txsocksx Documentation

Release 1.15.0.2

Aaron Gallagher

August 03, 2015

Contents

1 Examples **3**
1.1 Authenticating 3
1.2 Connecting to a thing over tor 3
1.3 Cancelling a connection 4
1.4 Making HTTP requests 4
1.5 Upgrading to TLS 5
1.6 Proxying over a proxy 5

2 API **7**
2.1 `txsocksx.client` 7
2.2 `txsocksx.http` 8
2.3 `txsocksx.tls` 9

Python Module Index **11**

`txsocksx` is SOCKS4/4a and SOCKS5 client endpoints for Twisted 10.1 or greater. The code is available on github: https://github.com/habnabit/txsocksx

CHAPTER 1

Examples

These examples assume familiarity with how to use Twisted endpoints. For simplicity, most of the examples will use SOCKS5.

1.1 Authenticating

One specifies authentication methods to a *SOCKS5ClientEndpoint* via the *methods* parameter. For example, to connect using the username `spam` and password `eggs`:

```
exampleEndpoint = SOCKS5ClientEndpoint(
    'example.com', 6667, proxyEndpoint, methods={'login': ('spam', 'eggs')})
```

However, this will disable anonymous authentication. To use either login or anonymous authentication, specify both methods:

```
exampleEndpoint = SOCKS5ClientEndpoint(
    'example.com', 6667, proxyEndpoint, methods={'login': ('spam', 'eggs'),
                                                 'anonymous': ()})
```

The `methods` dict must always map from a string to a tuple.

1.1.1 SOCKS4

SOCKS4 has no authentication, but does have a configurable "user ID" which defaults to an empty string:

```
exampleEndpoint = SOCKS4ClientEndpoint(
    'example.com', 6667, proxyEndpoint, user='spam')
```

1.2 Connecting to a thing over tor

To connect to `example.com` on port 6667 over tor, one creates a *SOCKS5ClientEndpoint* wrapping the endpoint of the tor server:

```
torServerEndpoint = TCP4ClientEndpoint(reactor, '127.0.0.1', 9050)
exampleEndpoint = SOCKS5ClientEndpoint('example.com', 6667, torServerEndpoint)
```

Establishing the connection from there proceeds like usual:

```
deferred = exampleEndpoint.connect(someFactory)
```

`txsocksx` will not do any DNS resolution, so the hostname `example.com` will not leak; tor will receive the hostname directly and do the DNS lookup itself.

Tor allows connections by SOCKS4 or SOCKS5, and does not expect a user ID to be sent when using the SOCKS4 client.

1.3 Cancelling a connection

Sometimes one tires of waiting and wants to abort the connection attempt. For example, to abort the whole connection attempt after ten seconds:

```
torServerEndpoint = TCP4ClientEndpoint(reactor, '127.0.0.1', 9050)
exampleEndpoint = SOCKS5ClientEndpoint('example.com', 6667, torServerEndpoint)
deferred = exampleEndpoint.connect(someFactory)
reactor.callLater(10, deferred.cancel)
```

This is a trivial example; real code should cancel the IDelayedCall returned by `reactor.callLater` when the deferred fires. The code would then look like this:

```
torServerEndpoint = TCP4ClientEndpoint(reactor, '127.0.0.1', 9050)
exampleEndpoint = SOCKS5ClientEndpoint('example.com', 6667, torServerEndpoint)
deferred = exampleEndpoint.connect(someFactory)
canceler = reactor.callLater(10, deferred.cancel)

def cancelCanceler(result):
    if canceler.active():
        canceler.cancel()
    return result
deferred.addBoth(cancelCanceler)
```

1.4 Making HTTP requests

Twisted's builtin Agent HTTP client did not support being handed an arbitrary endpoint before 15.0, so `txsocksx` provides an `Agent` for maximum compatibility.

While `txsocksx` requires only Twisted 10.1, *`txsocksx.http`* requires Twisted 12.1 or greater. Its usage is almost identical to normal `Agent` usage:

```
torServerEndpoint = TCP4ClientEndpoint(reactor, '127.0.0.1', 9050)
agent = SOCKS5Agent(reactor, proxyEndpoint=torServerEndpoint)
deferred = agent.request('GET', 'http://example.com/')
```

Note that the `proxyEndpoint` parameter *must* be passed as a keyword argument. There is a second, optional, keyword-only argument for passing additional arguments to the *`SOCKS5ClientEndpoint`* as *`SOCKS5Agent`* constructs it:

```
torServerEndpoint = TCP4ClientEndpoint(reactor, '127.0.0.1', 9050)
agent = SOCKS5Agent(reactor, proxyEndpoint=torServerEndpoint,
                    endpointArgs=dict(methods={'login': ('spam', 'eggs')}))
deferred = agent.request('GET', 'http://example.com/')
```

`SOCKS5Agent` transparently supports HTTPS via *`TLSWrapClientEndpoint`*.

For users with Twisted 15.0 or greater, *`SOCKS5Agent`* also implements IAgentEndpointFactory.

1.5 Upgrading to TLS

Sometimes one wants to switch to speaking TLS as soon as the proxy negotiation is finished. For that, there is *txsocksx.tls*. After wrapping an endpoint with *TLSWrapClientEndpoint*, the connection will be upgraded to using TLS immediately after proxy negotiation finishes:

```
torServerEndpoint = TCP4ClientEndpoint(reactor, '127.0.0.1', 9050)
exampleEndpoint = SOCKS5ClientEndpoint('example.com', 6667, torServerEndpoint)
tlsEndpoint = TLSWrapClientEndpoint(exampleEndpoint)
deferred = tlsEndpoint.connect(someFactory)
```

1.6 Proxying over a proxy

Because of `txsocksx`'s composable design, it's trivial to connect from one SOCKS proxy to another:

```
torServerEndpoint = TCP4ClientEndpoint(reactor, '127.0.0.1', 9050)
firstProxyEndpoint = SOCKS5ClientEndpoint(
    'first-proxy.example.com', 1080, torServerEndpoint)
secondProxyEndpoint = SOCKS4ClientEndpoint(
    'second-proxy.example.com', 1080, firstProxyEndpoint)
finalHop = SOCKS5ClientEndpoint(
    'example.com', 113, secondProxyEndpoint)
deferred = finalHop.connect(someFactory)
```

CHAPTER 2

API

2.1 `txsocksx.client`

SOCKS4/4a and SOCKS5 client endpoints.

class `txsocksx.client.`**`SOCKS4ClientEndpoint`** (*host*, *port*, *proxyEndpoint*, *user=''*)

An endpoint which does SOCKS4 or SOCKS4a negotiation.

Parameters

- **`host`** – The hostname or IP to connect to through the SOCKS4 server. If this is a valid IPv4 address, it will be sent to the server as a SOCKS4 request. Otherwise, *host* will be sent as a hostname in a SOCKS4a request. In the SOCKS4a case, the hostname will not be resolved by `txsocksx` but will be sent without modification to the SOCKS4 server to be resolved remotely.
- **`port`** – The port to connect to through the SOCKS4 server.
- **`proxyEndpoint`** – The endpoint of the SOCKS4 server. This must provide IStreamClientEndpoint.
- **`user`** – The user ID to send to the SOCKS4 server.

`connect` (*fac*)

Connect over SOCKS4.

The provided factory will have its `buildProtocol` method once a SOCKS4 connection has been successfully negotiated. Returns a `Deferred` which will fire with the resulting `Protocol` when negotiation finishes, or errback for a variety of reasons. For example:

1. If the `Deferred` returned by `proxyEndpoint.connect` errbacks (e.g. the connection to the SOCKS4 server was refused).
2. If the SOCKS4 server gave a non-success response.
3. If the SOCKS4 server did not reply with valid SOCKS4.
4. If the `Deferred` returned from `connect` was cancelled.

The returned `Deferred` is cancelable during negotiation: the connection will immediately close and the `Deferred` will errback with a `CancelledError`. The `Deferred` can be canceled before negotiation starts only if the `Deferred` returned by `proxyEndpoint.connect` is cancelable.

If the factory's `buildProtocol` returns `None`, the connection will immediately close.

class `txsocksx.client.`**`SOCKS5ClientEndpoint`**(*host*, *port*, *proxyEndpoint*, *methods={'anonymous': ()}*)

An endpoint which does SOCKS5 negotiation.

Parameters

- **`host`** – The hostname to connect to through the SOCKS5 server. This will not be resolved by `txsocksx` but will be sent without modification to the SOCKS5 server to be resolved remotely.
- **`port`** – The port to connect to through the SOCKS5 server.
- **`proxyEndpoint`** – The endpoint of the SOCKS5 server. This must provide IStreamClientEndpoint.
- **`methods`** – The authentication methods to try.

Authentication methods are specified as a dict mapping from method names to tuples. By default, the only method tried is anonymous authentication, so the default *methods* is `{'anonymous': ()}`.

The `anonymous` auth method must map to an empty tuple if provided.

The other method available by default is `login`. `login` must map to a tuple of `(username, password)`.

`connect`(*fac*)

Connect over SOCKS5.

The provided factory will have its `buildProtocol` method once a SOCKS5 connection has been successfully negotiated. Returns a `Deferred` which will fire with the resulting `Protocol` when negotiation finishes, or errback for a variety of reasons. For example:

1. If the `Deferred` returned by `proxyEndpoint.connect` errbacks (e.g. the connection to the SOCKS5 server was refused).
2. If the SOCKS5 server gave a non-success response.
3. If the SOCKS5 server did not reply with valid SOCKS5.
4. If the `Deferred` returned from `connect` was cancelled.

The returned `Deferred` is cancelable during negotiation: the connection will immediately close and the `Deferred` will errback with a `CancelledError`. The `Deferred` can be canceled before negotiation starts only if the `Deferred` returned by `proxyEndpoint.connect` is cancelable.

If the factory's `buildProtocol` returns `None`, the connection will immediately close.

2.2 `txsocksx.http`

class `txsocksx.http.`**`SOCKS4Agent`**(**a*, *proxyEndpoint*, *endpointArgs={}*, ***kw*)

An Agent which connects over SOCKS4.

See *`SOCKS5Agent`* for details.

`endpointFactory`

alias of `SOCKS4ClientEndpoint`

class `txsocksx.http.`**`SOCKS5Agent`**(**a*, *proxyEndpoint*, *endpointArgs={}*, ***kw*)

An `Agent` which connects over SOCKS5.

Parameters

- **`proxyEndpoint`** – The same as *proxyEndpoint* for *`SOCKS5ClientEndpoint`*: the endpoint of the SOCKS5 proxy server. This argument must be passed as a keyword argument.
- **`endpointArgs`** – A dict of keyword arguments which will be passed when constructing the *`SOCKS5ClientEndpoint`*. For example, this could be `{'methods': {'anonymous': ()}}`.

The rest of the parameters, methods, and overall behavior is identical to Agent. The `connectTimeout` and `bindAddress` arguments will be ignored and should be specified when constructing the *proxyEndpoint*.

If used with Twisted 15.0 or greater, this class will also implement IAgentEndpointFactory.

`endpointFactory`
alias of `SOCKS5ClientEndpoint`

2.3 `txsocksx.tls`

TLS convenience wrappers for endpoints.

class `txsocksx.tls.`**`TLSWrapClientEndpoint`**(*contextFactory*, *wrappedEndpoint*)
An endpoint which automatically starts TLS.

Parameters

- **`contextFactory`** – A ContextFactory instance.
- **`wrappedEndpoint`** – The endpoint to wrap.

`connect`(*fac*)
Connect to the wrapped endpoint, then start TLS.

The TLS negotiation is done by way of wrapping the provided factory with TLSMemoryBIOFactory during connection.

Returns A `Deferred` which fires with the same `Protocol` as `wrappedEndpoint.connect(fac)` fires with. If that `Deferred` errbacks, so will the returned deferred.

Python Module Index

t

txsocksx.client, 7
txsocksx.http, 8
txsocksx.tls, 9

Index

C

connect() (txsocksx.client.SOCKS4ClientEndpoint method), 7
connect() (txsocksx.client.SOCKS5ClientEndpoint method), 8
connect() (txsocksx.tls.TLSWrapClientEndpoint method), 9

E

endpointFactory (txsocksx.http.SOCKS4Agent attribute), 8
endpointFactory (txsocksx.http.SOCKS5Agent attribute), 9

S

SOCKS4Agent (class in txsocksx.http), 8
SOCKS4ClientEndpoint (class in txsocksx.client), 7
SOCKS5Agent (class in txsocksx.http), 8
SOCKS5ClientEndpoint (class in txsocksx.client), 7

T

TLSWrapClientEndpoint (class in txsocksx.tls), 9
txsocksx.client (module), 7
txsocksx.http (module), 8
txsocksx.tls (module), 9