

## IRC & Security - A comprehensive guide

### Introduction

Internet Relay Chat (IRC) is a form of [instant communication](#) over the Internet. It is mainly designed for group (many-to-many) communication in discussion forums called [channels](#), but also allows one-to-one communication.

If you are not familiar about what IRC is, then I would suggest you read the [Intro to IRC](#) on the [mIRC](#) homepage. If you are already well acquainted with IRC and you are looking for a comprehensive-commands-guide, then you can also checkout the [IRC command list](#) also found on the mIRC homepage.

This guide will show you the different methods you can use to increase the privacy of your online conversations as well as way to conceal your identity a bit better.

### Use `/identify` wherever possible

Once you have registered on a network you will need to identify yourself each time you return. Most people do this by using the command `/msg NickServ identify $password`. Although this works, if services are compromised (As was the case in the [recent Freenode hack](#)) then someone else can use the name `NickServ` and receive your password in plaintext.

The best way to protect yourself against this is to use the command `/identify` on any network that supports it. This command actually messages `nickserv@services.network.com` instead of just `NickServ`. If services were compromised and you were using the identify command then your password wouldn't be compromised.

### 3. Vanity hosts (also known as vHost)

It was very apparent when IRC was launched and became the premier tool for chatting that privacy would become a big issue. Then came the vanity host, or its more popular name...vHost.

A host is a name by which a user is known on a network. A vhost is, basically, an IRC tool to protect the privacy of a user by hiding his identity. It masks a user's identity, thereby making it highly difficult for anyone to launch attacks. There are basically two types of vHosts:

1. The old fashioned bounce vhost which stores real information of a user.
2. The newly launched form of vhost which protects a user by allowing entry into a customized hostname. This vhost requires checks to ensure that no real hostnames are duplicated and also makes sure that there is no other user with the same hostname. The most popular example is `hostserv`.

Most IRC servers have `hostserv` installed, so the following is a quick overview of some commands to get you set up and running with `hostserv`:

- ON

Syntax: */msg HostServ ON*

This command activates the vhost and automatically reverts to the default vhost assigned to you.

- OFF

Syntax: */msg HostServ OFF*

This command deactivates the vhost currently assigned to the user and gives the user his or her real identity back.

- SET

Syntax: */msg HostServ SET \$nick \$hostmask.*

This command sets up a hostname for the selected nick. This is generally limited to hostname setters on a server.

- DEL

Syntax- */msg HostServ DEL \$nick*

This deletes the vhost assigned to the nickname from the database. It is limited to host removers on the IRC network.

- LIST

Syntax: */msg HostServ LIST [\$key|<#X-Y>]*

This command lists entries from the database corresponding to the keyword.

X and y specifies the no. of entries. For example, #1-5 will display the first five entries.

This is limited to the IRC service operators.

- GROUP

Syntax: */msg HostServ GROUP*

This command sets up the vhost of a particular nickname to be the same as the hostname of all the other users in the group.

## **Proxies**

With the need to use IRC, comes the need to be anonymous. Although vhosts do fulfill some of the necessary needs but a proxy is an even more powerful tool to disguise all your net communications.

A proxy server is basically another computer through which you can route all your connections to any location on the net. This can be particularly useful in the case of IRC communications because it protects your real location, and therefore allows you anonymous access. There are many forms of proxies which act almost the same, and have only a few minor differences. I won't attempt to explain each particular type of proxy here as it would simply take too long. Instead, if you would like to know a bit more about a particular proxy type then click it to be taken to the relevant [Wikipedia](#) page: [Web Proxies](#), [SSL Proxies](#), [Transparent proxies](#), [Open proxies](#), [Reverse proxies](#), [Split Proxies](#).

Proxies have three main functions:

*Firewalling and filtering:*

Proxies use advanced form of firewalls to filter access to any web page they see unfit, they can also bar a connection from a remote computer.

*Connection sharing:*

Rather than allowing users on a network different direct connection, all connections can be routed through one or two proxies.

*Caching:*

Caching is a temporary storage area where frequently accessed data can be stored for rapid access. Caching can improve both load time and response time, thereby leading to a better usage of connection.

There are plenty of sites that you can find that will teach you how to use proxies (to avoid School or Uni webpage filtering and for IRC) so I will cover the most popular method here which is using a proxy through a web-browser. This is particularly useful if you are using web-based chat for your IRC sessions.

1. Go to 'Tools' in the menu of IE.
2. Go to 'Internet Option'
3. Now, click on the 'Connection' tab
4. Select 'LAN setting'
5. Once inside, check the use proxy box.
6. Now you can type in a proxy. If you need a proxy then you can easily find one by searching for "proxy list" on any search engine.

**Bounce networks** (*also known as BNC*)

A [BNC](#) is a program that runs as a [daemon](#) on a [server](#) and functions as a persistent [proxy](#).

A bouncer is used to maintain a connection to the [IRC server](#), acting as a relay between the server and the connecting client. It is named aptly, because should the client lose network connectivity or have another problem causing loss of connection to the IRC server, the bouncer saves all communications and messages for later by 'bouncing' the client to resuming a IRC session later on without any external disruption.

Some of the most popular bouncers are [Muh](#), used exclusively for single, one on one session, and [PsyBNC](#), the most popular BNC for multiple user chats such as a group. Another popular bouncer which is very feature rich is [ZNC](#).

Now, setting up a bounce system is very easy. The steps given below apply to all users, no matter if you use [Windows](#), [Linux](#), or [OS X](#). There are three more commonly used versions of bouncer, so I have given step by step instructions for the most popular (BNC 2.2.x). The steps are common for the second and third versions too, and can be easily found in the same directory as the first server. The BNC server can be changed, however, if you already have some other version, or if you want to install it in a different way.

### Instructions for setting up BNC 2.2.x...

First of all you will need to identify the latest version of BNC available for download from the [GotBNC website](#). At the writing of this tutorial the latest version available was BNC 2.8.4.

Once you have downloaded the package, extract it and run `./configure`, go through the steps of configuring your BNC and then type `make` or `gmake` to compile the package.

Once your BNC is compiled you can proceed to editing the `bnc.conf` file. The following explains some of the options in this file and gives you values you could set them too if you are unsure:

`pt:1234567`

Specifies the daemon port (local port) that BNC is to be mounted to

`ps:mypassword`

This specifies the password needed for the client which in this case has been set to "mypassword".

`mu:2`

Maximum users - the maximum connections you will allow to your BNC at any one time.

`dp:6666`

The default port for your vnc. This is the port used when you say `/conn [port]`

`vh:my.vhost.com`

This specifies the host to use.

### Logging in and using your BNC

Once you have configured your BNC open your IRC Client use the following commands to connect, and utilise you BNC.

1. `/server $vhost.address $port`
2. `/quote pass $password`
3. `/quote conn $server`

### Blowfish encrypted chat sessions

The most revolutionary encryption algorithm which changed the cryptography world is now known as [Blowfish](#).

Designed by [Bruce Schneier](#) in 1993, it was meant as a general-purpose algorithm, which wanted to end the problems of all contemporary algorithms. It is actually a keyed, symmetrical block cipher. It was unpatented, unlike all other secretly patented algorithms, and has remained unpatented even today.

The block size of the algorithm is 64 bits. The default key is 128 bits, but it can support 32-448 bits in steps of 8 bits. It is very well designed, and until now, only brute forcers have been able to crack it open.

This section gives details on how to install the most popular Blowfish encryption program, Fish 1.28a. This is an addon for IRC programs and automatically protects sensitive data. This program is open-source and you can download it from [fish.sekure.us](#).

### **Installation for mIRC in Windows**

1. Unload ALL old blowcrypt/Mircryption files in mIRC (ALT+R: blow.mrc, blowcrypt.mrc, mircryption.mrc, etc.)
2. Place FiSH.DLL, FiSH.mrc into same directory as mIRC.exe (as well as your old blow.ini, or use blow.ini-EXAMPLE)
3. Load FiSH.mrc into mIRC: `/load -rs1 FiSH.mrc`
4. Close mIRC
5. Apply the appropriate patch (mIRC.vX.X.FiSH-Addon.v1.0B.exe)

### **Installation for irssi in Linux**

1. Copy libfish.so to irssi's module directory and use `/load fish` or `/load /path/to/libfish.so` This is a plug-in, not a script - so do NOT use `/script` to load it
2. Locate your blow.ini in `~/.irssi/` (or wherever your irssi config file is stored) and configure it.

### **Installation in XChat for OS/X and other Operating Systems**

1. Copy xfish.so to XChat's plug-in directory. It will usually be loaded on XChat startup. If not, you could also use `/load /path/to/xfish.so`
2. XChat for windows expects xFiSH.dll in `\xchat\plugins`
3. In case you have a custom blow.ini password you could also use following command to load FiSH: `/load /path/xfish.so $password` (linux version only)
4. Blow.ini should be located in your XChat home directory  
Linux: `/home/user/.xchat2`  
Windows: `C:\Documents and Settings\User\Application Data\X-Chat 2`

## Blow.ini options explained

process\_incoming=1

Decrypt incoming messages?

process\_outgoing=1

Encrypt outgoing messages?

mark\_encrypted=" •"

see below

plain\_prefix="+p "

Messages starting with +p will be sent as plain-text

**If you want to mark INCOMING encrypted messages, add mark\_encrypted to blow.ini, in [FiSH] section.**

## Parting note

Well that about covers my tutorial on Internet Relay Chat Security! If you found anything in here useful or have seen something you feel to be incorrect then please leave a comment!

You can find me on my own network [Jamscone](#) at [#jamscone](http://irc.jamscone.com) as skelm. If you have any questions about this article or would just like to pop in for a chat! I would love to hear from you!

*Michael Skelton*

[info@codingo.net](mailto:info@codingo.net)