How to offer your existing Web Site as an I2P eepSite

A somewhat meta-tutorial and guided tour of mirroring a clear-web site to I2P. Unfortunately, it's probably impossible to *completely* cover all possible cases of making an existing web site available as an eepSite, there's simply too diverse an array of server-side software, not to mention the in-practice peculiarities of any particular deployment of software. Instead, I'm going to try and convey, as specifically as possible, the general process preparing a service for deployment to the eepWeb or other hidden services.

Much of this guide will be treating the reader as a conversational participant, in particular If I really mean it I will address the reader directly(i.e. using "you" instead of "one") and I'll frequently head sections with questions I think the reader might be asking. This is, after all, a "process" that an administrator must consider themselves "involved" in just like hosting any other service.

DISCLAIMERS:

While it would be wonderful, it's probably impossible for me to put specific instructions for every single kind of software that one might use to host web sites. As such, this tutorial requires some assumptions on the part of the writer and some critical thinking and common sense on the part of the reader. To be clear, **I have assumed that the person following this tutorial is already operating a clear-web service linkable to a real identity or organization** and thus is simply offering anonymous access and not anonymizing themselves.

Thus, **it makes no attempt whatsoever to anonymize** a connection from one server to another. If you want to run a new, un-linkable hidden service that makes server-to-server connections, additional steps will be required and will be covered in another tutorial.

That said: If you can be sure that a *brand new service* which is *not available to the clear-web* will never make a server-to-server connection and will not leak server metadata in responses to clients, then services configured in this way will be anonymous.

Process One: Prepare your Server

Step one: Determine what software you are running

In practice, your web service probably uses a number of things to enhance it's reliability and security. These things could be proxies, reverse proxies, containers, tunnels, Intrusion Detection Systems, rate-limiters, load balancers, among many other things. When you get started, you should go through your deployment and determine which software you are using, and what you are using it for.

As you examine your software, ask yourself these questions

These questions should help you evaluate what parts of your software stack are relevant to your I2P eepSite.

Does this software work based on IP addresses?

If you are using software which alters the behavior of traffic based on the IP address

of the sender, these things will probably not work with I2P, or may work in complicated or unexpected ways. This is because the address will usually be the localhost, or at least the host where your I2P router is running. Software which sometimes does things based on IP addresses could be Fail2Ban, iptables, and similar applications.

Does this software work by "Tagging" traffic with additional metadata?

Some software may be configured to add information to the traffic it handles. Obviously, if this information is identifying it should not be part of the chain of services that is exposed to the I2P network.

Does this software work by communicating with a remote resource? What triggers this behavior?

Some software may also draw from remote resources, to find up-to-date rules and block lists which can be used to prevent attacks. Some of these might be useful as part of the service that is exposed to I2P, but you should make sure that the rules are applicable and that a rules update cannot be triggered as a result of a normal client request. This would create a server-to-server communication which could reveal the timing of an I2P communication to a third party.

Step two: Determine which port to Forward to I2P and Optionally locate your TLS certificate

Now you've gathered all the information that you will require to forward your service to I2P. Once you've selected the point at which you would like to make your site available to I2P, you will need to note the port you wish to foward. In simple scenarios, this will probably just be port 80 or port 8080. In more sophisticated scenarios, this might be a reverse proxy or something like that. Make a note of the port.

Establishing a Common Identity for both the Clearnet and your eepSite

Should you be a non-anonymous organization that wishes to provide enhanced privacy to your users by providing a hidden service, you may wish to establish a common identity between versions of your site. However, since we can't add .i2p domains to clearnet TLS certificates, we have to do this in another way. To do this, **even if you are forwarding the HTTP port and not HTTPS**, make a note of the location of your TLS certificate for use in the final step.

Process Two: Forward your service to an eepSite

Congratulations! You've completed the most difficult part. From here on, the decisions you must make, and the consequences that they will have, are much more straightforward and easy to enumerate. Such is the beauty of a cryptographically secure network layer like I2P!

Step three: Generate your .i2p Tunnels and Addresses

For eepSites, you will need to create an HTTP Server Tunnel. This is an I2P destination with a few special features for hosting HTTP services to enable things like rate-limiting, filtering, and the inclusion of headers to identify the destination of the client to the server. These enable flexibility in how you handle connections in terms of load-balancing and rate-limiting on a case-by-case basis, among other things. Explore these options and how they relate to the applications which you considered in step one, even though a very simple setup is easy, larger sites may benefit from taking advantage of these features.

Create an HTTP Tunnel for your application

If you've configured a reverse proxy or an SSH tunnel before, then the general idea here should be very familiar to you. I2PTunnel, in essence, is just forwarding ports from the host to the I2P Network. To set this up using the web interface, go to the I2PTunnel configuration page.

At the bottom of the "I2P Hidden Services" section of the page, select an HTTP Service from the drop-down and click "Create."

It will immediately drop you into the granular tunnel configuration page, which we're about to explore from top-to-bottom. The first, most essential settings are the tunnel name and the target host:port. **The target host:port is the place where you input the address of the service you are forwarding to I2P**. Once you've configured that, your web site will become available over i2p. However, there are probably a few things that we can improve.

Next, you may want to pick a hostname to use for your eepSite. This hostname doesn't need to be universally unique, for now, it will only be used locally. We'll publish it to an address helper later. **If** the *Local Destination* field isn't populated with your Base64 Destination yet, you should scroll down to the bottom, save the tunnel configuration, and return to the tunnel configuration.

A little further down the configuration page, the tunnel options are available. Since you've got a site which is not intended to be anonymous, but rather to provide anonymous access to others by an alternate gateway, it may be good to reduce the number of hops the tunnel takes on the I2P network.

Next are the encrypted leaseset options. You can probably leave these as the defaults, since your site isn't anonymous it probably doesn't need features like blinding or encrypted leasesets. If you were to choose encrypted leasesets, you would not be accessible to anyone unless you shared a key with them in advance.

The next few parts may be especially useful to you if you run a high-traffic site or find yourself subject to DDOS attacks. Here you can configure various kinds of connection limits.

After that, there are a few other ways of filtering connections by client characteristics. First, you can block access via inproxies like I2P.to and similar. Since you have a clearnet presence already, changing this may be better if you want to encourage I2P users to only use your eepSite. You can also block accesses via specific user-agents, for instance blocking wget may be helpful if you want to prevent spidering. Finally, and of particular interest to Fail2Ban users, the "Unique local address per client" will give each client it's own local IP address instead of them all appearing to the server to be from 127.0.0.1.

You can probably leave these next few options to the defaults.

Lastly, you can set up an advanced filter definition. Writing filters is beyond what I'm prepared to do in this document, for more information see the format specification for now.

Multi-Home an Application

One interesting thing that I2P can do is host the same site on multiple servers at the same time transparently, which is referred to as "Multihoming." In order to multihome your application, you will need to return to the tunnel menu and change the location of your private key file to it's own, non-shared location.

When you're done, copy the new key file for your new multihomed service to a storage device. Now, you can re-produce your service/tunnel configuration with those same keys on any I2P router and increase your service's redundancy.

Step four: Publicize and Authenticate your eepSite

Since you're running an anonymously accessible instance of an existing clear-net service, you'll probably want to leverage some existing form of trust to distribute your eepSite URL, like a TLS Certificate signed by a recognized and reputable authority. What can I say we live in an imperfect world.

Place your .b32.i2p link on your clearnet page

The simplest way to provide a link to your eepSite using an existing site to distribute the link is to distribute a so-called "Base32" address on your existing web site. The Base32 address is the hash of the public key of your I2P destination, so it cannot be forged if it is provided by a reliable source. In the case of a clear-net site with a hidden service presence, one of those places is likely to be a web page.

Distributing an "Addresshelper" link from your clearnet page

Create a subscribable address feed

I2P Subscription Feed Commands

Distributing

See Also:

Most of the security issues of hosting Tor hidden services also apply to I2P. It would be advisable to take advantage of their resources as well as this one:

Misc Links

- Official Guide
- Riseup best Practices

- Blog about config fails
- Whonix Docs Onion Service
- Reddit thread

Stack Exchange

- Hosting clearnet site as onion service
- Securing a Tor Hidden Service
- Effects of hosting hidden and non-hidden services

Clearnet Web Sites announcing Public Services:

- Propublica
- Wikipedia Proposal

Important Questions from Systems Administrators

Got a new question? File an issue or visit us on IRC2P, and I will add the answer here!

Why would I want to mirror my Clear-Web site on.i2p?

I still get this question alot, and it kind of surprises me at this point. After all, many major clear-web services already have the very similar Tor Hidden Services, and Tor even added a feature called "Single-hop Onions" to help non-anonymous services be more effiecient and accessible to anonymous clients on their overlay network. This is so that you can add value to your service by allowing clients to access it anonymously. Besides that, there are also several services on.i2p which are accessible from clear-web counterparts, services such as Jabber, e-mail, and activitypub-based social-networking.

So the point of the DuckDuckGo hidden service, for example, isn't to anonmymize DuckDuckGo, that's obviously not possible. It is to provice anonymous, blocking resistant access to *visitors* who do not wish to reveal their possible coarse physical location in the form of their IP address. Moreover, it is actually very easy to allow this kind of access to your existing web service.

Why wouldn't I just let.i2p Users rely on their OutProxy?

Just in case you didn't know,.i2p has fairly few outproxies, all run by volunteers. We really appreciate the work of our outproxy operators, and they have a job that can get pretty complicated providing generic access to the clearweb for anonymous clients. As a web site operator, you are in a genuinely unique position to make this job easier. Only *you* can safely provide an eepSite for your existing clear-web service to accept anonymous users and only *you* can reduce the burden on the outproxy operators.

How can I ensure that my users are seeing the real eepSite for my Service?

This is the reason *your* participation in the.i2p network is so essential to providing anonymous access to your eepSite. After all, anyone can set up a fake gateway to say, reddit.com:80(Because somehow Reddit still lets you use a plain HTTP connection in 2019, fortunately, most browsers at this point won't let you use it in practice), as a Tor hidden service or.i2p eepSite and unless your users take the time to verify the .onion or .b32.i2p address manually, they seriously risk typing their password into a malicious site.

The difference *you* and **only you** can make is to use your TLS certificate to ensure that the hidden service that the user connects to is the same site, run by the same people, as the clear-web service, the behavior that the user not only expects but

absolutely counts on.

Do I really want to allow anonymous users first-class access to my site?

Honestly you don't have a choice between allowing, or not allowing, anonymous access to your site. If you block outproxies,.i2p users will simply use Tor. If you block Tor exits, Tor users will chain open HTTP Proxies. If all the open HTTP proxies disappeared overnight(they won't) then VPN's would still be a thing. And, to belabor a point already made by Tor, *botnet-based anonymizers used to conceal criminal activity* already exist in the wild and are more likely to be used maliciously than Tor, because that's what they're for! So the real choice is

1. Offer first-class access to anonymous users via a hidden service under your own control,
2. Encourage anonymous users to hide from you in escalatingly more anti-social ways.

And in light of that, I think the choice is pretty clear. If you want your anonymous users to be like your regular users, you should treat your anonymous users like regular users, after all, that's what they are. :)

How do I route server-to-server interactions over.i2p.

Your server may allow the user to trigger interaction with a remote outside of your control. For example, federated applications are able to make requests to other members of their federation, so in open federations it is especially important to ensure that all traffic outgoing from the server is routed to an anonymizing network. Other things to watch out for could be user-supplied avatars or globally-recognizable avatars, which are implemented in many popular pieces of software such as PHPbb, pingbacks from blogs, etc. In some cases, setting the *_PROXY environment variables in the context of the service is enough to route the traffic over the proxy. If this is the case with your server software, then it is also possible to use Privoxy to route server traffic to other.i2p eepSites, Tor Onion Services, and clearnet traffic.

Unfortunately, some software, in my experience much of it is in PHP, doesn't honor these environment variables. I don't have a straightforward solution to this yet, however, a Whonix-based setup or a SOCKSifier may be possible solutions.

I2P and TLS Certificates(not a step)

NOTE

It's perfectly possible to use a TLS certificate to identify a site inside of I2P in addition to.i2p's self-authenticating properties. However, the most useful application of this I can think of, using a TLS certificate to show that an eepSite and a clearnet site are served by the ame organization is currently impossible, or at least so difficult it would be counter-productive to instruct people to do it in this way. This section is therefore temporarily abandoned, to be revived if circumstances change for any reason.

My sincerest thanks to the helpful young lady at Digicert who made sure we exhausted all the options I didn't yet know about.

BEGIN

In order to ensure that your eepSite deployment gets the same benefits as your clear-web site as possible, it's usually a good idea to connect your service to I2P at the same point that you connect to the clear web, with some exceptions you might decide

on based on the questions above. If you are not a corporation offering .i2p access, you can probably skip this section entirely in favor of side-channel validation of the.i2p base32 and addresshelper links, which is arguably easier, much less expensive, and apolitical.

Important Reading:

- Tor Blog
- CA/Browser Forum Discusson
- Even more CA?Browser forum Siscussion
- Digicert announcement
- Let's Encrypt Discusson
- CA?Browser Forum Ballot/Spec
- Stack Exchange Question on Validation
- CertBot issue
- Blog entry
- Matt Traudt

If you want to use TLS for your site

If your site uses TLS, which it should on the clearnet unless it's literally Designed specifically to be hijacked at a McDonalds, then you may want to link your eepSite to your identity as authenticated by TLS. Unfortunately, this is easier said than done, I combed through every bit of documentation that I could find and there doesn't seem to be any way to add a .b32.i2p or .i2p domain name to a certificate *except* possibly for some Universal Communications Certificates, which I cannot afford. Unlike .onion, .i2p domain names are not recognized as special use domain names yet and cannot be included in the SAN field of other kinds of certificates.

So as a workaround to establish a trusted link between your clearnet presence and your.i2p presence, we're going to use a simple trick. Unfortunately, it won't fix all the usability issues that come with self-signed certificates, but it *will* allow visitors to confirm that a self-signed certificate was created by a person with access to your private key and establish a link to your corporate presence. We will generate this self-signed certificate in a future step, for now, identify the certificates and keys used to authenticate your clearnet site and make a note of their location.

Future work: TLS-in.i2p without self-signed certificates

In the future, it will probably be beneficial to authenticate .i2p domains in another way. darkweb-everywhere may be an interesting prospect.

Once you have determined whether or not it is beneficial to you to use TLS

Find the port that your services are using to communicate with the net, for example, with HTTP it will usually be 80 or 8080, with HTTPS it will usually be port 443. The exact answer will depend upon your set-up.