Onion Plan

Usability Roadmap - Technical details

Silvio Rhatto

2022.Q4

Onion Support Group - The Tor Project

Technical details

Technical details

1. Context.
2. What really matters?
3. What can easily be replaced or thrown away?
4. Why pluggable?
5. Proposal 279 overview.
6. Phase 1 proof of concept.

1. Context

1. Context

- Onion Services being used by major actors; narrative shift; possibility of funding to do Onion Services development.
- Increasing team members focused on Onion Services.
- Might be a good time to plan this, considering the upcoming arti 2.0 roadmap focusing on Onion Services.

2. What realy matters in this proposal?

2. What realy matters in this proposal?

- Basic properties: incremental, modular etc.
- Opportunist discovery of .onion addresses.
- Pluggable methods.

3. What can easily be replaced or thrown away?

4. Why pluggable?

4. Why pluggable?

- Allow third parties to provide their own discovery methods regardless of what Tor ship by default as supported technology.
- Technical and governance criteria.

5. Proposal 279

Proposal 279 (2016)

> *[...] a modular Name System API (NSA) that allows developers to integrate their own name systems in Tor. [...] It should be flexible enough to accommodate all sorts of name systems*
> *[...] Tor asks the name system to perform name queries, and receives the query results. [...] It aims to be portable and easy to implement.*

See https://gitlab.torproject.org/tpo/core/torspec/-/blob/main/proposals/279-naming-layer-api.txt

What it brings

```
# New torrc(5) config
OnionNamePlugin 0 .hosts.onion    /usr/local/bin/local-hosts-file
OnionNamePlugin 1 .zkey.onion     /usr/local/bin/gns-tor-wrapper
OnionNamePlugin 2 .bit.onion      /usr/local/bin/namecoin-tor-wrapper
OnionNamePlugin 3 .scallion.onion /usr/local/bin/community-hosts-file
```

Implementations

- TorNS (2017-2019):
  - Tor NS API proof of concept using txtorcon.
  - https://github.com/meejah/torns
- StemNS:
  - TorNS fork using Stem.
  - https://github.com/namecoin/StemNS
- C Tor, arti and Tor Browser:
  - Still to be developed.

What if. . . ?

```
# New torrc(5) config
OnionNamePlugin  0 .some.onion /usr/bin/some-onion-resolver    # Phase 3
OnionNamePlugin 98 *           /usr/bin/dns-to-onion-resolver  # Phase 1
OnionNamePlugin 99 *           /usr/bin/sauteed-onion-resolver # Phase 2
```

Which means

1. In Phase 1, the DNS-based address translation is implemented.
2. In Phase 2, the Sauteed Onions address translation is implemented.
3. In Phase 3, "pure" Onion Name plugins can be officially included.
4. Matching will happen from the specific (like `.some.onion`) to the general (*).
5. For non-.onion TLDs, priority will be from the DNS to the Sauteed Onion (or other fancier methods).

Does prop279 should be amended or replaced?

- Extensive evaluation of what needs to be defined/done.

6. Phase 1 proof of concept

DNS, TLS SNI and .onion: proof of concept

Setup:

- An existing site: https://autodefesa.org.
- It's existing Onion Service: `autodefcecpx2mut5medmyjxjg2wb6lwkbt3enl74frthemyoyclpiad.onion`.

Today's behavior

- Attempt to access https://autodefcecpx2mut5medmyjxjg2wb6lwkbt3enl74frthemyoyclpiad.onion.
- Address is hard to remember.
- HTTPS connection will fail since the certificate is not valid for the .onion address.

Querying for an onion TXT record

Just an example, without DNSSEC, output formatted for readability:

```
$ dig autodefesa.org TXT

[...]

;; ANSWER SECTION:
autodefesa.org.     3600    IN  TXT
  "onion=autodefcecpx2mut5medmyjxjg2wb6lwkbt3enl74frthemyoyclpiad.onion"

;; Query time: 60 msec
[...]
```

Testing SNI

If we use OpenSSL via Tor, we can get the cert via Onion Service using TLS SNI:

```
torsocks openssl s_client -servername autodefesa.org \
  -tlsextdebug -connect \
  autodefcecpx2mut5medmyjxjg2wb6lwkbt3enl74frthemyoyclpiad.onion:443
```

Using curl

This could work in theory to fetch the site via Onion Services using TLS SNI:

```
torsocks curl -vik --resolve \
  autodefesa.org:443:autodefcecpx2mut5medmyjxjg2wb6lwkbt3enl74frthemyoyclp
  https://autodefesa.org
```

But it won't work, since `curl(1)`'s `--resolve` requires an IP address.

Using OpenSSL

Workaround with OpenSSL:

```
echo -e \
  "GET / HTTP/1.1\r\nHost:autodefesa.org\r\n\r\nConnection: Close\r\n\r\n" |
  torsocks openssl s_client -quiet -servername autodefesa.org -connect \
  autodefcecpx2mut5medmyjxjg2wb6lwkbt3enl74frthemyoyclpiad.onion:443
```

Result: page is fetched via Onion Service and HTTPS with a validated certificate!