutzt werden soll. Typischerweise ist das der Fall, wenn Sie die generelle Nutzung des IPv6-Protokolls mit Hilfe von OPT_IPV6='yes' aktiviert haben. Die Belegung mit "no" stellt sicher, dass der Tunnel nur eingehende IPv4-Verbindungen akzeptiert.

Standard-Einstellung: STUNNEL_x_ACCEPT_IPV6=<Wert von OPT_IPV6>

Beispiel: STUNNEL_1_ACCEPT_IPV6='no'

STUNNEL_x_CONNECT Hiermit wird festgelegt, welches Ziel der SSL/TLS-Tunnel hat. Es gibt prinzipiell drei Möglichkeiten, wobei bei jeder der drei Möglichkeiten noch ein mit ":" abgetrennter Port angehängt werden muss:

- Eine numerische IPv4- oder IPv6-Adresse.

Beispiel 1: STUNNEL_1_CONNECT='192.0.2.2:443'

- Der DNS-Name von einem internen Host.

Beispiel 2: STUNNEL_1_CONNECT='@webserver:443'

- Der DNS-Name von einem externen Host.

Beispiel 3: STUNNEL_1_CONNECT='@www.example.com:443'

Wird ein interner Host eingetragen, der sowohl eine IPv4- als auch eine IPv6-Adresse besitzt, dann wird die IPv4-Adresse bevorzugt. Wird ein externer Host eingetragen, der sowohl eine IPv4- als auch eine IPv6-Adresse besitzt, dann hängt das verwendete Layer-3-Protokoll davon ab, welche Adresse als erstes vom DNS-Resolver zurückgegeben wird.

STUNNEL_x_OUTGOING_IP Mit dieser optionalen Variable kann die *lokale* Adresse für die *ausgehende* Verbindung des Tunnels angegeben werden. Dies ist nur dann sinnvoll, wenn das Ziel des Tunnels über mehrere Schnittstellen (Routen) erreicht werden kann, also wenn man z. B. zwei Internet-Anbindungen nutzt. Normalerweise muss diese Variable nicht gesetzt werden.

Beispiel: STUNNEL_1_OUTGOING_IP='IP_NET_1_IPADDR'

STUNNEL_x_DELAY_DNS Wird diese optionale Variable auf "yes" gesetzt, so wird ein in STUNNEL_x_CONNECT verwendeter externer DNS-Name erst dann in eine Adresse umgewandelt, wenn die *ausgehende* Tunnelverbindung aufgebaut wird, also wenn der erste Client sich lokal mit der eingehenden Seite des Tunnels verbunden hat. Dies ist dann nützlich, wenn das Ziel des Tunnels ein Rechner ist, der nur über einen dynamischen DNS-Namen erreicht werden kann und die Adresse hinter diesem Namen häufiger wechselt, oder auch wenn eine aktive Einwahl bereits beim Starten von "stunnel" verhindert werden soll.

Standard-Einstellung: STUNNEL_x_DELAY_DNS='no'

Beispiel: STUNNEL_1_DELAY_DNS='yes'

STUNNEL_x_CERT_FILE Diese Variable enthält den Dateinamen des Zertifikats, das für den Tunnel verwendet werden soll. Für Server-Tunnel (STUNNEL_x_CLIENT='no') ist dies

das Server-Zertifikat, das vom Client ggfs. gegen eine “Certificate Authority” (CA) validiert wird. Für Client-Tunnel (`STUNNEL_x_CLIENT='yes'`) ist dies ein (in der Regel optionales) Client-Zertifikat, das vom Server ggfs. gegen eine CA validiert wird.

Das Zertifikat muss im so genannten PEM-Format vorliegen und muss unterhalb von `<config-Verzeichnis>/etc/ssl/stunnel/` abgespeichert werden. Nur der Dateiname muss in dieser Variable gespeichert werden, nicht der Pfad.

Für einen Server-Tunnel ist ein Zertifikat zwingend erforderlich!

Beispiel: `STUNNEL_1_CERT_FILE='myserver.crt'`

STUNNEL_x_CERT_CA_FILE Diese Variable enthält den Dateinamen des CA-Zertifikats, das für die Validierung des Zertifikats der Gegenstelle verwendet werden soll. Typischerweise validieren Clients das Zertifikat des Servers, andersherum ist dies jedoch genauso möglich. Einzelheiten zur Validierung lesen Sie bitte in der Beschreibung der Variable [STUNNEL_x_CERT_VERIFY](#) (Seite 21) nach.

Das Zertifikat muss im so genannten PEM-Format vorliegen und muss unterhalb von `<config-Verzeichnis>/etc/ssl/stunnel/` abgespeichert werden. Nur der Dateiname muss in dieser Variable gespeichert werden, nicht der Pfad.

Beispiel: `STUNNEL_1_CERT_CA_FILE='myca.crt'`

STUNNEL_x_CERT_VERIFY Diese Variable steuert die Validierung des Zertifikats der Gegenstelle. Es gibt fünf Möglichkeiten:

- *none*: Das Zertifikat der Gegenstelle wird überhaupt nicht validiert. In diesem Falle kann die Variable `STUNNEL_x_CERT_CA_FILE` leer bleiben.
- *optional*: Stellt die Gegenstelle ein Zertifikat zur Verfügung, so wird es gegen das CA-Zertifikat geprüft, das mit Hilfe der Variable `STUNNEL_x_CERT_CA_FILE` konfiguriert wird. Stellt die Gegenstelle *kein* Zertifikat zur Verfügung, so ist dies kein Fehler und die Verbindung wird dennoch akzeptiert. Diese Einstellung ist nur sinnvoll für Server-Tunnel, weil Client-Tunnel *immer* ein Zertifikat vom Server erhalten.
- *onlyca*: Das Zertifikat der Gegenstelle wird gegen das CA-Zertifikat geprüft, das mit Hilfe der Variable `STUNNEL_x_CERT_CA_FILE` konfiguriert wird. Sendet die Gegenstelle kein Zertifikat oder passt es nicht zur konfigurierten CA, wird die Verbindung abgewiesen. Dies ist nützlich, wenn eine eigene CA verwendet wird, da man dann alle potentiellen Gegenstellen kennt.
- *onlycert*: Das Zertifikat der Gegenstelle wird mit dem Zertifikat verglichen, das mit Hilfe der Variable `STUNNEL_x_CERT_CA_FILE` konfiguriert wird. Es wird *nicht* gegen ein CA-Zertifikat geprüft, sondern es wird sichergestellt, dass die Gegenstelle *genau* das passende (Server- oder Client-)Zertifikat sendet. Die Datei, die mit Hilfe der Variable `STUNNEL_x_CERT_CA_FILE` referenziert wird, enthält in diesem Fall also kein CA-, sondern ein Host-Zertifikat. Diese Einstellung stellt sicher, dass sich wirklich nur eine bestimmte und bekannte Gegenstelle verbinden darf (Server-Tunnel) bzw. eine Verbindung nur zu einer bekannten Gegenstelle (Client-Tunnel) aufgebaut wird. Dies ist für Peer-to-Peer-Verbindungen zwischen Hosts nützlich, die man beide unter Kontrolle hat, für die man aber keine eigene CA verwendet.

- *both*: Das Zertifikat der Gegenstelle wird mit dem Zertifikat verglichen, das mit Hilfe der Variable STUNNEL_x_CERT_CA_FILE konfiguriert wird, *und* es wird zusätzlich sichergestellt, dass es zu einem CA-Zertifikat passt. Die Datei, die mit Hilfe der Variable STUNNEL_x_CERT_CA_FILE referenziert wird, enthält in diesem Fall also *sowohl* ein CA- *als auch* ein Host-Zertifikat. Es handelt sich also um eine Kombination der Einstellungen *onlycert* und *onlyca*. Im Vergleich zur Einstellung *onlycert* werden somit Verbindungen abgelehnt, falls das CA-Zertifikat abgelaufen ist (auch wenn das Zertifikat der Gegenstelle ansonsten passt).

Standard-Einstellung: STUNNEL_x_CERT_VERIFY='none'

Beispiel: STUNNEL_1_CERT_VERIFY='onlyca'

Anwendungsbeispiel 1: Zugang zur fli4l-WebGUI via SSL/TLS

Mit diesem Beispiel wird die fli4l-WebGUI um einen SSL/TLS-Zugang erweitert.

```
OPT_STUNNEL='yes'
```

```
STUNNEL_N='1'
```

```
STUNNEL_1_NAME='http'
```

```
STUNNEL_1_CLIENT='no'
```

```
STUNNEL_1_ACCEPT='any:443'
```

```
STUNNEL_1_ACCEPT_IPV4='yes'
```

```
STUNNEL_1_ACCEPT_IPV6='yes'
```

```
STUNNEL_1_CONNECT='127.0.0.1:80'
```

```
STUNNEL_1_CERT_FILE='server.pem'
```

```
STUNNEL_1_CERT_CA_FILE='ca.pem'
```

```
STUNNEL_1_CERT_VERIFY='none'
```

Anwendungsbeispiel 2: Via SSL/TLS gesicherte Kontrolle von zwei entfernten fli4l-Routern via imonc

Mit diesem Beispiel werden die bekannten Schwachstellen des imonc/imond-Protokolls (Senden von Passwörtern im Klartext) für WAN-Verbindungen umgangen. (Die LAN-Verbindung zum Tunnel kann natürlich weiterhin abgehört werden!)

Konfiguration des lokalen fli4l im LAN (Client-Tunnel):

```
OPT_STUNNEL='yes'
```

```
STUNNEL_N='2'
```

```
STUNNEL_1_NAME='remote-imond1'
```

```
STUNNEL_1_CLIENT='yes'
```

```
STUNNEL_1_ACCEPT='any:50000'
```

```
STUNNEL_1_ACCEPT_IPV4='yes'
```

```
STUNNEL_1_ACCEPT_IPV6='yes'
```

```
STUNNEL_1_CONNECT='@remote1:50000'
```

```
STUNNEL_1_CERT_FILE='client.pem'
```

```
STUNNEL_1_CERT_CA_FILE='ca+server1.pem'
```

```
STUNNEL_1_CERT_VERIFY='both'
```

```
STUNNEL_2_NAME='remote-imond2'
```

```
STUNNEL_2_CLIENT='yes'  
STUNNEL_2_ACCEPT='any:50001'  
STUNNEL_2_ACCEPT_IPV4='yes'  
STUNNEL_2_ACCEPT_IPV6='yes'  
STUNNEL_2_CONNECT='@remote2:50000'  
STUNNEL_2_CERT_FILE='client.pem'  
STUNNEL_2_CERT_CA_FILE='ca+server2.pem'  
STUNNEL_2_CERT_VERIFY='both'
```

Konfiguration des ersten entfernten fli4l (Server-Tunnel):

```
OPT_STUNNEL='yes'  
STUNNEL_N='1'  
  
STUNNEL_1_NAME='remote-imond'  
STUNNEL_1_CLIENT='no'  
STUNNEL_1_ACCEPT='any:50000'  
STUNNEL_1_ACCEPT_IPV4='yes'  
STUNNEL_1_ACCEPT_IPV6='yes'  
STUNNEL_1_CONNECT='127.0.0.1:5000'  
STUNNEL_1_CERT_FILE='server1.pem'  
STUNNEL_1_CERT_CA_FILE='ca+client.pem'  
STUNNEL_1_CERT_VERIFY='both'
```

Konfiguration des zweiten entfernten fli4l (Server-Tunnel):

```
OPT_STUNNEL='yes'  
STUNNEL_N='1'  
  
STUNNEL_1_NAME='remote-imond'  
STUNNEL_1_CLIENT='no'  
STUNNEL_1_ACCEPT='any:50000'  
STUNNEL_1_ACCEPT_IPV4='yes'  
STUNNEL_1_ACCEPT_IPV6='yes'  
STUNNEL_1_CONNECT='127.0.0.1:5000'  
STUNNEL_1_CERT_FILE='server2.pem'  
STUNNEL_1_CERT_CA_FILE='ca+client.pem'  
STUNNEL_1_CERT_VERIFY='both'
```

Eine Verbindung zu dem entfernten “imond” wird aufgebaut, indem eine Verbindung zum lokalen fli4l auf Port 50000 (erster entfernter fli4l) bzw. 50001 (zweiter entfernter fli4l) initiiert wird. Dieser fli4l verbindet sich dann via SSL/TLS-Tunnel mit dem jeweiligen entfernten fli4l, der wiederum die Daten über eine dritte (Host-interne) Verbindung letztlich an den entfernten “imond” weiterleitet. Die Einstellungen für die Validierung stellen sicher, dass jeder fli4l jeweils nur den anderen fli4l als Verbindungspartner akzeptiert.

Abbildungsverzeichnis

1.1	Fli4l in der Standardkonfiguration	12
1.2	Fli4l in der IPTV-Konfiguration	12

Tabellenverzeichnis

Index

IGMPPROXY_ALT_N, 16
IGMPPROXY_ALT_x_NET, 16
IGMPPROXY_DEBUG, 15
IGMPPROXY_DEBUG2, 15
IGMPPROXY_DOWNLOAD_DEV, 16
IGMPPROXY_QUICKLEAVE_ON, 15
IGMPPROXY_UPLOAD_DEV, 15
IGMPPROXY_WLIST_N, 16
IGMPPROXY_WLIST_x_NET, 16

OPT_IGMPPROXY, 10, 15
OPT_PRIVOXY, 3
OPT_SIPROXD, 8
OPT_SS5, 7
OPT_STUNNEL, 17, 18
OPT_TOR, 6
OPT_TRANSPROXY, 8

PRIVOXY_MENU, 3
PRIVOXY_N, 3
PRIVOXY_x_ACTIONDIR, 4
PRIVOXY_x_ALLOW_N, 4
PRIVOXY_x_ALLOW_x, 4
PRIVOXY_x_CONFIG, 5
PRIVOXY_x_HTTP_PROXY, 4
PRIVOXY_x_LISTEN, 3
PRIVOXY_x_LOGDIR, 5
PRIVOXY_x_LOGLEVEL, 5
PRIVOXY_x_SOCKS_PROXY, 4
PRIVOXY_x_TOGGLE, 4

SIPROXD_N, 9
SIPROXD_x_ALLOW_REG_N, 9
SIPROXD_x_ALLOW_REG_x, 9
SIPROXD_x_DEV_IN, 9
SIPROXD_x_DEV_OUT, 9
SIPROXD_x_RTP_PORT_MAX, 9
SIPROXD_x_RTP_PORT_MIN, 9
SIPROXD_x_SIP_PORT, 9

SIPROXD_x_TRANSPARENT, 9
SIPROXD_x_USER_N, 10
SIPROXD_x_USER_x_NAME, 10
SIPROXD_x_USER_x_PASS, 10
SS5_ALLOW_N, 7
SS5_ALLOW_x, 7
SS5_LISTEN_N, 7
SS5_LISTEN_x, 7
STUNNEL_DEBUG, 18
STUNNEL_N, 18
STUNNEL_x_ACCEPT, 19
STUNNEL_x_ACCEPT_IPV4, 19
STUNNEL_x_ACCEPT_IPV6, 20
STUNNEL_x_CERT_CA_FILE, 21
STUNNEL_x_CERT_FILE, 20
STUNNEL_x_CERT_VERIFY, 21
STUNNEL_x_CLIENT, 18
STUNNEL_x_CONNECT, 20
STUNNEL_x_DELAY_DNS, 20
STUNNEL_x_NAME, 18
STUNNEL_x_OUTGOING_IP, 20

TOR_ALLOW_N, 6
TOR_ALLOW_x, 6
TOR_CONTROL_PASSWORD, 6
TOR_CONTROL_PORT, 6
TOR_DATA_DIR, 6
TOR_HTTP_PROXY, 6
TOR_HTTP_PROXY_AUTH, 6
TOR_HTTPS_PROXY, 7
TOR_HTTPS_PROXY_AUTH, 7
TOR_LISTEN_N, 6
TOR_LISTEN_x, 6
TOR_LOGFILE, 7
TOR_LOGLEVEL, 7
TRANSPROXY_ALLOW_N, 8
TRANSPROXY_ALLOW_x, 8
TRANSPROXY_LISTEN_N, 8

TRANSPROXY_LISTEN_x, [8](#)
TRANSPROXY_TARGET_IP, [8](#)
TRANSPROXY_TARGET_PORT, [8](#)