jmox

I am `$(whoami)`.
About myself, I can say:

```
lslogins \
--noheadings \
-o=GECOS -l $(whoami)
```

Follow

Secure IRC connection to Freenode with Tor and WeeChat

5 minute read

Content

Tor

- Test it

WeeChat

- Improving privacy

Freenode

- Registering on Freenode
- Enabling SASL EXTERNAL
- Connection over TOR
- Shortcuts WeeChat
  - join/part channels

private messages
buffer/window management
Key bindings

Let's say we want to join a hax0rs irc channel to chat with some black hat folks. We wouldn't want to expose our ip address, right? We are going to do it using Tor to access freenode on the irc client WeeChat.

Pretty much the same would work if we wanted to connect to other IRC networks that have an onion address, like AlphaChat.

We want root.

```
sudo -i
```

Tor

No further configuration is needed here, just install Tor service and enable it:

```
apt install tor
systemctl enable tor.service
systemctl start tor.service
```

We will use the SOCKS proxy to connect to Tor. Tor listens for `SOCKS` connections on port `9050` and Tor Browser on `9150`.

In case some troubleshooting is needed, the files that might help here are:

`/var/log/tor/log`
`/etc/tor/torsocks.conf`

Also confirm that Tor is listening on `9050`.

```
# ss -ltap | grep 9050

LISTEN  0  0  127.0.0.1:9050  *:*  users:(("tor",pid=2177,fd=6))
```

Test it

```
curl --socks5 localhost:9050 \
     --socks5-hostname localhost:9050 \
     -s https://check.torproject.org/ \
     | cat | grep -m 1 Congratulations | xargs
```

The output should be something like this:

```
Congratulations. This browser is configured to use Tor.
```

That's the official documentation for Debian: Tor documentation.

WeeChat

WeeChat is the IRC client we will use to chat with the pals in freenode. There are other options out there. It's just a matter of taste.

```
apt-get install weechat
```

And we start WeeChat by running:

```
weechat-curses
```

Improving privacy

Add somes settings bellow to WeeChat.

```
/set irc.server_default.msg_part ""
/set irc.server_default.msg_quit ""
/set irc.ctcp.clientinfo ""
/set irc.ctcp.finger ""
/set irc.ctcp.source ""
/set irc.ctcp.time ""
/set irc.ctcp.userinfo ""
/set irc.ctcp.version ""
/set irc.ctcp.ping ""
/plugin unload xfer
/set weechat.plugin.autoload "*,!xfer"
```

Freenode

Freenode like other IRC networks has some requirements when it comes to connecting via Tor, like registering a nickname and connecting using SASL.

To connect to Freenode we will use the following hidden service as the server address provided by Freenode:

`ajnvpgl6prmkb7yktvue6im5wiedlz2w32uhcwaamdiecdrfpwwgnlqd.onion`

This is the link to the Tor freenode's docs:

- Accessing freenode via TOR

> *The hidden service requires SASL authentication. In addition, due to the abuse that led Tor access to be disabled in the past, we have unfortunately had to add another couple of restrictions.*

We must log in using `SASL EXTERNAL` or `ECDSA-NIST256P-CHALLENGE`:

- If you log out while connected via Tor, you will not be able to log in without reconnecting.
- It is recommended to use SASL EXTERNAL.

Connecting using SASL EXTERNAL requires connecting using SSL encryption.

The SSL certificates don't match the hidden services, therefore is not necessary to do any verification on the certs.

If you don't want to disable the verification in WeeChat, you can map the freenode address to the onion hidden service. Add this line to the `/etc/tor/torrc` file:

```
# torrc snippet:
MapAddress zettel.freenode.net ajnvpgl6prmkb7yktvue6im5wiedlz2w32uhcwaamdiecdrfpwwgnlqd.onion
```

Don't forget to reload tor service:

```
systemctl reload tor.service
```

Registering on Freenode

On WeeChat:

```
/server add freenode chat.freenode.net/6667 -autoconnect
/set irc.server.freenode.nicks mycoolnickname
/connect freenode
```

We have to create an account, this is a requirement to use TOR:

```
/msg NickServ REGISTER mypassword mycoolemail@example.com
```

```
/msg NickServ SET PRIVATE ON
```

Confirm registration after getting the mail with the code:

```
/msg NickServ VERIFY REGISTER mycoolnickname code1235678
```

To identify ourselves:

```
/msg NickServ IDENTIFY mypassword
```

Enabling SASL EXTERNAL

```
mkdir ~/.weechat/certs
cd ~/.weechat/certs
```

```
openssl req -x509 -new -newkey rsa:4096 -sha256 -days 1000 -nodes -out freenode.pem -keyout freenode.pem
```

Find sha1sum fingerprint:

```
openssl x509 -in freenode.pem -outform der | sha1sum -b | cut -d' ' -f1
```

And add the fingerprint on WeeChat, eg: fingerprint `0123456789abcdefghijklmnopqrst1234567890`

```
/msg nickserv cert add 0123456789abcdefghijklmnopqrst1234567890
/set irc.server.freenode.ssl_cert "%h/certs/freenode.pem"
/set irc.server.freenode.sasl_mechanism external
/set irc.server.freenode.ssl on
/set irc.server.freenode.addresses "chat.freenode.net/6697"
/reconnect freenode
```

Connection over TOR

Now that we have our nickname and our certificate we can connect to freenode via tor. Below are the steps needed but feel free to check out the official information:

- https://freenode.net/kb/answer/chat#accessing-freenode-via-tor

Add the Onion adress and the proxy:

```
/set irc.server.freenode.addresses "ajnvpgl6prmkb7yktvue6im5wiedlz2w32uhcwaamdiecdrfpwwgnlqd.onion/7000"
/proxy add tor socks5 127.0.0.1 9050
/set irc.server.freenode.proxy "tor"
```

We disable ssl_verify, which doesn't work with TOR.

```
/set irc.server.freenode.ssl_verify off
/reconnect freenode
```

Shortcuts WeeChat

This section is just for me to remember some WeeChat shortcuts. It's part of the official documentation of WeeChat that you can find here.

join/part channels

Join a channel:

```
/join #channel
```

Part a channel (keeping the buffer open):

```
/part [quit message]
```

Close a server, channel or private buffer (/close is an alias for /buffer close):

```
/close
```

Closing the server buffer will close all channel/private buffers.

Disconnect from server, on the server buffer:

```
/disconnect
```

private messages

Open a buffer and send a message to another user (nick foo):

```
/query foo this is a message
```

Close the private buffer:

```
/close
```

buffer/window management

A buffer is a component linked to a plugin with a number, a category, and a name. A buffer contains the data displayed on the screen.

A window is a view on a buffer. By default there's only one window displaying one buffer. If you split the screen, you will see many windows with many buffers at same time.

Commands to manage buffers and windows:

```
/buffer
/window
```

For example, to vertically split your screen into a small window (1/3 width), and a large window (2/3), use command:

```
/window splitv 33
```

To remove the split:

```
/window merge
```

Key bindings

WeeChat uses many keys by default. All these keys are in the documentation, but you should know at least some vital keys:

Alt+← / Alt+→ or F5 / F6: switch to previous/next buffer

F1 / F2: scroll bar with list of buffers (“buflist”)

F7 / F8: switch to previous/next window (when screen is split)

F9 / F10: scroll title bar

F11 / F12: scroll nicklist

Tab: complete text in input bar, like in your shell

PgUp / PgDn: scroll text in current buffer

Alt+a: jump to buffer with activity (in hotlist)

According to your keyboard and/or your needs, you can rebind any key to a command with the /key command. A useful key is Alt+k to find key codes.

For example, to bind Alt+! to the command /buffer close:

```
/key bind (press alt-k) (press alt-!) /buffer close
```

You'll have a command line like:

```
/key bind meta-! /buffer close
```

To remove key:

```
/key unbind meta-!
```

Tags: debian irc privacy tor weechat

Categories: sysadmin

Updated: June 13, 2019

Previous Next

YOU MAY ALSO ENJOY

Wireguard VPN Plugin for NetworkManager in GNOME

January 27, 2021 1 minute read

Wireguard is a layer-3 secure network tunnel for ipv4 and ipv6. It has managed to remove the complexity that other solutions/VPN

Atlassian Cloud: Pros and Cons

December 24, 2020 less than 1 minute read

Many companies are considering whether or not to migrate their Atlassian on-premise products to the Cloud option after announcing Atlassian that their Server...

Preparing ACP-600 Project Administration in Jira Server: Part II Labs

September 13, 2020 4 minute read

Preparing ACP-600 Project Administration in Jira Server: Part I Concepts to master

September 12, 2020 16 minute read

protocols like OpenVPN or IP...

Let’s talk about the 24 hours lab that Atlassian provides when you buy the ACP-600 preparation course + exam bundle.

Here you can find what helped me to pass the exam ACP-600 Project Administration in Jira Server a couple of days ago. I hope it helps you too.