Anonymity Challenge: You can(and should) add anonymity-supporting features to your application

Note: It’s not personal

- In some places in this presentation I am going to use the phrase “We assume competence.”
- This is *my* problem, not yours.
- Anonymity does not need to be your core competence, that is a bad requirement that I am working against. You can and should be competent at developing your application.
- I should be able to provide you with useful, safe ways to help you protect user privacy.

What do we do now(For clients):

- all_proxy, ALL_PROXY, http_proxy, https_proxy(Inconsistent, bad)
- torsocks/uwt(LD_PRELOAD)(Increasingly bad)
- Network Namespaces(Bad)
- Point-to-point tunnels(Bad)
- Browser Profile Managers(Slightly less bad)
- Tor Browser(Better)
- Whonix/Qubes-Whonix(Hard work)

What do we do now(For Servers):

- (Mostly) Assume competence
- Manual, text-file based configuration(Tor)
- GUI based configuration(I2P Java only)
- Filtering with reverse proxies
- FAIL. Badly.

What do we do now(For federations):

- (Dangerously) Assume competence
- Manual, text-file based configuration
- GUI based configuration(I2P Java only)
- Filtering with reverse proxies
- Neglect to anonymize server-to-server communication

This is extremely error prone!

- Uncareful use of SOCKS proxies leads to shared identity problems
- GNU wget has DNS leaks
- curl has approximately 76 gazillion proxy settings, *one* of which you should use with Tor
- curl and wget handle environment variables differently
- Chromiums allow proxy escapes
- Hardening Firefox is a Sisyphean task
- Whonix is a continuous, extensive audit of Debian applications

What should we be doing?

- Applications should support anonymity networks specifically, by name.
- Applications should handle connection setup, including identity management, automatically without requiring configuration by the user.
- Applications should be able to manage identity lifecycles alongside connection lifecycles.
- If requirements are not met, they should fail closed.

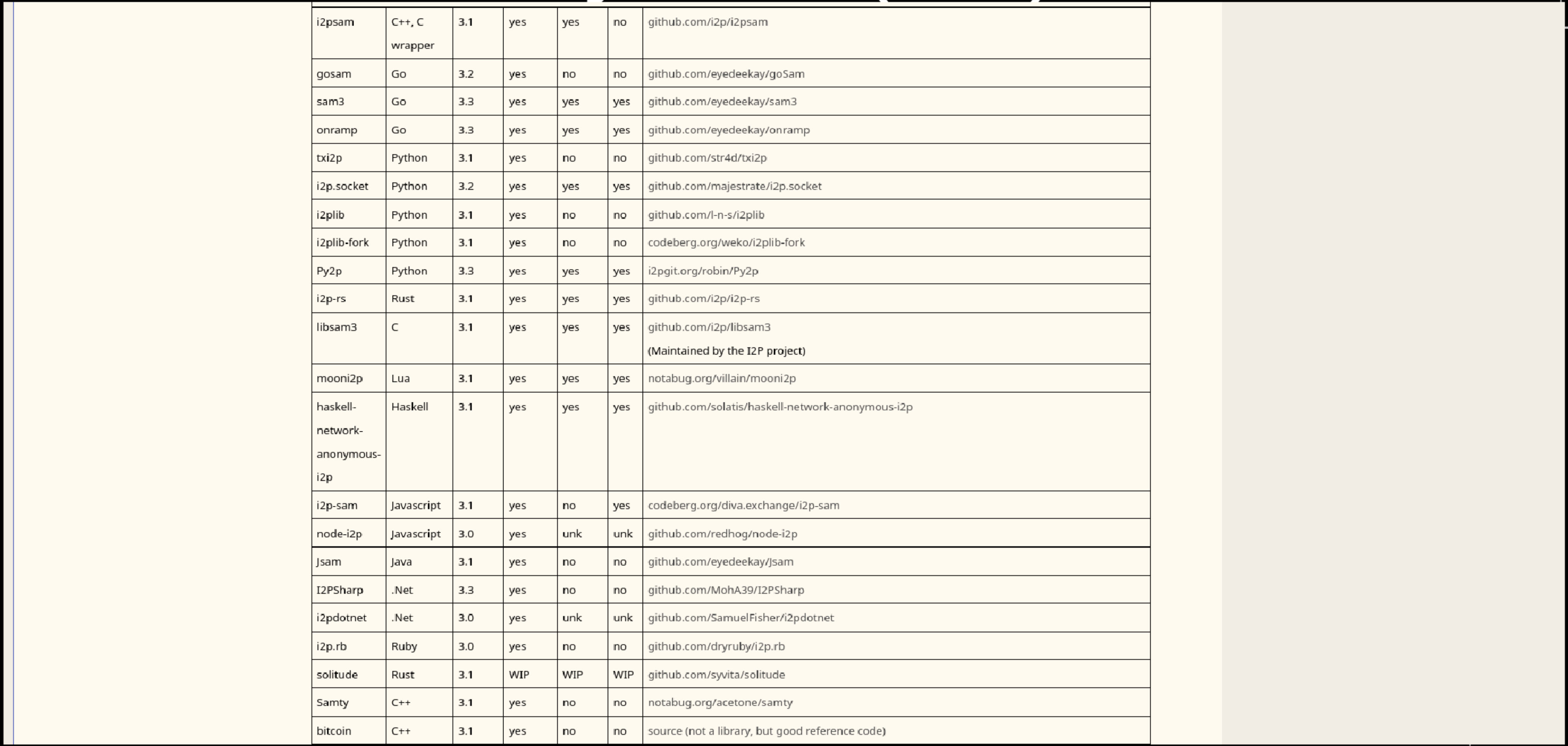
How?

- In I2P we have a protocol called SAMv3.3 API
  - A socket setup protocol consisting of a handshake phase, a configuration phase, and a socket phase
  - Supports streaming(TCP-Like) and 2 kinds of datagrams(datagrams with return addresses and datagrams without return addresses)
  - Identity management is automatic by default, all SAMv3 connections use distinct identities
- In Tor, you have to build upon their SOCKS API
  - If you’re using a Tor daemon on the system, you may need to read the `torrc`
  - OR you can start your own and configure it how you like
  - Identity management depends on configuration but usually involves passing information in the SOCKS authentication fields
  - It is possible to build your library so that identity management is automatic if you’re methodical
  - Arti has a chance to make this better

What’s good(I2P)?

- There are dozens of implementations of SAMv3.3 libraries, I maintain these ones:
  - `https://github.com/go-i2p/onramp` - Go
  - `https://github.com/go-i2p/sam3` - Go
  - `https://github.com/go-i2p/gosam` - Go
  - `https://github.com/i2p/i2psam` - C++, C
  - `https://github.com/i2p/libsam3` - C

There are many more(I2P):



i2psam	C++, C wrapper	3.1	yes	yes	no	github.com/i2p/i2psam
gosam	Go	3.2	yes	no	no	github.com/eyedeekay/goSam
sam3	Go	3.3	yes	yes	yes	github.com/eyedeekay/sam3
onramp	Go	3.3	yes	yes	yes	github.com/eyedeekay/onramp
txi2p	Python	3.1	yes	no	no	github.com/str4d/txi2p
i2p.socket	Python	3.2	yes	yes	yes	github.com/majestrate/i2p.socket
i2plib	Python	3.1	yes	no	no	github.com/l-n-s/i2plib
i2plib-fork	Python	3.1	yes	no	no	codeberg.org/weko/i2plib-fork
Py2p	Python	3.3	yes	yes	yes	i2pgit.org/robin/Py2p
i2p-rs	Rust	3.1	yes	yes	yes	github.com/i2p/i2p-rs
libsam3	C	3.1	yes	yes	yes	github.com/i2p/libsam3 (Maintained by the I2P project)
mooni2p	Lua	3.1	yes	yes	yes	notabug.org/villain/mooni2p
haskell-network-anonymous-i2p	Haskell	3.1	yes	yes	yes	github.com/solatis/haskell-network-anonymous-i2p
i2p-sam	Javascript	3.1	yes	no	yes	codeberg.org/diva.exchange/i2p-sam
node-i2p	Javascript	3.0	yes	unk	unk	github.com/redhog/node-i2p
Jsam	Java	3.1	yes	no	no	github.com/eyedeekay/Jsam
I2PSharp	.Net	3.3	yes	no	no	github.com/MohA39/I2PSharp
i2pdotnet	.Net	3.0	yes	unk	unk	github.com/SamuelFisher/i2pdotnet
i2p.rb	Ruby	3.0	yes	no	no	github.com/dryruby/i2p.rb
solitude	Rust	3.1	WIP	WIP	WIP	github.com/syvita/solitude
Samty	C++	3.1	yes	no	no	notabug.org/acetone/samty
bitcoin	C++	3.1	yes	no	no	source (not a library, but good reference code)

What about Tor?

- Confession: I’ve only ever used:
  - `https://github.com/cretz/bine` for automating interaction with Tor
  - Supports embedding ctor in Go
  - Handles identity automatically as it should

But there are others

- `https://github.com/dunglas/php-torcontrol` - PHP
- `https://www.torproject.org/getinvolved/volunteer.html.en#project-stem` - Python
- `https://github.com/dryruby/tor.rb` - Ruby
- `https://txtorcon.readthedocs.io/` - Also python

Does anyone know of any more?

- ACK: https://github.com/ajvb/awesome-tor?tab=readme-ov-file#development-and-research-tools

The SAMv3.3 Handshake

- Line-terminated socket-based protocol
- You can experiment with it using telnet

1) HELLO VERSION MIN=3.1 MAX=3.1
2) SESSION CREATE STYLE=STREAM ID=... DESTINATION=... i2cp.leaseSetEncType=4,0
3) STREAM ACCEPT ID=... (For services)
4) STREAM CONNECT ID=... DESTINATION=... (For clients)

If your app is a service:

- REMEMBER TO WRITE DOWN YOUR LONG-TERM KEYS

What do library developers do?

- In most languages, network sockets are an abstract type
- For example: `net.Conn` in Go, `net.Socket` in node.js and typescript, and `Socket` in Java
- Your goal, as a library developer, should be to implement these types and their requirements

Why? What is the effect?

- Your application will be able to automatically manage identity lifecycles by tying them to socket lifecycles
- Your application will not know how to connect to the outside world without the overlay network
- Your application will never need to know your own IP address
- User-error? *eliminated*. If something goes wrong, it’s the library developers fault.

What about Tor(Redux)

- Tor is a little tricker because it has multiple SOCKS-based isolation strategies and no dedicated APIs
- BUT it can be done, with a little study.
  - IsolateClientAddr is the default, and is mostly good but inflexible
  - IsolateClientProtocol, IsolateClientAddr, IsolateDestHost, IsolateDestPort are interesting but not useful for us
  - IsolateSocksAuth is the one we want!

Why do we need IsolateSocksAuth?

- We are taking control of our pseudonym, it could be short, medium, or long-term
- If we want our return-address(onion) to remain the same for our application, we need to pass in the X-Tor-Stream-Isolation header for our specific application

What about services?

- In Tor, to set up a hidden service, you need to either edit files on the disk or pass `ADD_ONION` to the control port
- If you don’t have access to the area of the disk where the Tor hidden services are configured, start your own ctor
- If you don’t have access to the control port and cannot edit the torrc file, start your own ctor
- IF YOU HAVE TO MANAGE YOUR OWN ctor, ALSO MANAGE IT’S LIFECYCLE!
- Running multiple ctor is comparatively cheap, but leaving them running is rude and irresponsible.

If your app is a service:

- REMEMBER TO WRITE DOWN YOUR LONG-TERM KEYS

OK so now we have libraries

- What does the rest of the work look like?

Good news! It’s basically grep/sed commands

- First, use `grep` to look for all the sockets your application sets up:
- `find . -name ‘*.go’ -exec grep –color=always -Hn net.Conn`
- Examine how the sockets are set up and fulfilled
- Replace vanilla sockets with your new anonymous sockets

Once you have an anonymous socket library, making an application use it is almost always formulaic

- Most applications can be ported in 24 hours or less
- Federated applications can use the same OR different identities for C2S and S2S communication
- Normal proxy escapes become impossible
- We’ll put your application on our website!

Caveats

- Some applications are inordinately complex, porting these is probably not going to solve all their problems
- Some developers start with concrete types instead of abstract types. If this is you, change. If this is somebody you know, convince them to change.

Thanks for coming!

- For a copy of this presentation, go to:
- https://github.com/eyedeekay/38C3