

# port80

## How do I use port 80 as a non root user?

On Unix based systems, port 80 is protected and can usually only be opened by the superuser root. As it is not desirable to run the server as root (for security reasons), the solution options are as follows:

- Start Jetty as the root user, and use Jetty's setuid mechanism to switch to a non-root user after startup.  
or
- Configure the server to run as a normal user on port 8080 (or some other non protected port). Then, configure the operating system to redirect port 80 to 8080 using ipchains, iptables, ipfw or a similar mechanism.

The latter has traditionally been the solution, however Jetty 6.1 has added the new setuid feature.

If you are using Solaris 10, you may not need to use this feature, as Solaris provides a User Rights Management framework that can permit users and processes superuser-like abilities. Please refer to the [Solaris documentation](#) for more information.

## Using Jetty's setuid (and setumask) feature

Create a jetty config file like so:

```
<?xml version="1.0"?>
<!DOCTYPE Configure PUBLIC "-//Mort Bay Consulting//DTD Configure//EN"
"http://jetty.mortbay.org/configure.dtd">
<Configure id="Server"
class="org.mortbay.setuid.SetUIDServer">
  <Set name="umask">UMASK</Set>
  <Set name="uid">USERID</Set>
</Configure>
```

Where you replace:

- **UMASK** with the umask setting you want the process to have, or optionally remove this line if you don't want to change this at runtime
- **USERID** with the id of the user you want the process to execute as once the ports have been opened.

### Hint

For your convenience, you'll find one of these ready made in the  
\$jetty.home/extras/setuid/etc/jetty-setuid.xml.

Then, you need to build the setuid feature for your operating system, as it requires native libraries. Go to the `$jetty.home/extras/setuid` directory and follow the instructions in the `README.txt` file, summarized here as:

```
> mvn install

> gcc -I$JDK_HOME/include/ -I$JDK_HOME/include/linux/ \
    -shared src/main/native/org_mortbay_setuid_SetUID.c \
    -o ../../lib/ext/libsetuid.so

> cp target/jetty-setuid-6.1-SNAPSHOT.jar ../../lib/ext/
> cp etc/jetty-setuid.xml ../../etc
```

Where:

- **\$JDK\_HOME** is same as `$JAVA_HOME`
- **linux** should be replaced by the name of your operating system.

#### 🚫 On Solaris

Leave out the `-shared` argument.

Then to run jetty as the root user, switching to the userid of your choice (and setting the umask of your choice if you chose to do that) you do:

```
sudo java -Djava.library.path=lib/ext -jar start.jar etc/jetty-setuid.xml
etc/jetty.xml
```

#### 🚫 Note!

You **must** ensure that the `etc/jetty-setuid.xml` file is first in the list of config files.

## Using ipchains

On some Linux systems the ipchains REDIRECT mechanism can be used to redirect from one port to another inside the kernel:

```
/sbin/ipchains -I input --proto TCP --dport
80 -j REDIRECT 8080
```

This basically means, "Insert into the kernel's packet filtering the following as the first rule to check on incoming packets: If the protocol is TCP and the destination port is 80, redirect the packet to port 8080." Your kernel must be compiled with support for ipchains. (virtually all stock kernels are.) You must have the "ipchains" command-line utility installed. (On RedHat the package is aptly named "ipchains".) You can run this command at any time, preferably just once since it inserts another copy of the rule every time you run it.

Once this rule is set up, a Linux 2.2 kernel will redirect all data addressed to port 80 to a server such as Jetty running on port 8080. This includes all RedHat 6.x distros. Linux 2.4 kernels, e.g. RedHat 7.1+, have a similar "iptables" facility.

## Using iptables

You need to add something like the following to the startup scripts or your firewall rules:

```
/sbin/iptables -t nat -I PREROUTING -p tcp
--dport 80 -j REDIRECT --to-port 8080
```

The underlying model of iptables is different to that of ipchains so the forwarding normally only happens to packets originating off-box. You will also need to allow incoming packets to port 8080 if you use iptables as a local firewall.

Be careful to place rules like this one early in your "input" chain. Such rules must precede any rule that would accept the packet, otherwise the redirection won't occur. You can insert as many rules as needed if your server needs to listen on multiple ports, as for HTTPS.

## Using usermod

On Solaris 10 (maybe earlier versions too) the OS allows you to grant privileged ports binding to "normal" users:

```
usermod -K defaultpriv=basic,net_privaddr
myself
```

Now the `myself` user will be able to bind to port 80.

## Using xinetd

With modern Linux flavours, inetd has a newer, better big brother xinetd. I'm not going to get into detail about it, there are plenty of man pages etc out there.

But the point is that you can use xinetd to redirect network traffic, and all you need is a text editor.

xinetd is driven by text files. Now there's 2 ways to give xinetd instructions:

1. Add a new service to `etc/xinetd.conf`
2. Add a new file to the directory `etc/xinetd.d`

Take your pick, the format is the same, if you have a look at the file/directory, you will get the picture.

The following entry will redirect all inward tcp traffic on port 80 to port 8888 on the local machine. Of course you can redirect to other machines for gimp proxying:

```
service my_redirector
{
  type = UNLISTED
  disable = no
  socket_type = stream
  protocol = tcp
  user = root
  wait = no
  port = 80
  redirect = 127.0.0.1 8888
  log_type = FILE /tmp/somefile.log
}
```

### Points to Note

- Space on either side of the '=' or it is ignored.
- `type = UNLISTED` means that the name of the service does not have to be in `/etc/services`, but you have to specify port and protocol. If you want to do use an existing service name, e.g. `http`:

```
service http
{
  disable = no
  socket_type = stream
  user = root
  wait = no
  redirect = 127.0.0.1 8888
  log_type = FILE /tmp/somefile.log
}
```

Have a browse in `/etc/services` and it will all become clear.

- Logging may present certain security problems, you might want to leave that out.
- RHEL5 for some reason doesn't contain `xinetd` by default for reasons best known to themselves. `yum install xinetd` will fix that.

Xinetd is a hugely powerful and configurable system so expect to do some reading.