Attacking and Improving the Tor Directory Protocol

IEEE S&P 2024

Zhongtang Luo\textsuperscript{1}  Adithya Bhat\textsuperscript{1}  Kartik Nayak\textsuperscript{2}  Aniket Kate\textsuperscript{13}

\textsuperscript{1}Purdue University,  \textsuperscript{2}Duke University,  \textsuperscript{3}Supra Research
Tor: the Onion Router
Tor: the Onion Router

- Anonymous communication service
- around 4 million users daily.

Source: https://www.torproject.org/
How Tor Works: 1

Step 1: Alice's Tor client obtains a list of Tor nodes from a directory server.

Source: https://tor-https.eff.org/
Step 2: Alice's Tor client picks a random path to destination server. Green links are encrypted, red links are in the clear.
Tor Directory Servers
Volunteered servers collect relay information (IP, public key, etc.) and publish them.

Clients pick one server and fetch the information.

— Tor directory protocol, version 3
Source: https://spec.torproject.org/dir-spec/
Volunteered servers collect relay information (IP, public key, etc.) and publish them.

Clients pick one server and fetch the information.

Problem?
- More servers = less security

"Every directory authority was a trust bottleneck: if a single directory authority lied, it could make clients believe for a time an arbitrarily distorted view of the Tor network. (Clients trusted the most recent signed document they downloaded.) Thus, adding more authorities would make the system less secure, not more."

— Tor directory protocol, version 3

Source: https://spec.torproject.org/dir-spec/
Clients **download from multiple servers** and aggregate the information locally.

Problem?

Directories had grown quite large. (∼5 MiB × 9 authorities × 4 million users daily = ∼180 TiB)

Partition Attack: different set of documents = different usage pattern

"It was possible under certain perverse circumstances for clients to download an unusual set of network status documents, thus partitioning themselves from clients who have a more recent and/or typical set of documents."

— Tor directory protocol, version 3

Source: [https://spec.torproject.org/dir-spec/](https://spec.torproject.org/dir-spec/)
Clients **download from multiple servers** and aggregate the information locally.

**Problem?**

- Directories had grown quite large. ($\sim 5 \text{ MiB} \times 9 \text{ authorities} \times 4 \text{ million users daily} = \sim 180 \text{ TiB}$)
- Partition Attack: different set of documents = different usage pattern

“It was possible under certain perverse circumstances for clients to download an unusual set of network status documents, thus **partitioning themselves** from clients who have a more recent and/or typical set of documents.”

— Tor directory protocol, version 3

Source: [https://spec.torproject.org/dir-spec/](https://spec.torproject.org/dir-spec/)
A small set (9 as of now) of semi-trusted directory authorities hard-coded

- **Every hour** send a signed summary of this view (a “status vote”) to the other authorities.
- Compute the result of this vote including relay properties, and sign a **consensus document**
- A consensus document is valid if **more than half** of the authorities signed on it.
- Directory caches download, cache, and re-serve consensus documents (saving bandwidth).

- Proposed around 2007
  - Longest running blockchain (kind of) earlier than Bitcoin!
How Secure Is the Current Tor Directory Protocol?
Empirically...

Tor failed to make a consensus 2010/08/29 17:00

Karsten found this in his log (note that aabelmoo is running on ubc(2)):

Dir auths using an unsustainable 400+ mbit/s, need to diagnose and fix

We've been having problems establishing a consensus lately. We realized that maatuska was rate limiting to only 10MBytes/s, and asked Linus to bump it up, so he did. Then today we realized that morial was unable to serve dirport answers because it was maxed out at its BandwidthRate of 30MBytes. I raised that to 50MBytes and it stayed maxed out. I have put it back down to 30MBytes so my host doesn't get too upset.

This is not a sustainable situation. We need to figure out what is asking the dir auths for so many bytes, and get it to stop or slow down.

This is a ticket to collect info and to brainstorm ideas.

To upload designs, you'll need to enable LFS and have an admin enable hashed storage. More Information

Source: https://gitlab.torproject.org/tpo/core/tor/-/issues/1890
Source: https://gitlab.torproject.org/tpo/core/tor/-/issues/33018

Suffers problems from network instability & DDoS

Attacking and Improving the Tor Directory Protocol
Mock example of three authorities:

- $E$ is bad
- $H_1$ votes 0
- $H_2$ votes 1

There are different consensuses!

- $E$ equivocates
  - votes 0 to $H_1$
  - votes 1 to $H_2$
- $H_1$ signs 0
- $H_2$ signs 1
Any attacker can find one property of some relay that authorities split their votes on
- Equivocating on it causes correct authorities to sign on different things!
- Recall that a consensus document is valid if more than half of the authorities signed it
- So if the division is 4-4, no valid consensus document
If the attacker signs and publishes one of the consensus documents, then the other is hidden (since a 4-signature document is not publishable)

- Client observes no difference as a consensus with 5 signatures is the same as one with 9
- The attacker can still sign on the other consensus privately and obtain 5 signatures
  - Bad things on the document — anonymity attack
  - Circumvents all public detection measures that Tor is heavily reliant on (depictor, TorDoc, etc.)
Attack 2: Bandwidth Attack

- Attacker publishes a new honeypot relay with very large available bandwidth
  - **Group of 3 authorities** can dictate bandwidth for one relay
  - Trick clients to use with very high probability
- Usually very “loud” (detectable with public tools)
  - Main defense mechanism is public detection
  - With equivocation attack the attacker can make it silent!
- Tested
  - Verified on testnet
/* Pick a bandwidth */
if (num_mbws > 2) {
    rs_out.has_bandwidth = 1;
    rs_out.bw_is_unmeasured = 0;
    rs_out.bandwidth_kb = median_uint32(measured_bws_kb, num_mbws);
} else if (num_bandwidths > 0) {
    rs_out.has_bandwidth = 1;
    rs_out.bw_is_unmeasured = 1;
    rs_out.bandwidth_kb = median_uint32(bandwidths_kb, num_bandwidths);
    if (n_authorities_measuring_bandwidth > 2) {
        /* Cap non-measured bandwidths. */
        if (rs_out.bandwidth_kb > max_unmeasured_bw_kb) {
            rs_out.bandwidth_kb = max_unmeasured_bw_kb;
        }
    }
}
Attack 3: Sybil Relay Attack

- Attacker publishes a ton of new honeypot relays
- Also usually "loud" (detectable with existing work and public tools)
  - The main defense mechanism is only public detection
  - No longer detectable with our attack!
Takeaway

- Tor makes heavy use of public audit mechanism
  - With the attack we can circumvent the audit in various ways
- Illustrates a meaningful relationship between real-life attacks and theoretical security
- We may not even know if such an attack happened in the past!
How Can We Fix & Improve the Protocol?
Reactively... We Can Detect Irregularities
TorEq: Online Equivocation Detector

- Detection mechanism to see if irregular activities take place
- Periodically polls every authority for every vote received
- Merged into Tor codebase and live online!

https://consensus-health.torproject.org/

Variety of votes
This table monitors the votes each authority receives from other authorities.

<table>
<thead>
<tr>
<th>Sender</th>
<th>Receiver</th>
</tr>
</thead>
<tbody>
<tr>
<td>moria1</td>
<td>moria1, tor26, dizum, gabelmo, dannenberg, maatuka, longlaw, bastet</td>
</tr>
<tr>
<td>tor26</td>
<td>moria1, tor26, dizum, gabelmo, dannenberg, maatuka, longlaw, bastet</td>
</tr>
<tr>
<td>dizum</td>
<td>moria1, tor26, dizum, gabelmo, dannenberg, maatuka, longlaw, bastet</td>
</tr>
<tr>
<td>gabelmo</td>
<td>moria1, tor26, dizum, gabelmo, dannenberg, maatuka, longlaw, bastet</td>
</tr>
<tr>
<td>dannenberg</td>
<td>moria1, tor26, dizum, gabelmo, dannenberg, maatuka, longlaw, bastet</td>
</tr>
<tr>
<td>maatuka</td>
<td>moria1, tor26, dizum, gabelmo, dannenberg, maatuka, longlaw, bastet</td>
</tr>
<tr>
<td>longlaw</td>
<td>moria1, tor26, dizum, gabelmo, dannenberg, maatuka, longlaw, bastet</td>
</tr>
<tr>
<td>bastet</td>
<td>moria1, tor26, dizum, gabelmo, dannenberg, maatuka, longlaw, bastet</td>
</tr>
</tbody>
</table>
■ Performance is good
  ■ Max \(~5\) min to generate report

■ Limitation
  ■ Client places trust on the detector, creating a new trust bottleneck
  ■ If the detector is compromised then all bets are off
Proactively... We Can Improve the Protocol
What Problem Are We Solving?

- Produce a consensus document from a set of input
- The rules are complicated so local computation is easier
- **The Interactive Consistency Problem**: (Informally) Every server outputs the same set of values; Every correct server has their input in the set
- Implemented with parallel **Byzantine Broadcast**: (Informally) One (potentially faulty) server broadcasts to everyone; Every correct server agrees on the value broadcasted
System Model

- Currently **bounded-synchrony** setting; round time is 150 seconds
- Assume the attacker is **Byzantine**, i.e. may behave arbitrarily
- Targeting \((2f + 1)\) security
- **Particularities of Tor**
  - A very low (9) number of nodes
  - Large document size (5MB)
  - Dated PKI structure
    - Public randomness, etc. can be hard to implement
  - Minimize attack surface
    - Minimize time in optimistic case
DirCast: New Protocol for Tor Directory

- Optimistically 4 rounds (lowest) with communication complexity $O(n^3)$
- Two Phases: Bootstrap and Agreement
  - Bootstrap Phase
    - 2 rounds: Detects equivocation
    - Gives everyone a certificate $C(x)$ for no equivocation
  - Agreement Phase
    - No crash: Everyone agrees, 2 rounds
    - If there is crash: Up to $(f + 1)$ rounds, same as previous upper bound (Dolev-Strong, 1983)
DirCast: New Protocol for Tor Directory

Bootstrap Phase

Agreement Phase

Propose Round  
Vote Round  
Synchronize Round 1  
Synchronize Round \( f + 1 \)

\[
\begin{align*}
PROPOSE & \quad (x, \sigma_s(x)) \\
VOTE & \quad (x_i, \sigma_s(x_i), \sigma_i(x_i)) \\
SYNC & \quad (x, C(x), S(x))
\end{align*}
\]

\[
\begin{align*}
NOTIFY & \quad (x, \sigma_s^N(x), C(x)) \\
SYNC & \quad (x, C(x), S(x))
\end{align*}
\]

\[
\begin{align*}
NOTIFY & \quad (x, \sigma_i^N(x), C(x))
\end{align*}
\]
**DirCast: New Protocol for Tor Directory**

- **Implementation**
  - Message Compression: broadcast only the diff from the previous consensus document
  - Drastically cuts message size (~20%)
- **Performance:** Comparable to the current insecure protocol!

---

**Attacking and Improving the Tor Directory Protocol**

---
Coordinated Disclosure

- Reported on Apr 27, 2022
- Acknowledged on May 6, 2022
- Mitigation: Detector merged into the codebase on Aug 8, 2023
- Protocol: Tor is migrating to a Rust codebase, Arti
  - Looking to implement Directory Authority in Rust
  - Interesting collaboration ahead
Summary

- Discovered a protocol design vulnerability in the Tor consensus protocol
  - Exploited in the testnet that results in deanonymization

- Developed **TorEq**, a mitigation solution that reactively detects irregularities (already deployed)

- Proposed & prototyped **DirCast**, a secure improvement for the protocol