peer to peer network with tor

git-annex has recently gotten support for running as a Tor hidden service. This is a nice secure and easy to use way to connect repositories in different locations. No account on a central server is needed; it's peer-to-peer.

dependencies

To use this, you need to get Tor installed and running. See their website, or try a command like:

```
sudo apt-get install tor
```

You also need to install Magic Wormhole.

```
sudo apt-get install magic-wormhole
```

pairing two repositories

You have two git-annex repositories on different computers, and want to connect them together over Tor so they share their contents. Or, you and a friend want to connect your repositories together. Pairing is an easy way to accomplish this.

(The instructions below use the command line. If you or your friend would rather avoid using the command line, follow the webapp walktrough. It's fine for one person to use the command line and the other to use the webapp.)

In each git-annex repository, run these commands:

```
git annex enable-tor
git annex remotedaemon
```

The enable-tor command may prompt for the root password, since it configures Tor. Now git-annex is running as a Tor hidden service, but it will only talk to peers after pairing with them.

In both repositories, run this command:

```
git annex p2p --pair
```

This will print out a pairing code, like "11-incredible-tumeric", and prompt for you to enter the other repository's pairing code.

Once the pairing codes are exchanged, the two repositories will be securely connected to one-another via Tor. Each will have a git remote, with a name like "peer1", which connects to the other repository.

Then, you can run commands like `git annex sync peer1 --content` to sync with the paired repository.

Pairing connects just two repositories, but you can repeat the process to pair with as many other repositories as you like, in order to build up larger networks of repositories.

how to exchange pairing codes

When pairing with a friend's repository, you have to exchange pairing codes. How to do this securely?

The pairing codes can only be used once, so it's ok to exchange them in a way that someone else can access later. However, if someone can overhear your exchange of codes in real time, they could trick you into pairing with them.

Here are some suggestions for how to exchange the codes, with the most secure ways first:

- In person.
- In an encrypted message (gpg signed email, Off The Record (OTR) conversation, etc).
- By a voice phone call.

starting git-annex remotedaemon on boot

Notice the `git annex remotedaemon` being run in the above examples. That command runs the Tor hidden service so that other peers can connect to your repository over Tor.

So, you may want to arrange for the remotedaemon to be started on boot. You can do that with a simple cron job:

```
@reboot cd ~/myannexrepo && git annex remotedaemon
```

If you use the git-annex assistant, and have it auto-starting on boot, it will take care of starting the remotedaemon for you.

speed of large transfers

Tor prioritizes security over speed, and the Tor network only has so much bandwidth to go around. So, distributing large quantities (gigabytes) of data over Tor may be slow, and should probably be avoided.

One way to avoid sending much data over tor is to set up an encrypted special remote someplace. git-annex knows that Tor is rather expensive to use, so if a file is

available on a special remote as well as over Tor, it will download it from the special remote.

You can contribute to the Tor network by running a Tor relay or bridge.

onion addresses and authentication

You don't need to know about this, but it might be helpful to understand how it works.

git-annex's Tor support uses onion address as the address of a git remote. You can `git pull`, push, etc with those onion addresses:

```
git pull tor-annnex::eeaytkuhaupbarfi.onion:4412
git remote add peer1 tor-annnex::eeaytkuhaupbarfi.onion:4412
```

Onion addresses are semi-public. When you add a remote, they appear in your `.git/config` file. For security, there's a second level of authentication that git-annex uses to make sure that only people you want to can access your repository over Tor. That takes the form of a long string of numbers and letters, like "7f53c5b65b8957ef626fd461ceaae8056e3dbc459ae715e4".

The addresses generated by `git annex peer --gen-addresses` combine the onion address with the authentication data.

When you run `git annex peer --link`, it sets up a git remote using the onion address, and it stashes the authentication data away in a file in `.git/annex/creds/`

When you pair repositories, these addresses are exchanged using Magic Wormhole.

security

Tor hidden services can be quite secure. But this doesn't mean that using git-annex over Tor is automatically perfectly secure. Here are some things to consider:

- Anyone who learns the onion address address and authentication data of a peer can connect to that peer, download the whole history of the git repository, and any available annexed files. They can also upload new files to the peer, and even remove annexed files from the peer. So consider ways that the authentication data of a peer might be exposed.

- While Tor can be used to anonymize who you are, git defaults to including your name and email address in git commit messages. So if you want an anonymous git-annex repository, you'll need to configure git not to do that.

- Using Tor prevents listeners from decrypting your traffic. But, they'll probably still know you're using Tor. Also, by traffic analysis, they may be able to guess if you're using git-annex over tor, and even make guesses about the sizes and types of files that you're exchanging with peers.

- There have been past attacks on the Tor network that have exposed who was running Tor hidden services. https://blog.torproject.org/blog/tor-security-advisory-relay-early-traffic-confirmation-attack

- An attacker who can connect to the git-annex Tor hidden service, even without authenticating, can try to perform denial of service attacks.

- Magic wormhole is pretty secure, but the code phrase could be guessed (unlikely) or intercepted. An attacker gets just one chance to try to enter the correct code phrase, before pairing finishes. If the attacker successfully guesses/intercepts both code phrases, they can MITM the pairing process.

  If you don't want to use magic wormhole, you can instead manually generate addresses with `git annex p2p --gen-addresses` and send them over an authenticated, encrypted channel (such as OTR) to a friend to add with `git annex p2p --link`. This may be more secure, if you get it right.

Security of P2P repo is unclear

In the security section, you say that

> Anyone who learns the address of a peer can connect to that peer, download the whole history of the git repository, and any available annexed files. They can also upload new files to the peer, and even remove annexed files from the peer. So consider ways that the address of a peer might be exposed.

Do you mean the addresses from `git annex peer --gen-addresses` here? Say, if someone has only my onion service address, and none of the authentication data that is normally placed in `.git/annex/creds/`, what can they do with my git repository? I think I might be confused by the use of "address" because of onion addresses, which are not private.

*Comment by **dvicory** — 7 months and 7 days ago*

comment 2

@dvicory if someone only knows the onion service address, they can do nothing to your repository except connect to it and get rejected due to failure to authenticate. They need the authentication data too in order to do any of those things. That was talking about the addresses generated by `git annex peer --gen-addresses`, which include authentication data.

I've improved the wording to avoid confusion between git-annex's addresses and onion addresses.

*Comment by **joey** — 7 months and 5 days ago*

onion-grater

you may want to consider using onion-grater to limit possible escalations
in the use of that control port:

RFP: Debian bug #859125

*Comment by **anarcat** — 6 months and 7 days ago*

comment 4

@anarcat, I'm not sure I follow how git-annex would use that. It does not currently use the tor control port at all when setting up its tor hidden service. But please file a todo item if I'm wrong and there's something to be improved.

(What would be useful is if tor came with the control port enabled by default, and perhaps protected by something like onion-grater. Then git-annex could use it when it installs its hidden service, instead of its current approach of prompting for the root password and modifying the torrc file.)

*Comment by **joey** — 6 months and 1 day ago*

Add a comment

Last edited 7 months and 5 days ago