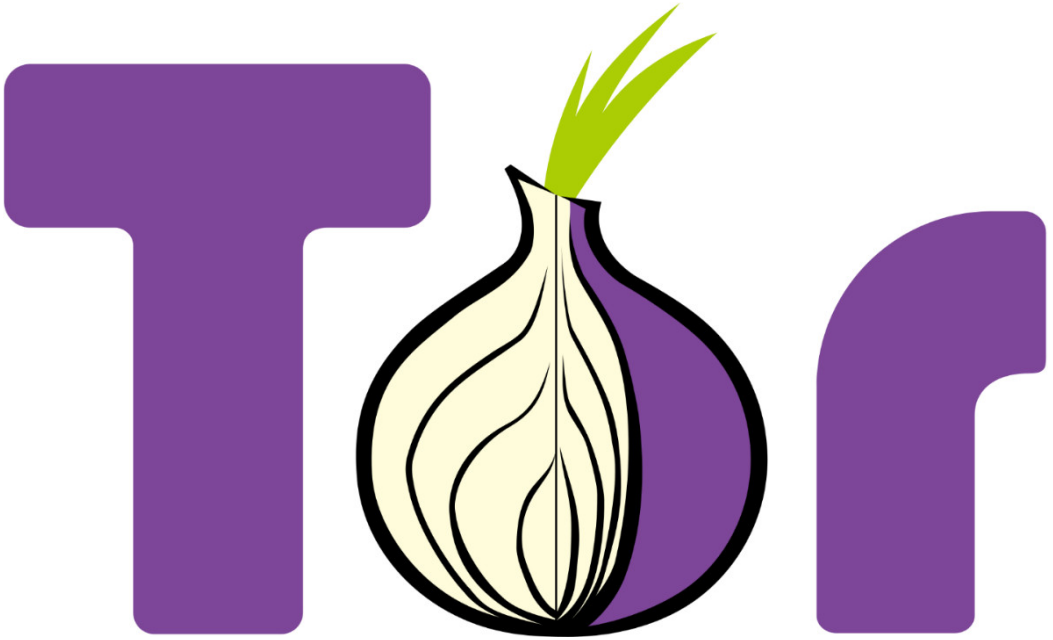

Tor

Topics

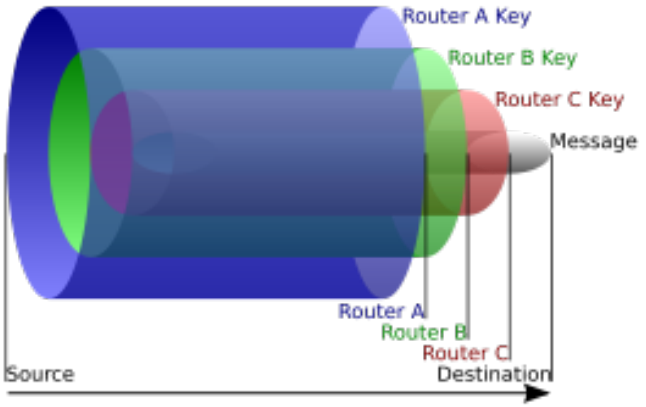
- Onion Routing
- Early History
- What is Tor?
  - Why do we need Tor?
  - Who uses Tor?
  - Elephants in the Room
- Vulnerabilities
- How It Works
- Onion Services
- Talking to Tor
- Other privacy tools

Onion Routing

Layers of an Onion

- Onion proxy client downloads list of nodes from a directory node.
- A route is chosen at random to the destination.

  This is called the circuit and can be any number of hops.
- A layer of encryption wraps the data for each node in the circuit.
- No node knows how many nodes constitute the circuit.



- No node knows if the preceding one was the initiator or just another node in the circuit.
- Every node only knows the nodes directly before and after it.
- The only node that knows the destination address is the last, the exit node. Similarly, its the only node that knows its place in the circuit.
- If the destination uses TLS, the circuit never knows the contents of the payload. Otherwise, the exit node can read the data.

Early History of Onion Routing, 1995 - 2006

1990s

1995

- Work on Onion Routing begins by the Office of Naval Research (ONR) to secure U.S. intelligence communications (Generation 0).
- Principals involved are Paul Syverson, Michael G. Reed and David Goldschlag.

1996

- Work on Generation 1 begins.
- First proof-of-concept prototype deployed.
- Interestingly, crypto is removed from code because of export restrictions.
- Publicly distributed as “open source” to instill trust.

1997

- DARPA begins funding under its High Confidence Networks Program.

1998

- Several Generation 0 and 1 networks are established. One such is a distributed network of 13 nodes with an average of over 50,000 hits a day, which maxes out the usage.
- A commercial network called Freedom Networks was established with many similarities to Onion Routing. Differences:
  - Ran over UDP
  - Commercially funded rather than volunteer-based
  - Management to limit use to a paid subscription model

1999

- Development is suspended on Onion Routing due to lack of funding and principal developers moving on to other pursuits.

2000s

2000

- Syverson and Roger Dingledine meet at a workshop, the seeds of Tor are sown.

2001

- Development resumes after renewed funding from DARPA.
- Edison Invention Award presented for the invention of Onion Routing.
- Freedom Network shutters from lack of commercial interest.

2002

- Generation 1 code abandoned as too crufty, work begins on Generation 2.
- Initial work that becomes Tor is done at Cambridge University, but by 2004 none of this code remains in the Tor codebase.
- Design is simplified, much in-house proxying code is removed:
  - Privoxy is used to filter (anonymize) data streams
  - SOCKS proxies are used.
  - Export restrictsion on crypto are lifted.
- Syverson, Dingledine and Nick Mathewson continue development on what will become Tor, the largest implementation of onion routing.

2003

- Increased funding from DARPA, ONR and the U.S. Naval Research Laboratory (NRL).
- Tor deployed in October by Syverson, Dingledine and Mathewson with ~12 volunteer nodes, all in the U.S. but one (in Germany). All code is publicly available under the MIT license.

2004

- Hidden services are deployed and the Tor design paper is published.
- Funding from DARPA and ONR ends, EFF begins its (continued) funding of the Tor project.
- By the end of the year, there are 100 Tor nodes on three continents.

2006

- Dingledine, Mathewson, et al. founded the Tor Project as a non-profit.

Post 2006

This brief history only covers the early years of Onion Routing.

For more history, and especially history specific to Tor, visit the Tor Project.

What is Tor?

The Tor Project

- A Massachusetts-based non-profit with a respected and well-known Board of Directors
  - Bruce Schneier
  - Matt Blaze
  - Cindy Cohn (EFF)
  - Et al.
- An implementation of Onion Routing (2nd generation)
- A switched packet network that defends against traffic analysis
  - Use cases:
    1. Routes traffic through Tor network to open Internet
    2. Traffic stays internal and never leaves Tor network
       - End-to-end anonymity
       - Tor onion services, aka “The Dark Web”
- Run by volunteers that donate bandwidth and processing power, i.e., relays (servers) that route the traffic through the Tor network
- Open source
- Projects:
  - Tor Browser
  - Tails
  - Orbot
  - Metrics Portal
  - Et al.

Why do we need Tor?

- Provides anonymity and privacy
- First Amendment protection
- Connections should be private by default
- Best strategy currently available*

* What about VPNs? Services often lie about what they log and for how long, and they are subject to FISA warrants.

Who uses Tor?

- Regular people
  - Anyone who values privacy
  - Citizens in repressive regimes
  - Programmers and sysadmins
  - Mom and Dad
- Journalists
- Activists
- Whistleblowers and dissidents
- Governments
- Law enforcement
- Bad actors

Elephants in the Room

- But there are bad actors!
  - They also use public proxies and end-to-end encryption (VPNs, GPG, etc).
- It’s the Dark Web!
  - Marketing term that MSM loves….scary!
  - Many legitimate sites, such as ProPublica, DuckDuckGo and Facebook, also run onion servers
- Let’s put in back doors for the “good guys”!
  - Bad actors will find and leverage back doors
  - And who are the “good guys”, anyway?
- Tor is safe!
  - Unless you’re being targeted by a nation state and/or an entity with deep pockets
  - If NSA wants in, they’re in
  - Safety is directly proportional to the number of Tor users!
- Tor isn’t safe!
  - Yes, the NSA runs many exit nodes, but that’s only half the battle
  - Traffic analysis needs a concerted effort
  - The more Tor users, the safer (needle in a haystack)

Vulnerabilities

Traffic-Analysis Attack

- Timing-analysis attack
- Try to deduce information from patterns in data flows, i.e., timing, frequency, size, etc. on one side of the network and look for the same patterns on the other side. This will tell an attacker the circuit a particular user is using.
  - Passive attack– Simply observe packets
  - Active attack – Alter timings of packets, inject extra packets into a data flow in a specific pattern (watermarking), etc.
- Some successful attacks of Tor have occurred in highly controlled academic environments with no timing noise added to the packets.
- Successful counter-measures include encryption, masking whereby data is continuously sent whether or not traffic is actually being transmitted (dummy traffic) and buffering to introduce delays to thwart timing analysis.
  - Adaptive Padding – add packets to flow
  - Defensive Dropping – remove added packets from flow
  - Gamma Buffering – buffer at node
- Resources:
  - https://resources.infosecinstitute.com/timing-analysis-attacks/

Mouse Fingerprinting Attack

- Mouse movements can be unique (a fingerprint) and then used to correlate and identify a user.
- The user would have had to visit the same fingerprinting site with both the Tor browser and a non-Tor browser.
- The same researcher demonstrated that a machine can also be fingerprinted by the time it takes to run a CPU-intensive task in JavaScript.
- There are currently multiple open bug reports on the Tor bug tracker to address this.
- The best solution is to turn off JavaScript, which is a branch (or at least a leaf) on the tree of evil.

How It Works

Brief Overview

- When started, Tor will create a hidden `.tor` directory in the user's home directory, the contents of which depends upon the contents of `torrc`.
- Builds a circuit of three relays (guard, middle and exit relays) between the initiator and the responder (Internet)
- Each node only knows about the relay before and after
  - The guard or entry node is the only relay that knows the true IP address of the initiator
  - The exit node is the only node that knows the destination of the responder
- Layers of encryption (*onion* routing) are generated by initiator before leaving host machine
  - Public key cryptography
  - Payload is at the center of the "onion" surrounded by layers of encryption
  - Each relay unwraps a layer of encryption
  - Exit node unlocks final encryption layer, is able to read destination `TCP/IP` headers and sends to responder over open Internet
  - Each packet is uniform in length (cell payloads are 509 bytes)

How do I know Tor is working?

- If using Tor browser:
  - Visit https://check.torproject.org/
  - Verify IP address of exit node:
    - IP Chicken
- Verify IP address of exit node in server logs:
  - `sudo tail -f /var/log/nginx/access.log`
- Sniff traffic network traffic:
  - Localhost
    - `sudo tcpdump -nX host <ip.of.guard.node> and port 443`
  - Remote
    - `sudo tcpdump -nX host <ip.of.exit.node> and port 80`

Tor traffic can be censored

- The IP addresses of all Tor exit nodes are published, so it’s trivial to blacklist those IPs.
  - Tor bridges are the solution!
- Censors can perform deep packet inspection between the sender and the guard relay.
- Tor has developed pluggable transports so traffic between the initiator and the first hop (guard relay) can’t be identified as Tor network traffic.

What not to do when using Tor

- Use a browser other than the Tor browser!
- Download browser plugins
- Download files, i.e., pdf and Word docs whose loader programs could then leak your true IP address if the docs contain links to other resources
- Use BitTorrent

Onion Services

http://lgewyajrjxytj4z6.onion/

- Formerly known as hidden services
- Provides end-to-end anonymity
  - Traffic never leaves the Tor network.
  - The responder is anonymous in addition to the initiator.
- No DNS
- Harder to find sites. Most onion addresses are passed by word-of-mouth, email, posted on individual sites, etc.
  - Here are some well-known ones:
    - ProPublica
    - The Intercept SecureDrop server
    - DuckDuckGo
    - Ahmia
    - Silk Road (defunct)
- The “Dark Web”

How does it work?

1) Select the introduction points.
2) Advertise that the service is available.
3) The client downloads the descriptor and sets up a rendezvous point.
4) The client requests an introduction of the host.
5) The onion service creates a circuit to the rendezvous point.
6) The client and service communicate via the rendezvous point.

Add HiddenService* directives to torrc

```
/usr/local/etc/tor:$ sudo cp torrc{,.orig}

/usr/local/etc/tor:$ su -
Password:

root@trout:~# cat >> /usr/local/etc/tor/torrc
HiddenServiceDir /home/btoll/hidden_service/
HiddenServicePort 80 127.0.0.1:1972

root@trout:~# exit
Logout

/usr/local/etc/tor:$ killall -HUP tor

# Tor adds new directory hidden_service/ to home directory.

~:$ ls -R hidden_service/
hidden_service/:
hostname  private_key

~:$ cat hidden_service/hostname
tiucrrm2slunknhb.onion
```

Add new `server` block to nginx

/etc/nginx/sites-available/default

```
server {
    listen 1972;
    listen [::]:1972;

    root /var/www/html;

    index index.html index.htm

    server_name tiucrrm2slunknhb.onion;

    location / {
        try_files $uri $uri/ =404;
    }
}
```

Talking to Tor

Add ControlPort and CookieAuth directives to torrc

```
/usr/local/etc/tor:$ sudo cp torrc{,.orig}

/usr/local/etc/tor:$ su -
Password:

root@trout:~# cat >> /usr/local/etc/tor/torrc
ControlPort 9051
CookieAuthentication 1

root@trout:~# exit
Logout

/usr/local/etc/tor:$ killall -HUP tor

# Tor adds new file control_auth_cookie to ~/.tor data directory.
```

Talk to Tor (telnet)

```
~:$ telnet localhost 9051
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
PROTOCOLINFO
250-PROTOCOLINFO 1
250-AUTH METHODS=COOKIE,SAFECOOKIE
COOKIEFILE="/home/btoll/.tor/control_auth_cookie"
250-VERSION Tor="0.3.4.0-alpha-dev"
250 OK

~:$ hexdump -e '32/1 %02xn' ~/.tor/control_auth_cookie
be9c9e18364e33d5eb8ba820d456aa2bc03444c0420f089ba4569b6aeecc6254

~:$ telnet localhost 9051
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
AUTHENTICATE be9c9e18364e33d5eb8ba820d456aa2bc03444c0420f089ba4569b6aeecc6254
250 OK
GETINFO version
250-version=0.2.5.1-alpha-dev (git-245ecfff36c0cecc)
250 OK
QUIT
250 closing connection
Connection closed by foreign host.
```

Stem python library

- Part of the Tor Project
- Script against the Tor process
- Useful for:
  - Creating onion services
  - Inspecting metadata about a Tor circuit and nodes
  - Subscribe to Tor events
  - Many others!
- Available for download with most (all?) Unix package managers
  - `sudo apt-get install python-stem python3-stem`

Talk to Tor (python)

```
~:$ tor-prompt
Jul 24 14:28:46.000 [notice] New control connection opened from 127.0.0.1.
Welcome to Stem's interpreter prompt. This provides you with direct access to
Tor's control interface.

This acts like a standard python interpreter with a Tor connection available
via your 'controller' variable...

  >>> controller.get_info('version')
  '0.2.5.1-alpha-dev (git-245ecfff36c0cecc)'

You can also issue requests directly to Tor...

  >>> GETINFO version
  250-version=0.2.5.1-alpha-dev (git-245ecfff36c0cecc)
  250 OK

For more information run '/help'.

>>> /info moria1
moria1 (9695DFC35FFEB861329B9F1AB04C46397020CE31)
address: 128.31.0.34:9101 (moria.csail.mit.edu, us)
tor version: 0.3.4.5-rc-dev
flags: Authority, Fast, HSDir, Running, Stable, V2Dir, Valid
exit policy: reject *:*
contact: 1024D/28988BF5 arma mit edu
```

Example: Query Consensus for Relay Descriptors

1) Create the python script:

```
~:$ cat > get_consensus.py
import stem.descriptor.remote

try:
  for desc in stem.descriptor.remote.get_consensus().run():
    print("found relay %s %s %s" % (desc.nickname, desc.address, desc.fingerprint))
except Exception as err:
  print("Unable to retrieve the consensus: %s" % err)
```

2) Write consensus to file and query for a specific relay node:

```
~:$ python get_consensus.py | tee consensus | ag loki
found relay lokid 212.19.17.213 4EC47AB2DB37C8EDB7068A04B36DA25BD6BC178F
found relay loki 51.15.145.150 5A6451D4E4B4FFDE0B2682D8D8DAA0D10A500066
found relay Loki 104.244.75.194 C8850DE0EBC07481808F32F2BAA76CA65CB659FB
```

3) Subsequent searches don’t need to burden the network:

```
~:$ cat consensus | ag moria1
found relay moria1 128.31.0.34 9695DFC35FFEB861329B9F1AB04C46397020CE31
```

Example: Query Tor Process for Relay Descriptors

1) Add to .torrc. As root:

```
root@trout:~# cat >> /usr/local/etc/tor/torrc
FetchDirInfoEarly 1
FetchDirInfoExtraEarly 1
FetchUselessDescriptors 1
DownloadExtraInfo 1
```

2) SIGHUP so it reloads its config (if started Tor as a daemon):

```
~:$ killall –HUP tor
```

3) List the .tor data directory, there will be new entries (triggered by the 3rd directive above). If anything listed below is missing, it will be there after restarting the Tor process:

```
cached-consensus
cached-descriptors
cached-descriptors.new
cached-extrainfo
Cached-extrainfo.new
```

4) Create the python script:

```
~:$ cat > get_descriptors.py
from stem.descriptor import parse_file

for desc in parse_file('/home/btoll/.tor/cached-consensus'):
     print('found relay %s %s %s' % (desc.nickname, desc.address, desc.fingerprint))
```

Other privacy tools

- Tails
- tor-resolve
- torsocks
- GNU Privacy Guard (GPG)
- Signal
- OTR
- DuckDuckGo

FIN

Benjamin Toll
benjamintoll.com
benjam72@yahoo.com