How to setup a hidden service

1. Setup OS

As secure as you can (disk encryption, custom kernel if possible, etc)

2. Setup dnscrypt-proxy

Check that there are no processes listening to port 53 (domain):

```
ss -lp 'sport = :domain'
```

If there are any, terminate them.

Install the dnscrypt-proxy package from your distribution.

Edit the dnscrypt-proxy.toml file **if required**, it should be in /etc/dnscrypt-proxy/dnscrypt-proxy.toml

If no dnscrypt-proxy.toml file is present, you are on Debian and you probably have a jurassic version of dnscrypt-proxy. I use arch btw.

It should work by default anyways.

If you are using **Arch Linux**, you probably need to set up netctl to use dnscrypt-proxy first. To do this, copy one of the configuration files on /etc/netctl/examples and edit it accordingly. For instance:

```
# cp /etc/netctl/examples/ethernet-dhcp /etc/netctl/enpXsY
```

```
(change enpXsY with the name of the network device you are using, you can know
with the command 'ip a').
```

Set it up accordingly, but the most important part is to set the DNS to localhost. So make sure the following line is present:

```
DNS=('127.0.0.1')
```

Save and then enable this configuration for your network device

```
netctl enable enpXsY
ip link set dev enp0s3 down
netctl start enpXsY
```

Wait a few seconds for your DHCP to pick up with you and then check if your DNS connection is working.

For other systems you might need to edit the resolv.conf file (*/etc/resolv.conf*). Comment whatever is there, and add the following lines:

```
nameserver 127.0.0.1
options enpXsY single-request-reopen
```

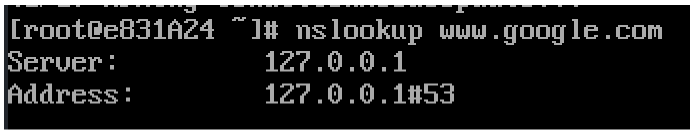
Start and enable dnscrypt-proxy

```
systemctl start dnscrypt-proxy
systemctl enable dnscrypt-proxy
```

To check if all of this is working, install the dnsutils package, then:

```
nslookup www.google.com
```

This should tell you that the nameserver is set to 127.0.0.1



```
[root@e831A24 ~]# nslookup www.google.com
Server:         127.0.0.1
Address:        127.0.0.1#53
```

Static IP with netctl

This is the content of your netctl profile if you need to have a static IP (change enpXsY with the name of your device):

```
Description='enpXsY Static IP'
Connection=ethernet # 'ethernet' for wired connections, 'wireless' for wireless
IP=static
Address=('192.168.1.XXX/24') # This is your IP (change XXX or the entire address)
Routes=('192.168.0.0/24 via 192.168.1.2') # This is optional
Gateway=('192.168.1.1') # This is your gateway, usually your router
DNS=('127.0.0.1')
```

Make sure to disable any DHCP service you have. Either:

```
systemctl stop dhcpcd@enpXsY
```

Or

```
systemctl stop dhcpcd
```

Depending on your configuration. If you are using any other DHCP daemon, stop it.

3. Setup web server

For this I'm using lighttpd. Install it, and if you want SSL with self-signed certificates, instal openssl too (otherwise ignore it).

# pacman -S lighttpd openssl

If you are using self-signed certificates, create one.

```
# mkdir /etc/lighttpd/certs
# openssl req -x509 -nodes -days 7300 -newkey rsa:2048 -sha256 -keyout
/etc/lighttpd/certs/server.pem -out /etc/lighttpd/certs/server.pem

# chmod 600 /etc/lighttpd/certs/server.pem
```

Make sure the information you give does not deanonimize your hidden service.

The configuration file must be created on /etc/lighttpd/lighttpd.conf with the following command:

```
lighttpd -t -f /etc/lighttpd/lighttpd.conf
```

Edit it. First thing you have to do is to disable the directory listing. This can be easily done by setting the dir-listing.activate option to “disable”.

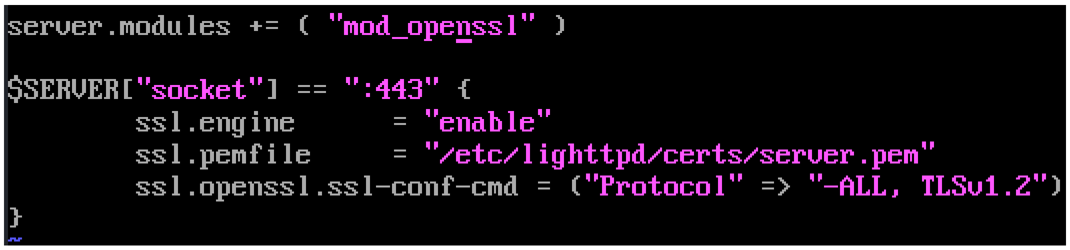
```
Dir-listing.activate    = “disable”
```

Next, if we have a SSL certificate, is to setup the port 443. Add the following code:

```
server.modules += ( “mod_openssl” )

$SERVER["socket"] == ":443" {
    ssl.engine                  = "enable"
    ssl.pemfile                 = "/etc/lighttpd/certs/server.pem"
    ssl.openssl.ssl-conf-cmd = ("Protocol" => "-ALL, TLSv1.2")
# ONLY if you are using a regular certificate (otherwise skip the next line)
    ssl.ca-file                 = "/etc/letsencrypt/live/domain/fullchain.pem"

 }
```



```
server.modules += ( "mod_openssl" )

$SERVER["socket"] == ":443" {
        ssl.engine      = "enable"
        ssl.pemfile     = "/etc/lighttpd/certs/server.pem"
        ssl.openssl.ssl-conf-cmd = ("Protocol" => "-ALL, TLSv1.2")
}
~
```

If you need to redirect your traffic from port 80, add this too:

```
$SERVER["socket"] == ":80" {
  $HTTP["host"] =~ ".*" {
    url.redirect = (".*" => "https://%0$0")
  }
}
```

Save and start lighttpd. Don’t forget to enable it.

```
systemctl start lighttpd
systemctl enable lighttpd
```

```
Make sure the server is up and running and that there are no errors

systemctl status lighttpd
```

Make a test html page. By default your content should be in *//srv/http unless other directory is specified on your lighttpd.conf file:*

```
echo “The server is working” >> /srv/http/index.html
```

Then connect to it using a web browser.

```
lynx https://127.0.0.1
```

You should see your page working.

4. Setup tor

Make sure there are no applications listening into port 9050 or whatever port you will be using

```
ss -lp 'sport = :9050'
```

If there are any, they must be terminated.

Install the tor package from your distro.

```
# pacman -S tor
```

Edit the torrc file. It’s default location should be /etc/tor/torrc

Uncomment/Add the following lines:

```
HiddenServiceDir /var/lib/tor/hidden_service/
HiddenServicePort 80 127.0.0.1:80
HiddenServicePort 443 127.0.0.1:443 #← ONLY WHEN USING SSL!
```

Do NOT activate the relay unless you know what’s good for you.

Save and restart TOR (systemctl restart tor).

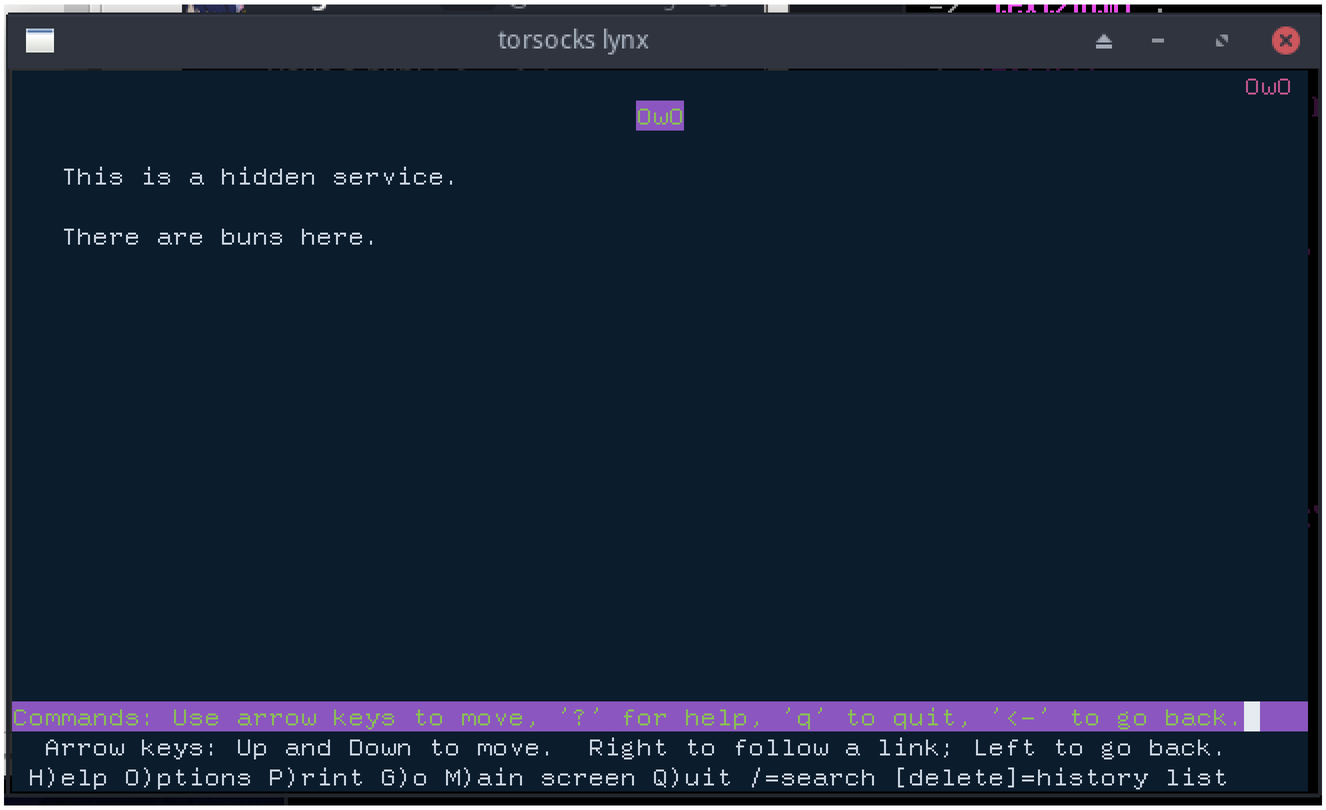
Check your address

```
cat /var/lib/tor/hidden_service/hostname
```

This is the address used to connect to your site. Use Tor browser or a torified browser to connect to it.

```
$ torsocks lynx https://whateverlongaddresstorgaveyou.onion
```

Your browser will warn you if your certificate is self-signed. Add a security exception to it (or answer 'yes' to all questions lynx asks you about it). You should see your site up and running.



```
                                                              OwO
                                     OwO

   This is a hidden service.

   There are buns here.

Commands: Use arrow keys to move, '?' for help, 'q' to quit, '<-' to go back.
  Arrow keys: Up and Down to move.  Right to follow a link; Left to go back.
 H)elp O)ptions P)rint G)o M)ain screen Q)uit /=search [delete]=history list
```

PHP

Using php-cgi

Install php and php-cgi (see also PHP and LAMP).

Check that php-cgi is working `php-cgi --version`

```
PHP 5.4.3 (cgi-fcgi) (built: May  8 2012 17:10:17)
Copyright (c) 1997-2012 The PHP Group
Zend Engine v2.4.0, Copyright (c) 1998-2012 Zend Technologies
```

If you get a similar output then php is installed correctly.

Create a new configuration file:

```
/etc/lighttpd/conf.d/fastcgi.conf

# Make sure to install php and php-cgi. See:
# https://wiki.archlinux.org/index.php/Fastcgi_and_lighttpd#PHP

server.modules += ("mod_fastcgi")

# FCGI server
# ===========
#
# Configure a FastCGI server which handles PHP requests.
#
index-file.names += ("index.php")
fastcgi.server = (
    # Load-balance requests for this path...
    ".php" => (
        # ... among the following FastCGI servers. The string naming each
        # server is just a label used in the logs to identify the server.
        "localhost" => (
            "bin-path" => "/usr/bin/php-cgi",
            "socket" => "/tmp/php-fastcgi.sock",
            # breaks SCRIPT_FILENAME in a way that PHP can extract PATH_INFO
            # from it
            "broken-scriptfilename" => "enable",
            # Launch (max-procs + (max-procs * PHP_FCGI_CHILDREN)) procs, where
            # max-procs are "watchers" and the rest are "workers". See:
            #
https://redmine.lighttpd.net/projects/1/wiki/frequentlyaskedquestions#How-many-
php-CGI-processes-will-lighttpd-spawn
            "max-procs" => 4, # default value
            "bin-environment" => (
                "PHP_FCGI_CHILDREN" => "1" # default value
            )
        )
    )
)
```

Make lighttpd use the new configuration file by appending the following line to your lighttpd configuration file:

```
/etc/lighttpd/lighttpd.conf

include "conf.d/fastcgi.conf"
```

Note: Remember that the order in which the modules are loaded is important, the correct order is listed in `/usr/share/doc/lighttpd/config/modules.conf`.

Reload lighttpd.

FastCGI

Install fcgi. Now you have lighttpd with fcgi support. If it was that what you wanted you are all set. People that want Ruby on Rails, PHP or Python should continue.

Note: New default user and group: Instead of group `nobody` lighttpd now runs as user/group `http` by default.

First copy the example config file form

/usr/share/doc/lighttpd/config/conf.d/fastcgi.conf to /etc/lighttpd/conf.d

The following needs adding to the config file, /etc/lighttpd/conf.d/fastcgi.conf

```
server.modules += ( "mod_fastcgi" )

#server.indexfiles += ( "dispatch.fcgi" ) #this is deprecated
index-file.names += ( "dispatch.fcgi" ) #dispatch.fcgi if rails specified

server.error-handler-404   = "/dispatch.fcgi" #too
fastcgi.server = (
    ".fcgi" => (
      "localhost" => (
        "socket" => "/run/lighttpd/rails-fastcgi.sock",
        "bin-path" => "/path/to/rails/application/public/dispatch.fcgi"
      )
    )
)
```

Then in /etc/lighttpd/lighttpd.conf:

```
include "conf.d/fastcgi.conf"
```