Onion Plan

Usability Roadmap Proposal

Silvio Rhatto

2022.Q4

Onion Support Group - The Tor Project

Intro

What

Onion Plan: an ongoing strategy to increase the adoption an enhance the usability of Onion Services.

Have you ever?

Have you ever considered that we work with one of the coolest technologies?

And that our job consists in making it even cooler?

Now imagine

Imagine a **communication technology** that has:

1. Built-in resistance against surveillance, censorship and denial of service.
2. Built-in end-to-end encryption.
3. An address space bigger than IPv6 and without allocation authority.
4. Support for multiple, pluggable naming systems.
5. And that also works as an anonymization layer.

Enhanced Onion Services

We may call this technology **Enhanced Onion Services**!

Note a shift in how the technology is presented: instead of first stating that it's an anonymization technology, now the focus is *protection against surveillance, censorship and DoS* with *built-in anonymity in the Onion Service protocol*. This can make it easier to showcase the techology and attract potential funders.

What's still missing

1. Built-in DoS resistance.
2. Pluggable discoverability (multiple naming systems).
3. Many other enhancements in usability and tooling.

Tracks

This plan is split into the following roadmap tracks:

1. Health: DoS protections, performance improvements etc.
2. **Usability: Onion Names, Tor Browser improvements etc.**
3. Tooling: Onionbalance, Onionprobe, Oniongroove etc.
4. Outreach: documentation, support, usage/adoption campaigns etc.

Usability

Usability

Proposals grouped in these categories:

1. **Address translation**: links a "traditional" domain name with an Onion Service address. Examples: *Onion-Location*; *Sauteed Onions*; *DNS-based*, *Alt-Svc*.
2. **Onion Names**: alternative schemes for human-friendly names linked with Onion Services. Examples: ruleset-based (like *Secure Drop's Onion Names*); blockchain-based (like *Namecoin*); other P2P-based (like GNUnet's *LSD*); etc.
3. **HTTPS certificates**: easier integration of CA-validated TLS certificates into Onion Services. Examples: *ACME for .onion*; *X.509 for .onion* (self-signed by the .onion address).

Status

1. **Address translation**: some implemented (*Onion-Location*, *Alt-Svc*), others are still research (*Sauteed-Onions*).
2. **Onion Names**: many proposals, difficult to evaluate, difficult to decide.
3. **HTTPS certificates**: needs work and currently the Let's Encrypt team may not be available for this.

So... what can we do???

Usability Roadmap

Disclaimer

- Here follows a **non-orthodox strategy** to improve Onion Services UX.
- It's meant to **balance** between the present and **urgent user needs** and the wish to have **fully distributed Onion Names in the future**.
- It's an **incremental** roadmap, focusing on what's more **feasible** to do first instead of targeting in systems that still need to mature.

Usability Roadmap

- **Focus: human-friendly** names for Onion Services with **HTTPS support**.
- **Goal: coexistence** between different methods and **opportunistic discovery**.
- **Characteristics**: **pragmatic, modular, incremental, backwards compatible, future-proof and risk-minimizing** phases.

Phases

- **Phase 0**: current functionality.
- **Phase 1**: **accessing URLs** like `https://torproject.org` **directly** using Onion Services and HTTPS!
- **Phase 2**: **opportunistic discovery** of .onion addresses (increased censorship resistance).
- **Phase 3**: bringing "pure" **Onion Names** into Tor.

Phases comparison

Phase	Category	Method	Technology	Status
0	Addr. trans.	Onion-Location v1, Alt-Svc	HTTP	Done
1	Addr. trans.	DNS-based discovery	DNS	Planning
2	Addr. trans.	Sauteed Onions	CT Logs	Research
3	Onion Names	?	P2P/Blockchain	Research

Decentralization comparison

Phase	Technology	Decentralization
0	HTTP headers	Centralized (a single point of failure)
1	DNS	Very decentralized, but hierarchical
2	CT Logs	Decentralized, non-hierarchical, but few nodes
3	P2P/Blockchain	Decentralized, non-hierarchical, many nodes

Censorship resistance comparison

Phase	Technology	Censorship resistance
0	HTTP headers	Does not work when the site is blocked
1	DNS	Even if site is blocked, not if DNS is
2	CT Logs	Even if site/DNS blocked, not if CT Logs is
3	P2P/Blockchain	Should be fully censorship resistant

Phase 0

We're at Phase 0, but not starting from zero! :)

- We have **Onion Services v3**!
- We have accumulated lots of **discussions**, **proposals** and **analyses**.

Phase 1

Objective: accessing URLs like `https://torproject.org` directly using Onion Services and HTTPS!

That means:

1. It *can be transparent*, either by always preferring the Onion Service or using it automatically if the regular site is blocked.
2. Users will not need to know the actual Onion Service address!
3. Can work for all clients and not only Tor Browser.

Tor Browser

For Tor Browser, it can be possible to have special interface indicators to inform users:

- How the connection to the site is happening.
- Which available connection options exists for the site (regular or via .onion) as an **improved ".onion available" widget**.

But how it would work?

1. **Transparent resolution** of `torproject.org` into `2gzyxa5ihm7nsggfxnu52rck2vv4rvmdlkiu3zzui5du4xyclen53wid.onion` using **DNS via Tor** with (optional?) **signature checking** (DNSSEC?).
2. Use the **existing HTTPS certificate** for `torproject.org`, with no need to have `2gzyxa5ihm7nsggfxnu52rck2vv4rvmdlkiu3zzui5du4xyclen53wid.onion` in the certificate!
3. **Transparent** TLS SNI (**Server Name Indication**) connection to `https://2gzyxa5ihm7nsggfxnu52rck2vv4rvmdlkiu3zzui5du4xyclen53wid.onion` using `torproject.org` as the server name.

What it needs to work?

1. Transparent resolution:
   - Proposal 279 - specs for a Tor Name System API: review and implementation.
   - Define a way to securely add Onion Service addresses entries into the DNS.
   - Write a Tor NS API plugin that securely maps regular domains into Onion Services.
   - Minimum UX changes in the Tor Browser.
2. HTTPS Certificates: Already supported! No need to coordinate with Let's Encrypt or any other Certificate Authority.
3. TLS SNI: Already supported!

Phase 2

Objective: *increase the censorship resistance* of accessing URLs like `https://torproject.org` directly using Onion Services and HTTPS!

That means:

- Implementing *opportunistic discovery* of Onion Service addresses by having an additional method to get the .onion address for `torproject.org`.
- In this phase, a Sauteed Onions Tor NS plugin will be created.

Phase 3

Objective: bring "pure" / "real" Onion Names into Tor.

That means:

- Transparent access to `http://somesite.some.onion`.
- Having techincal and governance specs to decide which Onion Names are officially accepted.
- Allocating a namespace (at `.onion`?) to each proposal.
- Optionally shipping the implementation into a bundle for distribution.

Technical details

Technical details

- Proposal 279 overview.
- DNS, TLS SNI and .onion proof of concept.

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
```

TorNS (2017-2019)

- Tor NS API proof of concept.
- https://github.com/meejah/torns

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).

A 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.

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!

More information

More information

Check the full Onion Plan Usability Roadmap Proposal.