

# The Tircproxy manual

---

Bjarni R. Einarsson, [bre@netverjar.is](mailto:bre@netverjar.is)

Version 0.4.5, May 2000

This manual describes Tircproxy version 0.4.5. The most recent versions of Tircproxy and this manual (in both printable and browsable form) may be downloaded from *the Tircproxy home page* <<http://bre.klaki.net/programs/tircproxy/>>. This manual and the software it describes are freely available under the terms of the *GNU* <<http://www.gnu.org/>> 6 (General Public License).

# Contents

<b>1</b>	<b>Introduction</b>	<b>5</b>
1.1	Background . . . . .	5
1.2	Author & credits . . . . .	5
<b>2</b>	<b>Basic concepts</b>	<b>7</b>
2.1	IRC basics . . . . .	7
2.1.1	IRC feuds . . . . .	7
2.1.2	The DCC protocol . . . . .	8
2.2	Firewalling basics . . . . .	9
2.2.1	TCP/IP concepts . . . . .	9
2.2.2	Firewalling techniques . . . . .	10
<b>3</b>	<b>Tircproxy features</b>	<b>12</b>
3.1	Basic operation . . . . .	12
3.2	IRC proxying . . . . .	12
3.2.1	Generic proxying (not IRC) . . . . .	13
3.3	Access controls . . . . .	13
3.4	DCC support . . . . .	13
3.4.1	DCC security features . . . . .	14
3.4.2	DCC filename mangling . . . . .	14
3.5	Broadcasts . . . . .	15
3.6	Identd issues . . . . .	15
3.6.1	Changing UIDs . . . . .	15
3.6.2	Cooperation with an identd . . . . .	16
3.7	Anonymizing mode . . . . .	16

<b>4</b>	<b>Installation and configuration</b>	<b>17</b>
4.1	What do you need? . . . . .	17
4.1.1	Internet service providers . . . . .	17
4.1.2	An IRC anonymizer . . . . .	18
4.1.3	Home networks . . . . .	18
4.1.4	Mobile users . . . . .	19
4.2	Compiling Tircproxy . . . . .	19
4.3	Activating the proxy . . . . .	21
4.3.1	Standalone operation . . . . .	22
4.3.2	Inetd operation . . . . .	22
4.4	Choosing between available IP addresses . . . . .	23
4.5	Transparent or dedicated proxying . . . . .	23
4.6	Access Control . . . . .	24
4.6.1	TCP wrapper compatibility . . . . .	25
4.6.2	Passwords and quizzes . . . . .	26
4.7	DCC support . . . . .	27
4.8	Broadcasts . . . . .	28
4.9	Identd support and user configuration . . . . .	28
4.9.1	Shared memory based identd support . . . . .	28
4.9.2	Filesystem based identd support (depraciated) . . . . .	30
4.9.3	Changing user IDs . . . . .	30
4.10	Debugging and logging . . . . .	31
4.10.1	Syslog information . . . . .	31
4.10.2	Debugging verbosity . . . . .	31
4.11	Miscellanious features . . . . .	32
4.11.1	Anonymizing mode . . . . .	32
4.11.2	Sleeping between connections . . . . .	32
4.11.3	The "mIRC DCC kludge" . . . . .	32
<b>5</b>	<b>Glossary</b>	<b>33</b>
5.1	Acronyms and technical terms . . . . .	33
5.1.1	chargen . . . . .	33
5.1.2	cracker . . . . .	33
5.1.3	CTCP . . . . .	33

5.1.4	DCC, DCC SEND and DCC CHAT . . . . .	33
5.1.5	DCC RESEND (and TRESEND) . . . . .	34
5.1.6	DCC RESUME . . . . .	34
5.1.7	DCC TSEND . . . . .	34
5.1.8	default gateway . . . . .	34
5.1.9	IP . . . . .	34
5.1.10	IPF . . . . .	34
5.1.11	IP Masquerading . . . . .	34
5.1.12	IP NAT . . . . .	34
5.1.13	IRC . . . . .	35
5.1.14	proxy . . . . .	35
5.1.15	route . . . . .	35
5.1.16	TCP . . . . .	35
5.1.17	tcp wrappers . . . . .	35
5.1.18	UDP . . . . .	35

**6 GNU General Public License** **36**

# Chapter 1

## Introduction

Tircproxy is a program designed to help people make use of everything 5.1.12 (IRC) (Internet Relay Chat) has to offer, from behind a firewall. It also aims to give administrators some measure of control over which features of IRC are available, and slow the spread of *trojan horses* which are commonly distributed on IRC.

Tircproxy is suitable for home users with small local networks and a dial-up connection to the Internet, and ISPs or larger companies with a direct, but firewalled connection to the Internet.

Tircproxy is distributed under the *GNU General Public License*, and as such is free for any and all use and distribution, so long as the license isn't changed. The Tircproxy code may be used as a basis for other GPL'ed projects.

### 1.1 Background

Tircproxy was originally created to address the limitations of the Linux kernel's (version 2.0) built in 5.1.10 (IP masquerading) support for IRC. Due to the implementation of IP masquerading, users located behind the *same* firewall were unable to communicate with each other using 5.1.3 (DCC).

The implementation in the kernel also lacked support for various recent DCC enhancements, such as 5.1.4 (DCC RESEND), 5.1.5 (DCC RESUME) and 5.1.6 (DCC TSEND), all of which are supported by Tircproxy.

As the program matured, support for various platforms (other than Linux) was added, as well as more advanced features. Tircproxy is a prime example of how a seemingly simple proxy can become relatively complicated when it has deal with buggy clients, picky servers and various security issues all at the same time...

### 1.2 Author & credits

This document and the Tircproxy program were written by *Bjarni R. Einarsson* <<http://bre.klaki.net/>>, after many years of IRC addiction, a few years of Linux administration, and some encounters with IRC-hostile firewalls.

The oidentd program was written by *Odin* ([odin@ojnk.nu](mailto:odin@ojnk.nu)), but the "CDIR" patch which allows it to cooperate with Tircproxy was written by Bjarni.

---

Other contributors, in no particular order:

- *Sigurbjörn Birkir Lárusson* <mailto:sibbi@margmidlun.is> made the first port to NetBSD and added IPF support. (*home page* <<http://www.scour.org/>>)
- The original version of Tircproxy was based on *John Saunders'* <mailto:john@nlc.net.au> transparent HTTP proxy for Linux.
- *Chris Gagne* <mailto:chris@tpln.net>.
- *NJ Verenini* <mailto:nverenin@san.rr.com>.

# Chapter 2

## Basic concepts

The following discussion is included for completeness. You won't need to read it to get Tircproxy up and running - but it never hurts to know what's going on under the hood!

### 2.1 IRC basics

IRC is an acronym for "Internet Relay Chat". IRC is a standard allowing users to communicate in real time, by passing simple text messages back and forth. The IRC client and server protocols are defined in *RFC1459*.

The largest chat systems on the Internet are based on the IRC protocol. Each consists of a network of **servers**, forming a "tree". The servers keep track of who is using the chat system at any given time, keep track of **channels** and deliver messages to users or channels on request. IRC is currently text-based, but client extensions have added sound and even video capabilities.

A **client** is the program a user uses to access IRC. IRC clients are available for most, if not all, operating systems which can access the Internet.

Clients connect to servers using a single 5.1.15 (TCP) connection, initiated by the client.

#### 2.1.1 IRC feuds

Due to the highly interactive nature of IRC, virtually all aspects of human nature manifest themselves online - people make friends and enemies, have fun and fight amongst themselves.

It is unfortunately rather common for arguments online to escalate until one or both parties (if they have the know-how) attempt to "attack" each other. Some people attack others without any provocation at all. Various methods are used:

1. **Flooding** involves sending large amounts of random data to the victim, in order to disrupt his IRC session or even saturate his network connection.
2. **Channel takeovers** occur when an unfriendly person takes control of the victim's channel. When this happens the channel usually becomes unusable.

3. **Nick collisions** occur when two users try to use the same nickname simultaneously - the IRC protocol doesn't allow this, so both users are usually disconnected.
4. **Clone bots** are automated IRC clients, usually used in large numbers to help implement attacks 1, 2 or 3.
5. **Nukes** are TCP/IP attacks used to disrupt the IRC network itself, creating an opportunity to launch 2 or 3.
6. **Bugs** in various IRC clients are frequently exploited.
7. **Trojans** can be sent via DCC, allowing the attacker to gain an essentially arbitrary amount of control over the victim's computer - if the victim can be tricked into accepting and activating the trojan. **Bugs** occasionally make "social engineering" unnecessary.
8. The IP address of the client is made available to the attacker by a simple server query, allowing 5.1.1 (crackers) to make arbitrary attacks, not limited by the IRC network.
9. etc. . .

Obviously, only attacks 2 and 3 can be considered entirely harmless, all others constitute security risks (6, 7, 8) or denial of service attacks for the client (1, 6, 7, 8) or the IRC network itself (1, 4, 5).

For these reasons, IRC is generally considered a **security risk** and is blocked at most firewalls. This is very unfortunate, as IRC can be a very useful, low bandwidth communication tool and has successfully been used for online meetings, lectures, parties, tech support and more.

Tircproxy attempts to address some of the above problems by giving the system administrator finer-grained control over how much access his users have to various IRC features and by protecting the users' privacy.

### 2.1.2 The DCC protocol

IRC clients use a special protocol, 5.1.2 (CTCP), which is implemented on top of IRC's basic messaging system to transfer "technical" information from client to client. **DCC** is a subset of CTCP, which allows clients to establish direct 5.1.15 (TCP) connections, bypassing the IRC network itself. This is the only aspect of IRC communications requiring special attention from Tircproxy.

DCC is mostly used for private discussions (DCC CHAT) and exchanging files (DCC SEND). When user **A** wants to send user **B** a file, the clients perform the following steps:

1. **A**'s client allocates a 2.2.1 (TCP port) and begins to listen for connections to this port.
2. Next a CTCP (DCC SEND) message is sent to **B**, which contains **A**'s 2.2.1 (IP address), the port number from step 1 and information about the file being offered.
3. **B**'s client tells the user that the file is available. If the user accepts the file (with some clients this happens automatically), the client attempts to make a TCP connection to the IP address and port number contained in the CTCP message.
4. If all goes well, the connection succeeds and the file is sent.

Note that no verification of *who* actually connects to the listening data port is usually done, so "hijacking" a DCC connection is relatively trivial. It is also trivial to forge a DCC offer, tricking **B** into connecting to an arbitrary server/port on the internet.

Other DCC variants, such as 5.1.3 (DCC CHAT), 5.1.6 (DCC TSEND), 5.1.4 (DCC RESEND) and more are implemented in basically the same way. The only exception is the 5.1.5 (DCC RESUME) protocol, which incidentally also violates the IRC messaging protocol.

Tircproxy understands the messages involved in the above process, and rewrites them, blocks them or ignores them according to the policies defined by the administrator. It also implements a few features 3.4.1 (designed to decrease the risks) discussed above.

## 2.2 Firewalling basics

Firewalls are used to selectively limit what can pass through a network connection and log information about what the connection is used for, with the goal of enforcing certain security policies.

The basic idea, is to pass all traffic through the firewall, thus giving the administrator a single point where the desired policies can be enforced. This single point of control allows the administrator to "hide" characteristics of a private network and protect it from "attacks" from the outside (usually the Internet).

Also, since the 2.2.1 (IP addresses) used on some private networks cannot be used for direct Internet access (see *RFC1597*). Firewalling techniques (such as 5.1.13 (proxies) and 5.1.11 (IP NAT)) can be used to provide various degrees of access in spite of the limitations imposed by such "hidden" networks.

Of course, mere machines can only enforce a small part of most security policies - clever humans can invariably avoid restrictions imposed by a firewall, no matter how advanced it is.

### 2.2.1 TCP/IP concepts

The following basic concepts about IP (the Internet Protocol) must be understood, for a discussion about firewalls to make any sense.

**IP address** are 32-bit numbers which identify machines on a TCP/IP network. IP addresses are usually written as four positive integers, separated by dots - the most significant first, and the least significant last.

**Example:** 10.123.44.98

**Port numbers** are 16-bit numbers which identify "ports" on a machine. Port numbers are used to differentiate between different data sessions and services which are active on the same machine (thus sharing the same IP address). Some common services, such as SMTP and DNS have standardized port numbers (SMTP=25, DNS=53), other services such as HTTP proxies can use any port number. Knowing what port a given service runs on is often critical for accessing it through a firewall. Port numbers are often appended to IP addresses, using a ':' as a delimiter.

**Example:** 10.123.44.98:53  
(the DNS service on 10.123.44.98)

Data on a TCP/IP network is transmitted in **IP packets**. Both **TCP** and **UDP** packets contain data, status bits and two pairs of IP addresses and ports - one pair indicating where the packet comes from and

the other where it is going. We are only interested in TCP packets, because both DCC and IRC use TCP connections.

A typical telnet session from 10.1.2.3 to 10.3.2.1 consists of a stream of **TCP** packets, where each one heading from the client to the server has the following information in it's header:

```
From 10.1.2.3:12423, To 10.3.2.1:23, Data
```

And the reply packets have a header like this:

```
From 10.3.2.1:23, To 10.1.2.3:12423, Data
```

## 2.2.2 Firewalling techniques

There are three methods commonly used in firewalls to enforce the local security policy. The three methods are commonly used together, each complementing the others' weaknesses.

*Note:* The terms used in this section may be at odds with those used for the same concepts elsewhere. The goal of this section is to explain the underlying principles, not to define a vocabulary for communicating with networking professionals.

### Packet filters

Packet filters operate on the IP level, scanning the headers of each IP packet crossing the firewall and comparing its characteristics to a fixed set of rules. These rules determine whether the packet is allowed to pass unhindered or not.

Characteristics recognized by packet filters are the source and destination IP addresses, the source and destination port numbers, various status bits in the header, and the direction the packet is travelling across the firewall.

Packet filters don't know anything about protocols above the TCP/IP layer - they are fast and simple, but not very flexible.

### Application layer proxies

Application layer proxies are applications running on the firewall, which users on one or both sides of the firewall can communicate with.

The proxies forward the users' requests to the actual servers which can give a response, possibly imposing rules on what sort of traffic is acceptable. From the viewpoint of the "actual servers", it appears as if the firewall is making the requests - not the client. This can provide a modicum of privacy and protection for the user behind the firewall.

Application layer proxies are in general the most flexible type of firewalling software, but they frequently require added configuration or skills from the user.

A common example of this is an HTTP proxy, which allows users to request web pages from anywhere but may refuse some requests or rewrite pages based on rules defined by the administrator. To make use of an HTTP proxy, you must enable and configure proxy support in your browser.

Tircproxy can function as an application layer proxy, and can hide the user's identity.

## Network layer proxies

Network layer proxies are a cross between application layer proxies and packet filters. Like packet filters, they scan the headers of the IP packets crossing the firewall - but are able to respond in more ways than packet filters which generally only "accept" or "reject".

5.1.10 (IP masquerading) and 5.1.11 (IP NAT) are simple network layer proxies, which merely replace the "hidden" network's IP addresses with an address from a list of "visible" ones, on a connection-by-connection basis. This way, machines on the hidden network can "borrow" visible IP addresses and/or source ports from the firewall itself, which can be used to communicate with the untrusted network.

To make these "dynamic tunnels" as narrow as possible, a list of active communication channels (5.1.15 (TCP), 5.1.17 (UDP), etc ..) is maintained, and only packets (from the "outside") which exactly match an active connection are "proxied" back into the hidden network. Only internal, trusted machines can open such a tunnel through the firewall.

In addition to this basic "translation" of network addresses and ports, some implementations contain built-in support for common protocols (such as FTP or IRC/DCC) which are dependant on external, untrusted hosts being able to 2.1.2 (initiate connections to the client). In general though, such protocols *won't work* through most firewalls based on these techniques.

Another type of network layer proxy is based on cooperation between specialized application proxies, and the operating system's network code. In addition to re-writing the IP addresses on the network packets, some packets may be passed for further processing to an application proxy on the firewall machine. This proxy can monitor and filter the flow of data between the client and the external server in a much more complicated (and therefore error-prone) manner than is usually allowed within an operating system's low level network code. These proxies are generally called **transparent proxies**, because they [can] operate in a manner completely invisible to the user.

Tireproxy can function as a transparent proxy, on some platforms.

## Chapter 3

# Tircproxy features

This chapter describes what Tircproxy can [and can't] do. For installation and configuration instructions, see the 4 (next chapter).

Tircproxy's basic goal is to allow users to connect to IRC through a firewall of some sort and enjoy everything IRC has to offer - including private chats (DCC CHAT) and the exchange of files (DCC SEND). For this reason, most features are enabled by default.

Tircproxy's many features are mostly designed with a large-scale environment in mind (such as an ISP), but Tircproxy can also be very useful for home users with a dialup connection to the 'net, or mobile users who wish to IRC from the same host all the time. Typical configurations for each of these users are covered in the 4 (next chapter).

### 3.1 Basic operation

On Unix systems, services which are seldom used can be automatically started when needed by the **inetd** super-daemon. This conserves resources, but incurs a minor delay when the connection is initiated. This also gives administrators a central point for implementing IP-based access controls (usually via. tcp wrappers). Tircproxy can be started on demand by inetd.

Alternately, if Tircproxy is being used alot or if you wish to take advantage of certain features unavailable when running from inetd, Tircproxy can run in stand-alone mode (as a daemon).

### 3.2 IRC proxying

IRC proxying can be dedicated, transparent or both.

When running as a *dedicated proxy*, all connections will be automatically forwarded to an IRC server of the administrators choice. Users won't be able to connect to any other servers. Unless used in combination with transparent proxying (*both*), the users will have to configure their IRC client to connect to the firewall itself.

Alternately, if the host running Tircproxy is a gateway on the 5.1.14 (route) from the users to the Internet, and it is running Linux, or another operating system with the 5.1.9 (IPF) package installed, Tircproxy can

run in *transparent mode*. This allows the users to connect to any IRC server they like, without any "unusual" configuration of their clients.

### 3.2.1 Generic proxying (not IRC)

As a fourth mode of operation, Tircproxy can be run as a generic TCP proxy with no special IRC-related behavior (the `-N` flag). This is useful if you want to take advantage of Tircproxy's 3.6 (ident features) for other protocols, such as HTTP or SMTP.

## 3.3 Access controls

Access to the proxy may be restricted in various ways. In addition to the "standard" methods (tcp wrappers, kernel firewalling rules) various restrictions are built into Tircproxy itself.

**Quiz mode** requires a password from the user, before any data can be sent from the client to the server. The password can be either the user's local Unix password, or one read from a user-defined file. The reason for calling this "quiz mode" is that the password mechanism can be configured to supply the user with a *random* question from a text file. This is meant to hinder [clone] bots from using the proxy, while allowing unlimited access to entities smart enough to answer the questions (real people).

The files `/etc/hosts.allow` and `/etc/hosts.deny` are used on many Unix systems to control access to various services. Tircproxy can also make use of these files, to control both access to the proxy itself and provide finer grained control over which internal and which external hosts may initiate DCC connections, and to dynamically add filenames to the mangling list.

If the administrator so chooses, features such as DCC support may be disabled either at run-time or compile time.

## 3.4 DCC support

Once an IRC connection has been established, Tircproxy monitors the traffic sent between the client and the server, in order to implement support for 5.1.3 (DCC) (and some other things).

As 2.1.2 (described) in the previous chapter, DCC relies on direct connections between clients - which would basically mean that the whole of the Internet would need unhindered access to any of the firewalled machines, for DCC to work. This is obviously unacceptable in many situations, and impossible in others.

Therefore, Tircproxy monitors the IRC connection, and when it sees a DCC request it takes measures to either block the request or facilitate it, depending on the configuration.

Tircproxy, as of version 0.4.0, recognizes and supports the following DCC requests:

- DCC CHAT
- DCC SEND
- DCC TSEND
- DCC RESEND

- DCC TRESPASS
- DCC RESUME

If given an unknown DCC request, Tircproxy attempts to handle it in the same way as it handles DCC CHAT.

### 3.4.1 DCC security features

Tircproxy's DCC proxying algorithm implements source port randomization (even on OSes which don't do this by default), providing a modicum of protection against DCC session hijacking and related attacks.

In addition the proxy will block bogus DCC requests which try to trick a user into connecting to a potentially dangerous port, such as the 5.1 (character generator), and will refuse DCC requests unless the original client is one of the endpoints. This discrimination significantly decreases the chance of Tircproxy being abused to 2.1.1 (flood) the user or attack services running on the internal network.

### 3.4.2 DCC filename mangling

When a user is offered (via DCC) a file, or offers another user a file, Tircproxy can check the file's name against a list. If the filename is on the local blacklist, Tircproxy will block the transmission. Alternately, the filename can be rewritten and the transmission allowed, forcing the recipient to manually rename the file to make use of it.

This feature was added to slow the spread of the most common trojan horses, the most famous of which is called `script.ini`. In the case of `script.ini`, the filename is critical since it has a special meaning to the popular Windows IRC client, mIRC.

Filenames currently on the default blacklist:

- `script.ini`
- `dmsetup.exe`
- `dmsetup2.exe`
- `winhelper.exe`
- `mschv32.exe`
- `mircc.ini` (renamed to `'mircc.in-'`).

The blacklist can be enlarged on-the-fly by the administrator, using 4.6.1 (the TCP wrapper support).

Please note that this defense will *not* block the transmission of trojan horses such as Back Orifice, Netbus or anything which is commonly distributed under unknown filenames.

(As a replacement, a method to take advantage of a full-fledged virus scanner would come in handy here - but due to a lack of time and a complete lack of requests for such a feature nothing like this is planned.)

## 3.5 Broadcasts

One of the things commonly missed by administrators of dial-up systems, who are used to a multi-user Unix environment is the ability to send the user a 'message of the day' when he logs on.

Tircproxy addresses this by adding support for arbitrary broadcasts from the administrator to all connected clients. A 'motd' style broadcast can easily be defined (just create the file `/etc/motd.irc`). Arbitrary messages can be sent at any time by creating another file (`/tmp/ircbroadcast`) and sending the proxy the 'hangup' signal.

## 3.6 Identd issues

On IRC, a common requirement for connection (or access to certain channels) is to have a valid username. Information about a user's identification is made visible to other users as a `username@hostname` string, such as `bre@ppp-248-123.isp.some.where.com`.

The IRC server acquires the username by querying the client machine's ident daemon, if one is running. As Tircproxy will usually be running under some system account on a multi-user Unix machine, this poses a problem for the users of the proxy - they will all be identified as `proxyuser@tircproxyhost`.

Since this identification is frequently used to grant privileges to users (or to punish them) sharing a common ID will cause problems.

This is of course not a problem if Tircproxy is run by a single user for personal use only (e.g. as a 4.1.4 (bouncer)).

For others, there are three possible solutions:

1. Disable the ident daemon on the proxy host.
2. Make Tircproxy change UIDs depending on who owns the connection.
3. Have Tircproxy communicate with the running identd.

In cases 2 and 3, Tircproxy must generate its own psuedo-usernames based on e.g. the IP address of the client, or read information about what user is using a given IP address from configuration files. Tircproxy in it's current incarnation can only map one user-name to each IP address at a time, which may cause problems for users sharing a single multi-user machine which is behind a firewall.

### 3.6.1 Changing UIDs

If all users of the proxy have a local username on the proxy host, then Tircproxy can simply change personalities before connecting to the IRC server.

For this to work, Tircproxy will need information about which client IP addresses belong to which users.

This simple solution has one serious drawback though - only processes running as 'root' can change their UIDs. This implies that any mistakes I may have made writing Tircproxy, could be abused to remotely gain administrator capabilities on the proxy host!

### 3.6.2 Cooperation with an identd

To avoid running the proxy as root, and to increase the control Tircproxy has over the replies provided by the system's ident daemon, Tircproxy can communicate directly with modified versions of some ident daemons. Check *the Tircproxy home page* <<http://bre.klaki.net/programs/tircproxy/>> for a list of available patches.

The first to be patched was the *oidentd* daemon, which was chosen for use with Tircproxy because it was relatively full-featured, and had recently been audited by members of the Linux Security Audit project.

Again, for this to work optimally, Tircproxy will need information about which client IP addresses belong to which users. Alternately, if no such information is available, Tircproxy can generate 'fake' IDs based on the client's IP address.

## 3.7 Anonymizing mode

Anonymizing mode makes the proxy hide as much user information as possible. The following things are done:

- CTCP replies from the client are intercepted and all answers to FINGER, CLIENTINFO, VERSION and USERINFO requests are rewritten.
- The IRCNAME visible in *whois* queries is replaced with a fixed, uninformative name.
- Fake usernames are generated based on the client's IP address (requires identd cooperation).
- All NOTICEs from the client which contain his IP address are blocked (mIRC sends such messages when DCCing).

Note that the default behavior (in non-anon mode) for the last item is to pass the NOTICE on, but replace the client's IP address with the proxy's visible address. This can reveal information about what IRC software the user is using (mIRC in particular), which is why the NOTICE is blocked entirely in anonymizing mode.

The generated username is based on the client's IP address, to allow the operators of the IRC server and channels to selectively ban abusive users - hopefully making it less likely that they'll resort to banning the proxy host completely.

# Chapter 4

## Installation and configuration

This chapter describes how to install and configure Tircproxy.

### 4.1 What do you need?

Depending on what you are using Tircproxy for, some of the proxies features will be useful to you, and some won't. Typical scenarios are described in the next four sections, along with references to the relevant sections and command line examples.

#### 4.1.1 Internet service providers

An ISP using Tircproxy to give firewalled users access to IRC, will probably use most of the features available:

- 4.3.1 (Standalone operation), for speed and reliability.
- 4.5 (Transparency) to make life easier for the users.
- 4.6.1 (TCP wrapper support) for access control and updating of the mangling blacklist.
- 4.9 (Cooperation with an identd), for correct identification without adding to the list of root servers.
- 4.7 (Filename mangling), to slow the spread of trojans.
- 4.8 (The IRC MOTD and broadcast features).
- 4.10 (Nickname logging), for tracking abuse.
- 4.11.2 (Delayed connections), to play nice with the servers.

Sample command line:

```
tircproxy -s <port> -b <internal address>
```

(The above is a real-life example, from an ISP with several thousand users.)

### 4.1.2 An IRC anonymizer

It has been a tradition online to offer people anonymous access to various services, such as email and usenet. Tircproxy can be used to grant anonymous access to IRC, by using the following features:

- 4.3.1 (Standalone operation), if alot of use is expected.
- 4.5 (Dedicated mode), to redirect users to a server allowing anonymous use.
- 4.6.1 (TCP wrapper support), for blocking abusive users.
- 4.9 (Cooperation with an identd), for generating 'fake' userids.
- 4.7 (Filename mangling).
- 4.8 (The IRC MOTD and broadcast feature).
- 4.6.2 (Random quiz mode), to block bots but allow 'humans' to use the proxy unhindered.
- 4.11.1 (Anonymizing mode), to hide as much user info as possible.
- 4.11.2 (Delayed connections), to play nice with the server.

Sample command line:

```
tircproxy -a -r <proxy uid> -s <port> -b <internal address> -q <quiz file> <irc server> <port>
```

### 4.1.3 Home networks

Home users with small networks (many machines) and a (single) dial-up connection to the Internet may want to use Tircproxy to give the indirectly connected machines access to IRC. In such situations Tircproxy would be installed on the machine with direct access to the net, which is commonly a Linux box with IP masquerading or other 2.2 (firewalling) facilities enabled, and the other machines would use it box as their "default router" to the Internet.

A minimal configuration might use the following features:

- 4.5 (Transparency) to make getting online as simple as possible.
- 4.9 (Setuid operation), to make ident replies consistant.
- 4.7 (Filename mangling) to block trojans.

Sample command line:

```
tircproxy -HID -s <port> -b <internal address>
```

Alternately, dedicated (non-transparent) mode could be used on operating systems which don't implement 5.1.10 (IP masquerading) or 5.1.9 (IPF):

```
tircproxy -HID -s <port> -b <internal address> <irc server> <port>
```

#### 4.1.4 Mobile users

Mobile users, who connect to the internet from many different locations (work, home, school, friends' homes, ...) often want a persistent IRC identity, which isn't dependant on where they are actually connecting from. This can be achieved in many ways, one of which involves using Tircproxy as a "bouncer".

The following features would be used by a bouncer:

- Standalone operation (normal users usually cannot modify inetd's configuration).
- Dedicated mode, to connect the user to his favorite server.
- Filename mangling, for security.
- Logging to stderr instead of syslog, since the user probably cannot access the system logs.
- A very simple quiz file, to put a password on the proxy.

Sample command line:

```
tircproxy -HIDL -s <port> -q <quiz file> <irc server> <port>
```

There are other programs which specialize as bouncers, which have different features from those offered by Tircproxy. Two of them are *bnc* <<http://bnc.dragondata.com/>> and *muh* <<http://mind.riot.org/muh/>>. People who want to have the features of two or more of these programs at once can quite easily daisy-chain them together, with no problems other than a slight increase in latency. The order may matter a bit though, as mentioned 4.11.3 (in the mIRC DCC kludge section).

## 4.2 Compiling Tircproxy

(If you have a binary distribution of Tircproxy you can 4.3 (skip) this section. Installation of .rpm and .src.rpm packages is not covered by this manual, consult the rpm documentation instead. You should be aware though, that binary distributions may not have all features enabled - the only way to be sure everything is enabled is to compile it yourself!)

Compiling Tircproxy should be relatively straightforward on most systems. You should follow these steps:

1. Tircproxy has few prerequisites, it should detect the features offered by your system automatically and configure itself accordingly. But if you want to be sure a feature will be available, you should check this list, and install the things you need before continuing:
  - Wietse Venema's tcp wrapper library, **libwrap**, should be installed if you want to control access to the proxy with `/etc/hosts.allow` and `/etc/hosts.deny`.
  - The *UDB* <<http://bre.klaki.net/programs/udb/>> package should be installed if you want Tircproxy to use shared memory to 4.9 (cooperate with oidentd) (recommended).
  - On Linux, you will need the header files from the kernel sources.
2. Download the most recent "tircproxy-X.Y.Z.tar.gz" file from <http://bre.klaki.net/programs/tircproxy/>

3. Extract the tar file, and enter the created source directory:

```
gzip -dc tircprozy-X.Y.Z.tar.gz | tar xvf -
cd tircproxy-X.Y
```

4. Edit the file `tircproxy.h` if you wish to *disable* specific features, customize the compiled-in messages, the DCC blacklist or modify file locations.
5. Execute the following commands to build the program:

```
./configure
make
```

The 'make' command will build the proxy, and print out warnings to tell you what features are being compiled in. Pay attention to these messages - some features are not available on all systems, and reading this might save you some head-scratching later on.

6. If all goes well, and you are the system administrator, give the following command:

```
su -c "make install"
```

If you aren't the system administrator, simply copy the executable file `tircproxy` to some convenient location.

7. Finally, to clean up, give the command:

```
make clean
```

On a Linux system, the build sequence should look something like this:

```
[bre@discordia tircproxy-0.4]$ ./configure
creating cache ./config.cache
checking host system type... i486-pc-linux-gnu
checking target system type... i486-pc-linux-gnu
checking build system type... i486-pc-linux-gnu
checking for gcc... gcc
checking whether the C compiler (gcc ) works... yes
checking whether the C compiler (gcc ) is a cross-compiler... no
checking whether we are using GNU C... yes
checking whether gcc accepts -g... yes
checking for a BSD compatible install... /usr/bin/install -c
checking for main in -lwrap... yes
checking for main in -lnsl... yes
checking for main in -lsocket... no
checking for main in -lcrypt... yes
checking for strip... /usr/bin/strip
checking for OS quirks... goody - it's Linux!
checking how to run the C preprocessor... gcc -E
checking for ANSI C header files... yes
checking for sys/wait.h that is POSIX.1 compatible... yes
checking for fcntl.h... yes
checking for sys/ioctl.h... yes
```

```
checking for sys/time.h... yes
checking for crypt.h... yes
checking for syslog.h... yes
checking for unistd.h... yes
checking for netinet/ip_nat.h... no
checking for working const... yes
checking for uid_t in sys/types.h... yes
checking whether gcc needs -traditional... no
checking whether setpgrp takes no argument... yes
checking return type of signal handlers... void
checking for select... yes
checking for socket... yes
checking for strerror... yes
checking for strstr... yes
checking for vsnprintf... yes
checking for vprintf... yes
updating cache ./config.cache
creating ./config.status
creating Makefile
creating config.h
```

```
[bre@discordia tircproxy-0.4]$ make
gcc -c -I. -I. -g -O2 tircproxy.c
tircproxy.c:69: warning: #warning MIRC DCC kludge active
tircproxy.c:72: warning: #warning Nickname logging active
tircproxy.c:96: warning: #warning Configuration via /etc/hosts.* available
tircproxy.c:119: warning: #warning IPF support not available
tircproxy.c:133: warning: #warning Linux transparent proxying available
tircproxy.c:139: warning: #warning Quiz mode available
gcc tircproxy.o -o tircproxy -lcrypt -lnsl -lwrap
[bre@discordia tircproxy-0.4]$ su -c "make install"
Password:
/usr/bin/install -c -o bin -g bin -m 555 tircproxy /usr/local/sbin/tircproxy
[bre@discordia tircproxy-0.4]$ make clean
rm -f tircproxy.o tircproxy *~ *.bak
[bre@discordia tircproxy-0.4]$
```

### 4.3 Activating the proxy

To see if the installation succeeded, and to view a summary of the arguments and options understood by the proxy, give the command:

```
tircproxy -h
```

If your terminal window is too small to display all the output at once, try this:

```
tircproxy -h 2>&1 | more
```

The listed options and other configuration issues will be discussed in the following sections, but performing a quick test right away is very simple - the most important thing is to disable all the features you haven't configured yet. Something like this should work:

```
tircproxy -d9 -s 7666 -MILRH irc.undernet.org 6667
```

Here, the capitalized command line options are disabling most of the features of the proxy. Test the proxy by pointing your IRC client at port 7666 of the host running Tircproxy. DCC SEND and DCC CHAT should work, and you should see the proxy output debugging messages as it detects and handles the requests. Cancel the program by hitting CTRL-C.

### 4.3.1 Standalone operation

The 4.3 (above example) is typical for standalone operation of the proxy: just specify the options you wish to use on the command line, or within a startup script.

When in standalone mode, the **-s port** option is mandatory - it tells the proxy to run as a server on the named port. Any port number should do, but in this document most examples use port 7666. Without this option the proxy assumes it's being run from 4.3.2 (inetd)

Note that without the **-d9** flag (debug level 9), the proxy will put itself in the background.

### 4.3.2 Inetd operation

Configuring Tircproxy to run from inetd is slightly more complicated, but can be very convenient when offering dedicated access to multiple servers.

Assuming Tircproxy is installed as `/usr/local/sbin/tircproxy`, add lines like the following to `/etc/inetd.conf`, one for each server you wish to grant access to.

```
7666  stream  tcp    nowait  root
      /usr/local/sbin/tircproxy tircproxy <options> irc.undernet.org 6667
7667  stream  tcp    nowait  root
      /usr/local/sbin/tircproxy tircproxy <options> irc.som.ewh.ere.com 6667
```

**Notes:** The above example should be *two* lines, not four. The lines were split to make them more legible. The syntax described here applies to the inetd distributed with the Linux Netket version 0.9. The correct syntax for your operating system may be somewhat different.

The numbers in the first column are the port numbers used by each proxy. Thus connecting to proxyhost:7666 will give access to irc.undernet.org, but connecting to proxyhost:7667 will connect you to irc.som.ewh.ere.com. These numbers are completely arbitrary - choose any numbers you like.

Replace `<options>` with the options you are really going to use with the proxy (read on for details). Note that the **-L**, **-s** are not applicable when running the proxy from inetd.

Experienced admins will note that I do not recommend wrapping Tircproxy inside 5.1.16 (TCP wrappers). The reason for this is that Tircproxy has 4.6.1 (built-in support) for the TCP wrapper configuration files (`/etc/hosts.allow` and `/etc/hosts.deny`) on many systems, including Linux. The 4.2 (build sequence) tells

you if your system has this feature (alternately, if you run Tircproxy with no arguments it will print a list of options and flags - if **-H** is on the list then your version of Tircproxy includes TCP wrapper compatibility).

Transparent mode does not seem to work when the proxy runs from inetd. Solutions are welcome.

## 4.4 Choosing between available IP addresses

Some machines have more than one IP address with direct access to the internet and/or the internal network. By using the **-b**, **-i** and **-o** options, you may choose which of them are used and when.

The **-b** option tells the proxy on which IP address to listen for incoming IRC connections. By using this option you can make sure that Tircproxy is only listening to your internal address, which could suffice to keep outsiders from attempting to use the proxy. This flag is incompatible with 4.3.2 (inetd operation).

The **-i** option tells the proxy which IP address to use for DCC communication with it's clients. This value can be autodetected, except on Linux in transparent mode (because of how transparency is supported by the Linux kernel).

The **-o** option tells the proxy which IP address to use to connect to a remote server. Thus if the host running Tircproxy has two external addresses, e.g. 1.2.3.4 (example.com) and 1.2.3.5 (example.org), you can choose which is used for communication with the Internet (both IRC servers and external DCC). This lets you choose whether the proxy users appear to come from example.com or example.org. Without this option Tircproxy will let the operating system decide which address to use.

Using the above addresses and the added assumption that the machine's internal addresses are 10.11.12.13 and 10.11.13.14, these flags could all be tested at once by running Tircproxy like this:

```
tircproxy -d9 -s 7666 -i 10.11.12.14 -b 10.11.12.13
-o example.com -MILRH irc.undernet.org 6667
```

Trying to connect to port 7666 on most of the machines addresses should fail, only connecting to 10.11.12.13:7666 will open a connection to IRC. That connection would originating from example.com. The interface 10.11.12.14 would be used for internal DCC-related communication.

## 4.5 Transparent or dedicated proxying

The above examples all assumed you will be running the proxy in dedicated mode, where the server is explicitly named on the command line, and the IRC client must connect directly to the host running the proxy. If dedicated access to multiple hosts is to be granted, the admin must create several configurations of Tircproxy, and the user needs to know which port on the proxy host corospond to the server he wishes to connect to.

This can quickly get quite out of hand.

Transparent mode solves this problem. On Linux 2.0 and up, and on other systems which support the 5.1.9 (IPF) package (FreeBSD, NetBSD, OpenBSD, Solaris, ...) the proxy can be configured to intercept direct connection attempts which would otherwise fail or offer degraded service (e.g. broken DCC, see 2.1.2 (chapter 2) for details).

If you want to run Tircproxy in transparent mode, you must install it on the border of your firewall - it must have direct access both to the clients and the Internet. In addition, the host Tircproxy is installed on must be the clients' 5.1.7 (default gateway). (These conditions are usually all met on home networks, when the machine running Tircproxy is the one with the modem.)

Running Tircproxy in transparent mode means you don't specify a server and port to connect to when you run the proxy - the proxy automatically connects to the server which the user had selected. Under Linux, additional information will be needed for this to work - specifically the internal address of the proxy, so it can rewrite DCC requests in a fashion that guarantees that the clients can accept what's offered. This is done using the `-i` flag. Example:

```
tircproxy -d9 -s 7666 -MILHR -i 10.10.10.254
```

This is not enough though - the operating system's firewalling subsystem must also be configured to redirect connections to the proxy. How this is done varies slightly between operating systems. Examples:

#### Linux 2.0:

```
ipfwadm -I -i accept -P tcp -S 10.10.10.0/24 -D 0.0.0.0/0 6667 -r 7666
```

#### Linux 2.2:

```
ipchains -A input -j REDIRECT 7666 -p tcp -s 10.10.10.0/24 -d 0.0.0.0/0 6667
```

#### IPF:

```
rdr ep0 10.10.10.0/24 port 6667 -> 127.0.0.1 port 7666 tcp
```

#### Notes:

- It is possible in each case to add multiple rules to intercept more common IRC ports, such as ports 6666, 6668, etc.
- These examples assume your local network is number 10.10.10.0, with a 24-bit netmask (255.255.255.0), and that your Tircproxy is running on port 7666, on a machine with the internal address 10.10.10.254.
- The IPF rule was tailored for a NetBSD box, with it's local (internal) ethernet interface named `ep0`.
- On most platforms, the order of the rules given to the firewalling code matters - if the Tircproxy redirection rule is appended to a ruleset *after* some other more generic rule, the Tircproxy rule will be useless. Consult your operating system's documentation for more details.

## 4.6 Access Control

Access to the proxy and it's features can be restricted in many different ways:

1. TCP wrapper files, `/etc/hosts.allow` and `/etc/hosts.deny`,
2. passwords and quizzes,

3. using the right interfaces,
4. kernel firewalling rules.

The third method is discussed in 4.4 (the section about choosing between IP addresses).

The fourth requires no specific configuration of Tircproxy, and is also highly platform dependant. It is therefore beyond the scope of this document, refer to your operating system's documentation for more information.

Using many (or all) of these methods at the same time may be a good idea.

### 4.6.1 TCP wrapper compatibility

This feature is *enabled* by default. Disable it with the **-H** flag.

Many systems have adopted the use of Wietse Venema's TCP wrapper system, to restrict access to daemons invoked through inetd. Originally this was implemented using a standalone application, which was read it's configuration from files named `/etc/hosts.allow` and `/etc/hosts.deny`. Tircproxy can make direct use of these files, without needing the wrapper program, whether it is running as a standalone daemon or invoked from inetd.

From the `hosts_access(8)` man page (tcp wrappers 7.6):

The access control software consults two files. The search stops at the first match:

1. Access will be granted when a (daemon,client) pair matches an entry in the `/etc/hosts.allow` file.
2. Otherwise, access will be denied when a (daemon,client) pair matches an entry in the `/etc/hosts.deny` file.
3. Otherwise, access will be granted.

A non-existing access control file is treated as if it were an empty file. Thus, access control can be turned off by providing no access control files.

Tircproxy checks for the following daemon tags:

- "**tircproxy**" is checked whenever a client initiates a connection to IRC, through the proxy. When running in 3.2.1 (generic mode) (the **-N** flag), the tag "**proxy**" is used instead.
- "**tircproxy\_dcc\_files**" is matched against the filename, when a file is offered for DCC send. This allows the administrator to add trojan filenames to the Tircproxy blacklist without recompiling the program.
- "**tircproxy\_dcc\_in**" is compared to the internal user's IP and username (or hostname), whenever a DCC connection is requested.
- "**tircproxy\_dcc\_out**" is compared to the external user's IP address (or hostname), whenever a DCC connection is requested by an external user. This does not effect connections initiated by the internal user.

**Note:** The user name provided by the 3.6 (identd) support code is used instead of sending an authentication request to the client.

It is generally a good idea to disable this feature when you are first testing the proxy, since many sites have a catch-all "ALL: ALL" rule in the `/etc/hosts.deny` file, which will cause the proxy to reject all connections.

### An example:

Assuming you wanted to allow only the 10.11.12.x network to connect to your proxy, but didn't want the user on 10.11.12.13 to be able to do any DCC stuff, you might use the following configuration:

```
/etc/hosts.allow:
```

```
tircproxy_dcc_out:  ALL
```

```
/etc/hosts.deny:
```

```
tircproxy:          ALL except 10.11.12.0/255.255.255.0
tircproxy_dcc_in:   10.11.12.13
tircproxy_dcc_in:   ALL except 10.11.12.0/255.255.255.0
tircproxy_dcc_files: bo.exe passwd
```

This also adds the file names "bo.exe" and "passwd" to the DCC SEND blacklist.

## 4.6.2 Passwords and quizzes

These features are *disabled* by default. Activate them with the **-q** or **-p** options.

In many cases it is more useful to limit access to the proxy by requiring a password than by IP address or hostname. Tircproxy supports two methods of password authentication, one based on the local user list (usually `/etc/passwd`) and the other based on a quiz file.

In both cases, events will happen like this:

1. The user will connect to the proxy, and the proxy will connect to the IRC server. It will choose a semi-random nickname for the user.
2. The proxy will send any messages originating from the server to the client immediately, but will postpone all messages from the client until authentication has succeeded. If the client sends over 25kB of data the connection will be dropped.
3. The proxy will send the user a message requesting either a username and password, or an answer to a question from the "quiz file".
4. The user will respond by either sending the proxy a message or by embedding the answer in an IRC "PASS" command (this is used by the password support built into most clients).
5. If the answer is incorrect, the proxy will prompt the user to try again. Three wrong answers are permitted, and if authenticating against the local user database a slight delay will occur after each failure.

6. If the answer is correct the proxy will send all stored output from the client to the server, and begin normal operation. This should suffice to change the client's nickname to the value requested by the user.

Using passwords from the proxy server's user list is selected by the **-p** flag, and may require Tircproxy to run as root, depending on whether shadow passwords or some comparable system is installed or not. Administrators should beware that this option will require the passwords to be sent unencrypted over the network, which is generally not a good idea.

When using local usernames and passwords, the client should send its authentication formatted like this, "username%password", or this, "username password" (without the quotes). The latter is easiest when the user identifies himself by sending a message, but the former is useful when using the clients built-in password support command.

Using a quiz file (the **-q filename** option) is a much more flexible way to authenticate your users. It allows you to customize both the messages sent to the users when they connect and what replies to accept. A simple quiz file might contain only a single question, e.g. "Password please?" and a list of accepted passwords. A more interesting example is a file with many different questions. When Tircproxy is told to use such a file it will select a question at random. This gives proxy administrators a simple but effective way to keep people from running "bots", since the bots won't be smart enough to answer the questions presented to them.

Note that the quiz system is not case sensitive, it will treat "APPLES" and "apples" as the same word.

A simple quiz file might look like this:

```
# This is a comment, ignored by the proxy
#
!This is a generic message, sent to all users no matter which
!question is selected.
This is a question?:yes:no:maybe
Are elephants big?:yes
Is IRC a waste of time?:yes:no:maybe:of course not!
```

## 4.7 DCC support

The DCC features are all *enabled* by default.

Use the **-M** flag to disable the DCC blacklist (filename mangling), which will otherwise reject files with suspicious-looking names. Which file names are actually on the list can be configured at 4.2 (compile time) or using 4.6.1 (the TCP wrapper support).

Use the **-S** flag to disallow all DCC file transmissions.

Use the **-C** flag to disallow DCC chat.

Use the **-U** flag to disallow other (unknown) kinds of DCC.

The implications of these options are discussed in more detail in 2.1.2 (the chapter about IRC basics) and 3.4 (the Tircproxy feature overview).

## 4.8 Broadcasts

The broadcast and MOTD features need no command line switches, to activate them you merely make sure the required files (`/tmp/ircbroadcast` and `/etc/motd.irc` respectively) exist. The format for these files is the same - they may contain an arbitrary amount of text, which will be sent almost unmodified to the client. The only modification made to the text is the replacement of all occurrences of `$N$` with the current nickname of the client.

The format of the text file is therefore dictated by the IRC standard. Examples:

```
:nick!user@host PRIVMSG $N$ :the IRC proxy is going down for maintenance now!  
:user@hostname 999 * :this is a fake server message
```

Care should be taken not to send too much data to the user at once, since many users run client extensions (scripts) which will detect and suppress anything looking like a 2.1.1 (flood).

When sending messages like this, it is polite to actually be online under the nickname you used in the broadcast, so your users can respond to the message.

The file `/etc/motd.irc` will be sent to the user immediately after the "message of the day" from the remote IRC server has completed. The `/tmp/ircbroadcast` file will be sent whenever the Tircproxy process receives the 'hangup' signal (SIGHUP).

## 4.9 Identd support and user configuration

These features are *enabled* by default. To disable communication with the identd, use the `-I` flag.

If you are running Tircproxy for more than one user, you will probably want it to cooperate with the ident daemon on your system for reasons described in 3.6 (the Tircproxy feature overview) and 2.1 (the chapter about IRC basics).

This section does not cover the configuration or installation of the identd itself, but describes the different methods used for communication and how they are configured.

Two issues are involved - the configuration of Tircproxy's user list (who owns what IP address), and the mechanism used by Tircproxy to communicate with the local ident daemon. The user list is primarily used to determine *what* is sent to the identd, but it can also be used for access control, as described in 4.6.1 (the TCP wrapper compatibility chapter).

### 4.9.1 Shared memory based identd support

This is the preferred way to cooperate with ident daemons and configure the user list. If the UDB libraries are present on your system at compile time, support for this feature will be built in and enabled by default. It can be manually disabled by editing `tircproxy.h` before compiling.

Shared memory configuration requires that *UDB* <<http://bre.klaki.net/programs/udb/>> support be compiled into the proxy, and it's a good idea to have the UDB administration tools installed as well. If the local ident server is also UDB-aware (see *the UDB home page* <<http://bre.klaki.net/programs/udb/>> for a list) the shared memory tables will automatically be used for communication between the two programs.

This is ideal, since it does not require Tircproxy to run with root-privileges (thus avoiding many potential security issues), consumes very few resources and is quite fast.

UDB provides a persistent shared-memory database of user information, including information mapping users to IP addresses, and users to TCP/IP connections (such as those made by Tircproxy). Thus UDB can be used by Tircproxy to both acquire information about who is using it (by mapping IPs to user names), and pass this information on to a local UDB-enabled ident daemon (by mapping IRC connections to user names).

Users may be divided into two groups, depending on how they are configured - static users (users who always have the same IP address) and dynamic users. The more interesting case involves dynamic users. Since their address changes each time, you will somehow need to add each user and his IP address to the database when he connects, and remove the data when he disconnects. How to do this depends entirely on your environment - but a couple of common cases are covered in the following sections, followed by a brief discussion of how to handle static users.

### **Linux: dynamic PPP dial-in users**

When a PPP session is initiated on a Linux box, the Linux PPP daemon will run a script named `/etc/ppp/ip-up`. When they disconnect, it will run a script named `/etc/ppp/ip-down`. You can use these scripts, and the tools included with UDB, to record these changes. Simply add the following commands to `/etc/ppp/ip-up`:

```
TTY=$(echo $2|sed -e 's!/dev/!!')
USR=$(w -f -s -h |grep $TTY|cut -f1 -d\ )

udb_ipuser $5 $USR ""
```

And the following to `/etc/ppp/ip-down`:

```
udb_rm ip ip $5
```

(Note that on recent RedHat Linux systems (and possibly others), you should alter the files `/etc/ppp/ip-up.local` and `/etc/ppp/ip-down.local`, instead of the files named above.)

### **Dynamic users, authenticated using RADIUS**

At many ISPs, RADIUS servers are used to authenticate users dialing into specialized dial-in boxes (e.g. from Cisco or Cyclades).

At the moment no RADIUS server supports UDB, but support by the *FreeRADIUS* <<http://www.freeradius.org/>> server is in the works.

### **Static users**

The static users may simply be configured at system start-up time, since the UDB database is never deleted. This would probably be done either by creating a custom initialization script containing `udb_ipuser` commands like those described above, or by using the startup script included in the UDB distribution, and adding users to the `/etc/udb.conf` file.

### **Problems to avoid**

People unfamiliar with System V shared memory may run into problems using UDB for configuration and/or communication, since the same type of user and group permissions apply to shared memory segments as to

files in the filesystem. If the Tircproxy user doesn't have permission to write to the shared memory tables, or the ident daemon cannot read them, then communication will fail. If the Tircproxy user cannot read the tables, then the user information they contain will have no effect.

See the UDB documentation for details about how to use and/or troubleshoot UDB-aware applications (such as Tircproxy) effectively.

## 4.9.2 Filesystem based identd support (depreciated)

The alternative to shared-memory based identd support, is to use the filesystem. This method is slow and prone to race conditions, because it relies on files being created or deleted relatively quickly when dynamic users log on or off (user list configuration) or when new connections are established (identd communication). For these reasons (and others), the filesystem method is discouraged - people should use the UDB shared memory strategy if at all possible. By default this feature is built into the proxy at compile time, if (and only if) the UDB libraries are unavailable.

In spite of the drawbacks, the filesystem method usually works reasonably well and like the shared memory method it can allow Tircproxy to communicate with an ident daemon without running as root.

Assigning an IP address to a given user is accomplished by creating a text file name `user-<ip address>` containin nothing but the username on the first line.

Where to put this file depends on whether you have oidentd support built in or not. If oidentd support is built in, the default location is `/var/oidentd/`, but without it the default is `/var/run/`. Both values may be altered at compile time by editing `tircproxy.h`.

Configuring a static user (e.g. for IP 10.11.12.13) is as simple as this:

```
echo username > /var/oidentd/user-10.11.12.13
```

Dynamic users could be handled by adding similaur commands to scripts or logon systems (such as RADIUS).

A patched version of oidentd 1.4 was the only ident daemon able to communicate with Tircproxy using the filesystem. The necessary patches and a copy if the daemon are available on *the Tircproxy home page* <<http://bre.klaki.net/programs/tircproxy/>>. The modified oidentd daemon will check `/var/oidentd/` for files mapping individual TCP/IP connections to usernames, which are automatically created by Tircproxy if this feature is enabled.

## 4.9.3 Changing user IDs

This feature is *enabled* by default, if (and only if) the proxy is run as root. Disable it by specifying a non-root user (e.g. "nobody") on the command line, with the `-r username` arguement, or by directly running the proxy as a normal user.

If neither shared memory or filesystem based communication is desired (e.g. because it is not supported by your identd), the last resort to guarantee correct identification is to make the proxy actually assume the identity of the user it is working for. This requires that each user of the proxy have an account on the system (an entry in `/etc/passwd`), and that either the filesystem or UDB tables map all client IPs to such usernames.

If the client's IP address isn't found in the proxy user list, the connection will be rejected, unless the **-R** flag (relaxed mode) is present on the command line. Relaxed mode is not recommended on systems with standard ident daemons, because it can both reveal that the proxy is running as root (thus encouraging attackers to try to hack the proxy) and cause different users to receive the same identification (root).

Please note that running Tircproxy as root (especially in relaxed mode) is discouraged, because of the security risks involved: flaws in Tircproxy could expose your system to root compromise by outsiders - the worst kind of security hole on Unix-like systems. You have been warned!

## 4.10 Debugging and logging

Logging to syslog is *enabled* by default, disable it with the **-L** flag. When syslog use is disabled all log messages will be sent to "standard error".

Logging of nicknames is *enabled* by default, disable it with the **-D** flag.

Debugging logs are *disabled* by default. Enable them with the **-d level** option (level is a number from 0 to 9).

### 4.10.1 Syslog information

Tircproxy will by default send all log messages to syslog, using the DAEMON facility and tagging each line with the process ID and either the string **tircproxy** or **proxy**, depending on whether the proxy is running as an IRC proxy or in 3.2.1 (generic mode).

### 4.10.2 Debugging verbosity

If you are having trouble getting the proxy to work as expected, you can increase the verbosity of the logs it creates, by setting the debug level to a nonzero value. The defined levels are:

- **Level 0:** No debugging: normal logging only. This is the default.
- **Level 1:** Feature debugging: reports information about DCC handling and Quiz mode operation.
- **Level 2:** Basic debugging: warns about common "errors" (closed sockets etc.) and adds various status reports.
- **Level 8:** Trace mode: Prints various trivial messages about which parts of the code are being entered, the data looked up etc. Useful for isolating errors in the proxy.
- **Level 9:** Foreground mode: Prevents the proxy from forking into the background and forces all log messages to be sent to standard output (instead of syslog), like the **-L** flag.

The levels are incremental, level 9 includes all messages from level 8, level 8 contains all messages from level 7, and so on.

Levels 3-7 are at the moment identical to level 2, but are reserved for future development.

## 4.11 Miscellaneous features

### 4.11.1 Anonymizing mode

This feature is *disabled* by default. Enable it with the **-a** option.

Anonymizing mode is discussed in 3.7 (the Tircproxy feature list).

### 4.11.2 Sleeping between connections

This feature is *disabled* by default.

To prevent the proxy from flooding an IRC server with many simultaneous connections (e.g. if the proxy is rebooted), the proxy can be configured to enforce a one-second delay between connections under  $T$  seconds apart. This is done with the **-t  $T$**  option, like this:

```
tircproxy ... -t 2 ...
```

Without this option, some servers may temporarily refuse connections from the proxy host if many people try to connect at once. Since many clients will automatically try again, a vicious cycle can result, compounding the difficulties your users will have connecting to that server (the "crash, burp, get rejected, burp, ..." problem).

### 4.11.3 The "mIRC DCC kludge"

This feature is *enabled* by default, disable it with the **-K** flag.

The mIRC DCC kludge causes the proxy to ignore the IP address supplied by its clients when they request DCC connections, and instead use the address that actually connected to this proxy.

This was necessary because some versions of mIRC sent out invalid IP addresses in their DCC requests. This also closes some potential security holes where carefully constructed DCC requests could be used by the clients to punch arbitrary holes in your firewall.

This feature should probably never be disabled, unless you happen to be connecting to Tircproxy from *another* proxy (e.g. a bouncer) which doesn't have native support for DCC requests and is located on a different machine.

# Chapter 5

## Glossary

### 5.1 Acronyms and technical terms

This section contains definitions of some of the acronyms and technical concepts used in this manual. This probably won't suffice to get you up to speed if you are completely unfamiliar with IRC and networking, but hopefully it will help.

#### 5.1.1 chargen

"Chargen", or the character generator, is a TCP/IP service commonly available on Unix systems. It is generally located on port 19, and does nothing but recite the alphabet for you, repeatedly, as fast as it can.

Very stimulating!

#### 5.1.2 cracker

"Cracker" is the preferred term for a person who breaks into or otherwise violates computer system security without permission. Calling such people "hackers" really gets on *real* hackers nerves.

Read *the Jargon file's definition* <<http://www.tuxedo.org/~esr/jargon/html/entry/cracker.html>>.

#### 5.1.3 CTCP

CTCP, or Client To Client Protocol, defines a few special types of messages which the clients use to initiate DCC transmissions, trigger sounds or implement actions.

#### 5.1.4 DCC, DCC SEND and DCC CHAT

DCC, or Direct Client Connection, is a protocol used by IRC clients to send messages (DCC CHAT) or files (DCC SEND) directly, bypassing the IRC network.

See the 2.1.2 (subsection about the DCC protocol) for information about how DCC CHAT and DCC SEND are implemented.

### 5.1.5 DCC RESEND (and TRESEND)

DCC RESEND and TRESEND are enhancements to normal DCC SEND.

Before data transmission begins, the clients exchange information about at what file offset to begin, allowing the user to resume an uncompleted transmission.

### 5.1.6 DCC RESUME

DCC RESUME is a non-standard extension to DCC, introduced by the *mIRC* client to allow people to resume a previous, uncompleted file transmission.

This solution is inferior to the DCC RESEND protocol implemented in BitchX and other Unix-based clients, and requires special support from Tircproxy to work, because it violates the IRC protocol by replying to a CTCP request with another CTCP request instead of a reply.

### 5.1.7 DCC TSEND

DCC TSEND is an enhanced version of DCC SEND, which uses larger packet sizes to speed up transfers. Implementation is otherwise almost identical to traditional DCC SEND.

### 5.1.8 default gateway

A default gateway is a "last resort" router. That is, the router a machine sends traffic to if it has no better idea of where the traffic should go. Connections to the internet are usually default gateways.

### 5.1.9 IP

IP is the **Internet protocol**!

### 5.1.10 IPF

IPF is the "IP Filters" package, used by the BSD family of operating systems to allow fine-grained control over IP traffic (firewalling).

### 5.1.11 IP Masquerading

IP Masquerading is a variant of IP NAT built into the Linux kernel. It performs on the fly translation of multiple internal IP addresses into a single visible (external) address.

### 5.1.12 IP NAT

IP NAT is a type of firewalling service, which translates IP addresses on the fly from one source address to another for outgoing packets, and does the reverse for incoming packets.

This allows machines with otherwise invalid (non-unique) IP addresses to communicate with the rest of the Internet with relative ease.

### **5.1.13 IRC**

IRC is an acronym for "Internet Relay Chat". See the chapter about 2.1 (IRC basics) for more information.

### **5.1.14 proxy**

A proxy is an intermediate server, which makes requests on behalf of it's clients. A client doesn't need direct access to a server, if it can communicate with a proxy that does have such access.

### **5.1.15 route**

A "route" is the path followed by a TCP/IP packet as it passes from it's source to it's destination.

### **5.1.16 TCP**

TCP implements a reliable stream oriented full duplex stream between two sockets. TCP ensures that packets are not reordered and retransmits them when they are dropped. It generates and checks a per packet checksum to catch transmission errors.

*Text adapted from the tcp(4) manual page in the Linux Programmer's Manual.*

### **5.1.17 tcp wrappers**

TCP wrappers is the name of an access control method commonly used on Unix machines to limit access to certain TCP/IP services based on who is requesting them.

### **5.1.18 UDP**

User Data Protocol (a.k.a. Unreliable Data Protocol) implements a connectionless, unreliable datagram packet service. Packets may be reordered or duplicated before they arrive. UDP generates and checks checksums to catch transmission errors.

*Text adapted from the udp(4) manual page in the Linux Programmer's Manual.*

## Chapter 6

# GNU General Public License

GNU GENERAL PUBLIC LICENSE

Version 2, June 1991

Copyright (C) 1989, 1991 Free Software Foundation, Inc.  
675 Mass Ave, Cambridge, MA 02139, USA

Everyone is permitted to copy and distribute verbatim copies  
of this license document, but changing it is not allowed.

### Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software--to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Library General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.

#### GNU GENERAL PUBLIC LICENSE

#### TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program", below, refers to any such program or work, and a "work based on the Program" means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term "modification".) Each licensee is addressed as "you".

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

1. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

a) You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.

b) You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.

c) If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to

exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:

a) Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

b) Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

c) Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

4. You may not copy, modify, sublicense, or distribute the Program

except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

5. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

7. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made

generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

8. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

9. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

10. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

#### NO WARRANTY

11. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE

PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING,  
REPAIR OR CORRECTION.

12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING  
WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR  
REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES,  
INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING  
OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED  
TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY  
YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER  
PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE  
POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

#### Appendix: How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest  
possible use to the public, the best way to achieve this is to make it  
free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest  
to attach them to the start of each source file to most effectively  
convey the exclusion of warranty; and each file should have at least  
the "copyright" line and a pointer to where the full notice is found.

```
<one line to give the program's name and a brief idea of what it does.>  
Copyright (C) 19yy <name of author>
```

```
This program is free software; you can redistribute it and/or modify  
it under the terms of the GNU General Public License as published by  
the Free Software Foundation; either version 2 of the License, or  
(at your option) any later version.
```

```
This program is distributed in the hope that it will be useful,  
but WITHOUT ANY WARRANTY; without even the implied warranty of  
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the  
GNU General Public License for more details.
```

```
You should have received a copy of the GNU General Public License  
along with this program; if not, write to the Free Software  
Foundation, Inc., 675 Mass Ave, Cambridge, MA 02139, USA.
```

Also add information on how to contact you by electronic and paper mail.

If the program is interactive, make it output a short notice like this  
when it starts in an interactive mode:

Gnomovision version 69, Copyright (C) 19yy name of author  
Gnomovision comes with ABSOLUTELY NO WARRANTY; for details type 'show w'.  
This is free software, and you are welcome to redistribute it  
under certain conditions; type 'show c' for details.

The hypothetical commands 'show w' and 'show c' should show the appropriate parts of the General Public License. Of course, the commands you use may be called something other than 'show w' and 'show c'; they could even be mouse-clicks or menu items--whatever suits your program.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a "copyright disclaimer" for the program, if necessary. Here is a sample; alter the names:

Yoyodyne, Inc., hereby disclaims all copyright interest in the program  
'Gnomovision' (which makes passes at compilers) written by James Hacker.

<signature of Ty Coon>, 1 April 1989  
Ty Coon, President of Vice

This General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Library General Public License instead of this License.