# Falcon Documentation

This document describes Falcon, its components and important facts.
This file can be downloaded here (html, gz, zip) or here (pdf).

## Contents



## 1. Basic Concepts

Falcon is the "Free application-level connection" kit. It’s a collection of different proxy-programs that use one clearly configfile.

Its main advantages are:

Falcon is structured. Because of its well thought out structure and its clearness it gives the Admin a sedative feeling and lets him sleep well at night.

Falcon is lightweight. There are no tons of command-line parameters, different configuration-options etc. Falcon minimizes complexity whilst maximizing flexibility.

Falcon is free. It’s released under the terms of the GNU
General Public License so everyone is very welcome to distribute source code to the project. Since it’s written in Perl, a free and easy to learn scripting-language, everyone should be possible to participate in the project.

The concept behind Falcon is pretty simple. It consists of three main parts:

- Self-written proxy applications and configure-/logging facilities. These are all written in Perl.

- Third party applications like BIND, Squid, Qmail.

- Concepts/instructions/tools for hardening the OS you want to run Falcon on.

Some third party proxies maybe replaced by self-written ones in the future (it's up to you ;-)

back to TOC

# 2. System Requirements

coming soon!

# 3. The Falcon Part

## 3.1 Installation

After you downloaded the tgz-file create a directory where you want to have Falcon installed. Then unpack the the tarball into that directory:

```
$ tar xvzf falcon-x.x-x.tgz
```

Note: you need GNUtar and GNUzip to do so.

If the RPM-File is what you downloaded then simply type

```
$ rpm -i falcon-x.x-x.rpm
```

Falcon will then be installed under /usr/local/falcon. If you want it to install somewhere else you can achieve that with the option --prefix. Just say:

```
$ rpm -i --prefix /your/preferred/path falcon-x.x-x.rpm
```

Falcon expects the Perl-interpreter under /usr/bin/perl. If you have installed it somewhere else you have to link/copy it to that destination (by the way: if you havent installed Perl at all, you can get a copy from www.perl.com).

back to TOC

## 3.2 Configuration

### 3.2.1 General Configuration

The configuration of Falcon is made in a central configuration file which has an easy to understand syntax. You can find it in the etc directory under your Falcon-home (normaly /usr/local/falcon). The file is called falcon.cfg. The philosophy behind the Falcon-configuration is that you have a bunch of different proxies, that get configured by one file.
You can even start multiple instances of the same proxy-type with different config-parameters. The file is separated into sections, each section describes one instance of a proxy program. The program has to be started with the section-name as parameter and then it reads the options for that section. A section looks like this:

```
section a-nice-section:
{
 keyword1 value
 keyword2 value
 ...
}
```

The paranthesis have to be in an extra line! "a-nice-section" is the name of that section.
The keywords you can use for a section depend on the proxy-type that section is for.
There is also a section called "global-settings" that defines options for all

proxies.
If an option is also named in the local section for a proxy it overwrites the
entry in the global section.

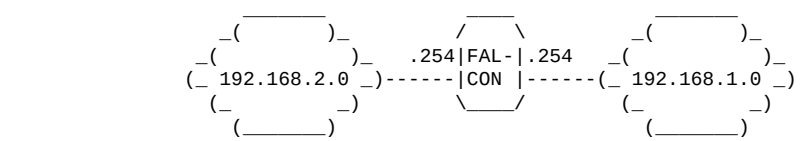
Here is a list of the proxy-types/keywords that are valid:

tcp-plug.pl:

| | |
|---|---|
| chroot-path | The path that becomes the new root for the plug during the execution. |
| run-with-uid | The effective user-id will be changed to that value after startup. |
| run-with-gid | The effective group-id will be changed to that value after startup. |
| listen-interface | The interface on which the proxy should listen for connections. This value is an IP-Address. |
| listen-port | The port on which the proxy should listen for connections. This value is numerical. |
| target-host | The host that should be connected (Hostname or IP). |
| target-port | The port that gets connected on the target-host. |

So, now lets have a look at a real world scenario:

We have two Class C subnets 192.168.1.0 and 192.168.2.0. The Gateway between these
nets is a Falcon environment. It must have at least two network interfaces, lets
say their addresses are 192.168.1.254 and 192.168.2.254.




Lets say we have a POP3-Server in 192.168.2.0 with the interface-address .3 and we
want to get our mail with a client in the 192.168.1.0 network. The connection between
the client in 192.168.1.0 and the POP3-server 192.168.2.3 should be established over
tcp-plug.pl on the Falcon-host.
The configuration-file for the Falcon would include the following section:

```
section pop3-stuff:
{
   # Change root environment
   chroot-path               /usr/local/falcon/jail/192.168.1.0

   # Don't run with an arbitrary uid/gid!
   run-with-uid              nobody
   run-with-gid              nogroup

   # Interface/Port to listen for clients
   listen-interface          192.168.1.254
   listen-port               110

   # Server to contact (IP-address or hostname)
   target-host               192.168.2.3
   target-port               110
}
```

After startup the program changes its root-directory to
/usr/local/falcon/jail/192.168.1.0 and switches its user-/groupid to nobody/nogroup.
It listens on the interface 192.168.1.254 port 110 for connections of some clients.
So the client has to be configured to connect to 192.168.2.3 port 110.

NOTE: The listen-port-number can be any port you want if you have a POP3-client that
is able to connect a POP3-server on another port than 110 (which is actually the
well-known-port for POP3).

The host that gets connected has the IP 192.168.2.3 and runs an POP3-Server on port 110.

back to TOC

#### 3.2.2 Generating a chroot-environment

##### 3.2.2.1 Setting up a chroot for Falcon-proxies

As described, every Falcon-proxy should run in a chroot-environment. This means that the program uses a different directory as root-directory than / and there's no way to change this value back during the program execution. Lets say we have configured a proxy to run with /usr/local/falcon/jail/192.168.2.4 as its root directory, then this application can not access files up from that path. You can say it thinks that /usr/local/falcon/jail/192.168.2.4 IS the root-directory. These circumstances cause that every file that is used by the application has to reside in the chroot environment. It has to be in the same path under the virtual root as it would be under /.

For example if you need the file /etc/localtime under your chroot then you have to copy/hardlink it to /usr/local/falcon/jail/192.168.2.4/etc/localtime.
The following files are needed to run Falcon-Proxies under the chroot-environment:

/etc/localtime - This is needed for the correct time in the logfile!
/etc/protocols - For applications that do some socket stuff
/etc/resolv.conf - For programs that have to do some name-resolving

back to TOC

##### 3.2.2.2 Setting up a chroot for other popular system-/network-services

One might want to run some services on a Falcon-host that do not belong to the Falcon proxy kit like named, squid and so on.
The next section describes how to run some of those popular system-/network-services in chroot-environment.

###### 3.2.2.2.1 named

The following files have to be present in the chroot-environment under that the named should run:

for Linux with glibc:

```
/lib/ld-linux.so.2
/lib/libc.so.6
/etc/named.conf
/var/lock/subsys/*
/var/named/*
/usr/sbin/named
/usr/sbin/named-xfer
/usr/sbin/ndc
```

The path of the new root is given to the named with the parameter -t at startup.
Programms running in the chroot environment need the file /etc/resolv.conf for name-resolving.

back to TOC

# 4. The 3rd Party Part

## 4.1 Squid

#### 4.1.1 Installation

0.
Download the squid package <squid-2.2.STABLE5-1.i386.rpm> (or whatever the actual name is)
from our web site and transfer it onto the allocated computer.

1.
To install squid, simply use the rpm install option with our squid package:
rpm -i <YOUR_PACKAGE_PATH>/squid-2.2.STABLE5-1.i386.rpm


2.
The user and group 'squid' will be created by our script 'create-squid-user-group.sh'
so that we can run squid as user 'squid' who belongs to the group 'squid'.
This user/group will even be added to our reduced passwd/group file in the chroot
environment.

Our point of view is that squid is a reasonable user but to resolve all doubts, the
existence of user 'squid' is not a must. You can configure squid using squid.conf to
force squid switching to a different user than squid.

Now launch the script 'create-squid-user-group.sh' while you are in the squid directory!

```
cd /usr/local/falcon/jail/usr/local/squid
./create-squid-user-group.sh
```


3.
Next you have to modify the configuration file 'squid.conf' in
/usr/local/falcon/jail/usr/local/squid/etc to meet your requirements.
Especially the following entries have to be altered!

File squid.conf in extracts:

```
# NETWORK OPTIONS
# ----------------------------------------------------------------------------
http_port 8080
icp_port 0

# OPTIONS WHICH AFFECT THE NEIGHBOR SELECTION ALGORITHM
# ----------------------------------------------------------------------------
#--
#-- cache_peer must point to a neighbour cache or to your www-server!
#--
cache_peer srv02.naw.de parent 80 7 default

# LOGFILE PATHNAMES AND CACHE DIRECTORIES
# ----------------------------------------------------------------------------
#--
#-- For the cache_dir you should use an independent and big partition.
#-- This is the entry mentioned in step 5!
#--
cache_dir /var/squid/cache 900 16 256

# ACCESS CONTROLS
# ----------------------------------------------------------------------------
#--
#-- Define here your access policy.
#--
acl ...
http_access ...

# ADMINISTRATIVE PARAMETERS
# ----------------------------------------------------------------------------
#--
#-- Modify this entry only if you want to have a different user than
#-- squid responsible for the proxy.
#--
cache_effective_user squid
cache_effective_group squid
```


4.
Make sure that you have the directory /var/squid/cache in your chroot
environment (in other words /your/safe/place/var/squid/cache) and that
it belongs to user squid.

Our preferred and recommended way is to have a separate partition for the cache_dir,
following this idea you have to create this 'mount point' and add an appropriate entry to
the /etc/fstab file. If you differ from this way, bear in mind that the /var directory
in the chroot environement is actually under /usr (/usr/local/falcon/jail/var/squid/cache)!


5.
Now it is time to create the squid swap directories by executing the
script 'create-cache-dir.pl'.

Squid utilizes its own mechanism for building the swap directories. Their
location is communicated to squid by the parameter 'cache_dir' in squid.conf.
We won't circumvent this mechanism and decided therefore to let squid build
the swap directories in his own manner.

Because this is controlled by a squid.conf entry you have to do the
modification and afterwards launch the above mentioned script.

Now launch the script 'create-cache-dir.pl' while you are in the squid directory!

```
cd /usr/local/falcon/jail/usr/local/squid
./create-cache-dir.pl
```

6.
Hooray! Squid is ready to use!

For your convenience and for security reasons there is a script ('start-squid.pl') which
will launch squid as user squid in a chroot environment.

To stop squid use 'pass-args2squid.pl -k shutdown' and if you have made some changes while
squid is running to the 'squid.conf' file a 'pass-args2squid.pl -k reconfigure' will cause
squid to re-read its configuration files.
The script 'pass-args2squid.pl' is used to pass command line parameters to the chrooted squid.

Query package (Squid):
---

If you're not sure if squid is installed you can check it with rpm's query option:
rpm -qi squid-2.2.STABLE5-1

back to TOC

#### 4.1.2 Uninstall

Uninstallation of package squid is done by:

rpm -e squid-2.2.STABLE5-1

In the final step there is a little clean up to do in some extant directories.

Verify the cache location before you unintentional erase something useful!

Get rid of the squid swap directories:
rm -r /usr/local/falcon/jail/var/squid/cache/

If you find remaining stuff in your chroot which is no longer required by anyone
you can delete it.

back to TOC

## 5. OS hardening

coming soon!

Last modified: TE/12.01.00