Hosting Onion Services - riseup.net

13-17 minutes

How to use this guide.

Here you can find information about running Onion Services based on our experiences running them and helpful tips from people like you. If you have a helpful tip, or can translate this into another language, please contribute!

“Onion Services” were previously known as “Tor Hidden Services”, but have been renamed since “Hidden Service” didn’t accurately describe what was possibile. This guide uses the new name.

Installing and configuring Onion Services

For information on configuring onion services, please read the Tor Project’s guide

Make sure your Tor software is updated

It is not enough to simply install Tor and configure your onion service and then forget about it. You must keep it up to date so that critical security flaws are fixed. All software has bugs, and Tor is no exception. Make sure you are keeping your software up-to-date.

Many things can be made into onion services

You can do a lot of things over onion services, not just make a website available! You can also provide IMAP, or SMTP, or deliver mail between MTAs, among many other possibilities. Spread the onions far and wide! But be careful, if the service makes DNS request for whatever reason (like resolving where that SMTP server is to send the email), then you leak information. One way to work around this is to have the machine running your service fully iptabled to go through Tor all the time.

Don’t run a relay at the same time

Do not run a relay and an onion service on the same instance. Having a relay and an onion service on same IP and/or machine helps traffic correlation and fingerprinting. However, Tor is smart enough to **not** choose itself as a node for the circuit so it's not a disaster but ideally you want to avoid it.

Monitor your onion service(s) availability

Although their stability has improved greatly, onion services can still fail for a number of reasons. Set up some monitoring to regularly connect to your onion service(s) to make sure that they are still functioning.

Multiple ports for one onion service

You don't need to create a different onion service for every service you want to make available, just add more HiddenServicePort lines, for example:

```
HiddenServiceDir /usr/local/etc/tor/other_hidden_service/
HiddenServicePort 6667 127.0.0.1:6667
HiddenServicePort 22 127.0.0.1:22
```

If you want to run multiple onion services from the same Tor client, just add another HiddenServiceDir line to the config file.

SSL/TLS isn't necessary

You don't really need SSL/TLS in an onion address (ie. https) since it's a complete encrypted tunnel + PFS (perfect forward secrecy), **but** it does not hurt having extra layers in that onion!

Although it is true that extra layers are good beware that usually redirecting to SSL/TLS will mean that the certificate will not validate (because the hostname will be *.onion, instead of the certificate that you have for your public service). If you can get a .onion certificate, that works!

If your onion service does use TLS, make sure that it does not send a certificate for an external website.

Onion services and Rails 4

In order to get a .onion site to play nice with rails, and have the site also work over HTTPS when not using the .onion, you need change a few defaults.

The first thing that must be changed is to not use the `config.force_ssl = true` option. This option is the default for rails apps in production. This setting forces secure cookies and forces HSTS. Change `my_rails_app/config/environments/production.rb` to be:

```
config.force_ssl = false
```

Once we set `force_ssl = false`, we want to add back the ability to enforce secure cookies and HSTS when using normal HTTPS. So, to do this, we make sure the web server is setting the HSTS headers for the HTTPS virtualhost, and we add the secureheaders gem to enforce secure cookies. The `secureheaders` gem will set the `Secure` cookie flag only for HTTPS connections, unlike the rails `force_ssl` flag. This allows use to have secure cookies for the regular HTTPS site and insecure cookies for the .onion site, which is what we want.

Install the secureheaders gem for your application, in `my_rails_app/Gemfile`:

```
gem 'secure_headers', '~> 3.5'
```

(replace 3.5 with whatever the current version of secureheaders is available)

Add a secureheaders configuration, in `config/initializers/secureheaders.rb`:

```
SecureHeaders::Configuration.default do |config|
  config.cookies = {
    secure: true,
    httponly: true,
    samesite: {
      strict: true
    }
  }
end
```

NOTE: When configuring apache or nginx in this setup, do not set the X_FORWARDED_PROTO environment variable to be https **on the port 80 onion virtual host**. You **should** set it on the port :443 non-onion virtual hosts.

Onion services can be found

If you are not very careful and keep your server from revealing identifying information about you, your computer, or your location, then the onion service will no longer be hidden!

Leaking the real server

A common misstep here is server signatures, for example it is easy to determine if a webserver is thttpd or Apache, or learn about your operating system because the banner tells the version of the running service and operating system.

Another way that your onion address will get out is via the referrer header in browsers when a client browses a hidden service website and then clicks on a clearnet/hidden service link. The Tor browser has taken care of many of these tiny leaks, so be sure to encourage your users to use an up-to-date tor browser instead of using their own browser with Tor.

If the server running the onion service is also exposed to the clearnet, make sure that when you connect to either the clearnet service or the onion service, you cannot specify in the host header the other service and get a response. You should ensure the onion service is only listening on the internal IP and your external service is only listening on the external IP address. The easiest way to ensure there are no failures here is this is to run your service on a machine that has no external IP address.

Make sure the time on your server is correct, and is corrected automatically by NTP, so that time skews do not help identify your server.

Make sure you are not inadvertently exposing information, for example with PHP you may disclose the server's real name/address if you leak phpinfo() or $_SERVER, or expose error messages!

Look into protecting yourself against Server Side Request Forgery (SSRF). This attack works by getting the server to perform an external connection (DNS lookup, etc.) which can expose your machine's real location. Strict egress firewalling is one way to mitigate against this problem.

The longer an onion service is online, the higher the risk that its location is discovered. The most prominent attacks are building a profile of the onion service's availability and matching induced traffic patterns.

There are currently ways in the protocol that a bad relay can learn about your onion address, even if you don't tell anybody. Follow the discussion on the subject if you want to stay on top of how the Tor project is working on fixing these issues.

OnionScan

Use the [OnionScan→onionscan.org] tool to scan HTTP onion services to look for leaks. It will look for IP

addresses, EXIF metadata in images, and things like enabled mod_status that can leak the real IP address of the server.

Onion services don't need to be hidden!

You can provide a onion service for a service that you offer publically on a server that is not intended to be hidden. Onion services are useful to protect users from passive network surveillance, they keep the snoopers from knowing where users are connecting from and to.

Make your onion services easy to find

If you provide onion services, make them known to your users by advertising their existance, their onion hostnames and ports that they provide in a way that authenticates they are the ones that are legitimate (for example, you could digitially sign the list of onion addresses like Riseup does, or put them in DNS txt records).

Ask your favorite online service to provide an onion service!

Advocate for more onion services by asking those who provide the services that you use to make them available. They are easy to setup and maintain, and there is no reason not to provide them!

Moving onion services

You can move onion services between systems, just copy the `/var/lib/tor/<hidden_service>` directory to the new system and make sure the `torrc` on the new system has the same configuration as the old one. Be sure to disable and stop the old one before starting the new one. The onion service directory simply contains the hostname of the onion service, and the private key.

Protecting your services

Protect your private keys

Keep the onion service private key private! That key should not be available to the public, it should not be shared and it should have proper permissions set so it is not readable by anyone on your system, except for the Tor process.

Backup your private keys

If you plan to keep your service available for a long time, you might want to make a backup copy of the private_key file somewhere safe.

Be careful of localhost bypasses!

You should take very careful care to not accidentally expose things on your server that are restricted to the local machine. For example, if you provide /server-status in apache (from mod_status which is enabled per default in debian's apache) to monitor the health of your apache webserver, that will typically be restricted to only allow access from 127.0.0.1, or you may have .htaccess rules that only allow localhost, etc.

There are a few ways you can solve this problem:

- **different machine**: consider running the onion service on a different machine (real or virtual) than the actual service. This has the advantage that you can isolate the service from the onion service (a compromise of one doesn't compromise the other) and helps with isolating potential information leaks
- **isolation**: similarly to the above, you can also isolate Tor and the service so it will run on a different network namespace than the service. Tails uses a Onion Service configuration-or-fail packet filter.
- **public ip**: configure the onion service to connect to the public IP address of the service instead of localhost/127.0.0.1, this should make Tor not pick 127.0.0.1 as the source address and avoid most misconfigurations. For example like this:

```
HiddenServiceDir /var/lib/tor/hidden/ftp/
HiddenServicePort 80 192.168.1.1:81
```

Note: This makes your server and vhost potentially reachable to an external entity. There has been a growing number of attempts to discover the true location of sites behind cloudflare that are badly configured because they still expose their true httpd on a public IP address. People regularly use masscan and zmap to scan the entire ipv4 address space and try to connect to a publicly exposed httpd and request “high-value” onion addresses from the httpd to see if they send a Host header and make the site serve their probed vhosts content.

Binding to a port that is different from the “true” port is a source of a potential leak on Apache. If there is a directory, e.g. `foo.onion/css/` then a request to `foo.onion/css` will cause apache to emit a 301 redirect, but when it does issue it, it will include the port that it thinks the service is listening on. Instead of sending a 301 to `foo.onion/css/` it would send a 301 for `foo.onion:81/css/` this both breaks the website and reveals the port the httpd is really running on.

- **unix socket**: consider using unix socket support instead of a TCP socket (requires 0.26 or later Tor) – if you do this, then the onion service will be running on the same server as the service itself. With a socket approach, you should be able to run with privatenetwork=yes in systemd unit which gets you some really great isolation, for example:

```
HiddenServicePort 80 unix:/etc/lighttpd/unix.sock
```

But then the service itself needs to support unix sockets, otherwise you have to setup some socat redirection from tcp <→ unix (nginx, twisted, lighttpd all support this).

audit carefully: carefully audit, and regularly re-audit your system for configurations that allow localhost/127.0.0.1, but prohibit everywhere else and configure those to work around the problem (for example make /server-status operate on a different IP; make the webserver listen on a different port for /server-status; make it password protected, etc.).

You can make onion services require authentication to use.

If you set HiddenServiceAuthorizeClient (see man page), then it is only available for authorized clients. This will mean that you can’t even attack the service unless you break Tor (or have the authorization key).

Protect your onion services from advanced attacks

If you run a high-security onion service which is under attack by sophisticated adversaries, you should install the Vanguards addon which defends against various advanced attacks. Please read Tor project’s blog post on how to install and use this tool.

In addition, you can also enable Sandbox 1 in your torrc to enable the built in sandboxing.