Ten ways to discover Tor bridges

Roger Dingledine
arma@torproject.org

Tor Tech Report 2011-10-002
October 31, 2011

0 Introduction

While we're exploring smarter ways of getting more bridge addresses [1], and while the bridge arms race hasn't heated up yet in most countries (or has surpassed the number of bridges we have, in the case of China), it's the perfect time to take stock of bridge address enumeration attacks and how well we can defend against them.

For background, bridge relays[1] (aka bridges) are Tor relays that aren't listed in the main Tor directory. So even if an attacker blocks all the public relays, they still need to block all these "private" or "dark" relays too.

Here are ten classes of attacks to discover bridges, examples of them we've seen or worry about in practice, and some ideas for how to resolve or mitigate each issue. If you're looking for a research project, please grab one and start investigating!

1 Overwhelm the public address distribution strategies

China broke our https bridge distribution strategy[2] in September 2009 by just pretending to be enough legitimate users from enough different subnets on the Internet. They broke the Gmail bridge distribution strategy[3] in March 2010. These were easy to break because we don't have enough addresses relative to the size of the attacker (at this moment we're giving out 176 bridges by https and 201 bridges by Gmail, leaving us 165 to give out through other means like social networks), but it's not just a question of scale: we need better strategies that require attackers to do more or harder work than legitimate users.

[1] https://www.torproject.org/docs/bridges
[2] https://bridges.torproject.org/
[3] https://www.torproject.org/docs/bridges#FindingMore

2 Run a non-guard non-exit relay and look for connections from non-relays

Normal clients use guard nodes[4] for the first hop of their circuits to protect them from long-term profiling attacks; but we chose to have bridge users use their bridge as a replacement for the guard hop, so we don't force them onto four-hop paths which would be less fun to use. As a result, if you run a relay that doesn't have the Guard flag, the only Tors who end up building circuits through you are relays (which you can identify from the public consensus) and bridges.

This attack has been floating around for a while, and is documented for example in Zhen Ling et al's Extensive Analysis and Large-Scale Empirical Evaluation of Tor Bridge Discovery[5] paper.

The defense we plan is to make circuits through bridges use guards too. The naive way to do it would be for the client to choose a set of guards as her possible next hops after the bridge; but then each new client using the bridge increasingly exposures the bridge. The better approach is to make use of Tor's loose source routing feature to let the bridge itself choose the guards that all of the circuits it handles will use: that is, transparently layer an extra one-hop circuit inside the client's circuit. Those familiar with Babel's design [3] will recognize this trick by the name "inter-mix detours".

Using a layer of guards after the bridge has two features: first, it completely removes the "bridges directly touch non-guards" issue, turning the attack from a deterministic one to a probabilistic one. Second, it reduces the exposure of the bridge to the rest of the network, since only a small set of relays will ever see a direct connection from it. The tradeoff, alas, is worse performance for bridge users. See proposal 188[6] for details.[7]

3 Run a guard relay and look for protocol differences

Bridges are supposed to behave like relays with respect to the users using them, but like clients with respect to the relays they make connections to. Any slip-ups we introduce where the bridge acts like a relay with respect to the next hop are ways the next hop can distinguish it. Recent examples include "bridges fetched directory information like relays rather than like clients"[8], "bridges didn't use a CREATE_FAST cell for the first hop of their own circuits like clients would have"[9], "bridges didn't reject CREATE and CREATE_FAST cells on connections they had initiated like clients would have"[10], and "bridges distinguish themselves in their NETINFO cell"[11].

There's no way that's the end of them. We could sure use some help auditing the design and code for similar issues.

[4] https://www.torproject.org/docs/faq#EntryGuards

[5] http://www.cs.uml.edu/~xinwenfu/paper/Bridge.pdf

[6] https://gitweb.torproject.org/torspec.git/blob/HEAD:/proposals/188-bridge-guards.txt

[7] A friendly researcher pointed out to me that another solution here is to run the bridge as multi-homed, meaning the address that the relay sees isn't the address that the censor should block. That solution also helps resolve issues 3-5!

[8] https://trac.torproject.org/projects/tor/ticket/4115

[9] https://trac.torproject.org/projects/tor/ticket/4124

[10] https://lists.torproject.org/pipermail/tor-commits/2011-October/036726.html

[11] https://trac.torproject.org/projects/tor/ticket/4348

4 Run a guard relay and do timing analysis

Even if we fix issues 2 and 3, it may still be possible for a guard relay to look at the "clients" that are connecting to it, and figure out based on latency that some of the circuits from those clients look like they're two hops away rather than one hop away.

I bet there are active tricks to improve the attack accuracy. For example, the relay could watch round-trip latency from the circuit originator (seeing packets go towards Alice, and seeing how long until a packet shows up in response), and comparing that latency to what he sees when probing the previous hop with some cell that will get an immediate response rather than going all the way to Alice. Removing all the ways of probing round-trip latency to an adjacent Tor relay (either in-protocol or out-of-protocol) is a battle we're not going to win.

The research question remains though: how hard is this attack in practice? It's going to come down to statistics, which means it will be a game of driving up the false positives. It's hard to know how best to solve it until somebody does the engineering work for the attack.

If the attack turns out to work well (and I expect it will), the "bridges use guards" design will limit the damage from the attack.

5 Run a relay and try connecting back to likely ports on each client that connects to you

Many bridges listen for incoming client connections on port 443 or 9001. The adversary can run a relay and actively portscan each client that connects, to see which ones are running services that speak the Tor protocol. This attack was published by Eugene Vasserman [8] and by Jon McLachlan [6], both in 2009.

The "bridges use guards" design partially resolves this attack as well, since we limit the exposure of the bridge to a small group of relays that probably could have done some other above attacks as well.

But it does not wholly resolve the concern: clients (and therefore also bridges) don't use their entry guards for directory fetches at present. So while the bridge won't build circuits through the relay it fetches directory information from, it will still reveal its existence. That's another reason to move forward with the "directory guard"[12] design.

6 Scan the Internet for services that talk the Tor protocol

Even if we successfully hide the bridges behind guards, the adversary can still blindly scan for them and pretend to be a client. To make it more practical, he could focus on scanning likely networks, or near other bridges he's discovered. We called this topic "scanning resistance" in our original bridge design paper [2].

There's a particularly insidious combination of 5 and 6 if you're a government-scale adversary: watch your government firewall for SSL flows (since Tor tries to blend in with SSL traffic), and do active followup probing to every destination you see. Whitelist sites you've

[12] https://lists.torproject.org/pipermail/tor-relays/2010-April/000112.html

already checked if you want to trade efficiency for precision. This scale of attack requires some serious engineering work for a large country, but early indications[13] are that China might be investigating exactly this approach.

The answer here is to give the bridge user some secret when she learns the bridge address, and require her to prove knowledge of that secret before the bridge will admit to knowing the Tor protocol. For example, we could imagine running an Apache SSL webserver with a pass-through module[14] that tunnels your traffic to the Tor relay once she's presented the right password. Or Tor could handle that authentication itself[15]. BridgeSPA: Improving Tor Bridges with Single Packet Authorization [7] offers an SPA-style approach, with the drawbacks of requiring root on both sides and being OS-specific.

Another avenue to explore is putting some of the bridge addresses behind a service like Telex [9], Decoy Routing [5], or Cirripede [4]. These designs let users privately tag a flow (e.g. an SSL handshake) in such a way that tagged flows are diverted to a Tor bridge while untagged flows continue as normal. So now we could deploy a vanilla Apache in one place and a vanilla Tor bridge in another, and not have to modify either of them. The Tor client bundle would need an extra piece of software though, and there are still some engineering and deployment details to be worked out.

7 Break into the Tor Project infrastructure

The bridge directory authority[16] aggregates the list of bridges and periodically sends it to the bridgedb service[17] so it can parcel addresses out by its various distribution strategies. Breaking into either of these services would give you the list of bridges.

We can imagine some design changes to make the risk less bad. For one, people can already run bridges that don't publish to the bridge directory authority (and then distribute their addresses themselves). Second, I had a nice chat with a Chinese NGO recently who wants to set up a bridge directory authority of their own, and distribute custom Vidalia bridge bundles to their members that are configured to publish their bridge addresses to this alternate bridge directory authority. A third option is to decentralize the bridge authority and bridgedb services, such that each component only learns about a fraction of the total bridge population—that design quickly gets messy though in terms of engineering and in terms of analyzing its security against various attacks.

8 Just watch the bridge authority's reachability tests

You don't actually need to break in to the bridge authority. Instead, you can just monitor its network connection: it will periodically test reachability of each bridge it knows, in order to let the bridgedb service know which addresses to give out.

[13] https://trac.torproject.org/projects/tor/ticket/4185
[14] http://dl.dropbox.com/u/37735/index.html
[15] https://gitweb.torproject.org/torspec.git/blob/HEAD:/proposals/187-allow-client-auth.txt
[16] https://svn.torproject.org/svn/projects/design-paper/blocking.html#tth_sEc5.2
[17] https://gitweb.torproject.org/bridgedb.git/tree

We could do these reachability tests through Tor, so watching the bridge authority doesn't tell you anything about what it's testing. But that just shifts the risk onto the rest of the relays, such that an adversary who runs or monitors a sample of relays gets to learn about a corresponding sample of bridges.

One option is to decentralize the testing such that monitoring a single location doesn't give you the whole bridge list. But how exactly to distribute it, and onto what, is messy from both the operational and research angles. Another option would be for the bridges themselves to ramp up the frequency of their reachability tests (they currently self-test for reachability before publishing, to give quick feedback to their operator if they're misconfigured). Then the bridges can just anonymously publish an authenticated "still here" message once an hour, so (assuming they all tell the truth) the bridge authority never has to do any testing. But this self-testing also allows an enumeration attack, since we build a circuit to a random relay and then try to extend back to our bridge address! Maybe bridges should be asking their guards to do the self-testing—once they have guards, that is?

These questions are related to the question of learning whether a bridge has been blocked in a given country[18]. More on that in a future report.

9 Watch your firewall and DPI for Tor flows

While the above attacks have to do with recognizing or inducing bridge-specific behavior, another class of attacks is just to buy some fancy Deep Packet Inspection gear and have it look for, say, characteristics of the SSL certificates or handshakes that make Tor flows stand out from "normal" SSL flows. Iran has used this strategy[19] to block Tor twice, and it lets them block bridges for free. The attack is most effective if you have a large and diverse population of Tor users behind your firewall, since you'll only be able to learn about bridges that your users try to use.

We can fix the issue by making Tor's handshake more like a normal SSL handshake[20], but I wonder if that's really a battle we can ever win. The better answer is to encourage a proliferation of modular Tor transports[21], like obfsproxy[22], and get the rest of the research community interested in designing tool-neutral transports that blend in better.

10 Zig-zag between bridges and users

Start with a set of known bridge addresses. Watch your firewall to see who connects to those bridges. Then watch those users, and see what other addresses they connect to. Wash, rinse, repeat.

As above, this attack only works well when you have a large population of Tor users behind your firewall. It also requires some more engineering work to be able to trace source addresses

[18] https://svn.torproject.org/svn/projects/design-paper/blocking.html#subsec:geoip

[19] https://blog.torproject.org/blog/iran-blocks-tor-tor-releases-same-day-fix

[20] https://gitweb.torproject.org/torspec.git/blob/HEAD:/proposals/179-TLS-cert-and-parameter-normalization.txt

[21] https://gitweb.torproject.org/torspec.git/blob/HEAD:/proposals/180-pluggable-transport.txt

[22] https://gitweb.torproject.org/obfsproxy.git/blob/HEAD:/doc/tor-obfs-howto.txt

in addition to destination addresses. But I'd be surprised if some major government firewalls don't have this capability already.

The solution here probably involves partitioning bridge addresses into cells[23], such that zig-zagging from users to bridges only gives you a bounded set of bridges (and a bounded set of users, for that matter). That approach will require some changes in our bridgedb design[24] though. Currently when a user requests some bridge addresses, bridgedb maps the user's "address" (IP address, gmail account name, or whatever) into a point in the keyspace (using consistent hashing[25]), and the answers are the k successors of that point in the ring (using DHT[26] terminology).

Dan Boneh suggested an alternate approach where we do keyed hashes[27] of the user's address and all the bridge fingerprints, and return all bridges whose hashed fingerprints match the user's hash in the first b bits. The result is that users would tend to get clustered by the bridges they know. That feature limits the damage from the zig-zag attack, but does it increase the risks in some distribution strategies? I already worry that bridge distribution strategies based on social networks will result in clusters of socially related users using the same bridges, meaning the attacker can reconstruct the social network. If we isolate socially related users in the same partition, do we magnify that problem? This approach also needs more research work to make it scale such that we can always return about k results, even as the address pool grows, and without reintroducing zig-zag vulnerabilities.

11 . . . What did I miss?

References

[1] Roger Dingledine. Strategies for getting more bridge addresses. Technical Report 2011-05-001, The Tor Project, May 2011. https://research.torproject.org/techreports/strategies-getting-more-bridge-addresses-2011-05-13.pdf.

[2] Roger Dingledine and Nick Mathewson. Design of a blocking-resistant anonymity system. Technical Report 2006-1, The Tor Project, November 2006. http://freehaven.net/anonbib/#tor-blocking.

[3] Ceki Gülcü and Gene Tsudik. Mixing E-mail with Babel. In *Proceedings of the Network and Distributed Security Symposium - NDSS '96*, pages 2–16. IEEE, February 1996. http://freehaven.net/anonbib/#babel.

[4] Amir Houmansadr, Giang T. K. Nguyen, Matthew Caesar, and Nikita Borisov. Cirripede: Circumvention infrastructure using router redirection with plausible deniability. In *Proceedings of the 18th ACM conference on Computer and Communications Security (CCS 2011)*, October 2011. http://freehaven.net/anonbib/#ccs2011-cirripede.

[23] http://en.wikipedia.org/wiki/Clandestine_cell_system
[24] https://gitweb.torproject.org/bridgedb.git/blob/HEAD:/bridge-db-spec.txt
[25] http://en.wikipedia.org/wiki/Consistent_hashing
[26] http://en.wikipedia.org/wiki/Chord_%28DHT%29
[27] http://en.wikipedia.org/wiki/HMAC

[5] Josh Karlin, Daniel Ellard, Alden W. Jackson, Christine E. Jones, Greg Lauer, David P. Mankins, and W. Timothy Strayer. Decoy routing: Toward unblockable internet communication. In *Proceedings of the USENIX Workshop on Free and Open Communications on the Internet (FOCI 2011)*, August 2011. http://freehaven.net/anonbib/#foci11-decoy.

[6] Jon McLachlan and Nicholas Hopper. On the risks of serving whenever you surf: Vulnerabilities in Tor's blocking resistance design. In *Proceedings of the Workshop on Privacy in the Electronic Society (WPES 2009)*. ACM, November 2009. http://freehaven.net/anonbib/#wpes09-bridge-attack.

[7] Rob Smits, Divam Jain, Sarah Pidcock, Ian Goldberg, and Urs Hengartner. Bridgespa: Improving Tor bridges with single packet authorization. In *Proceedings of the Workshop on Privacy in the Electronic Society (WPES 2011)*. ACM, October 2011. http://freehaven.net/anonbib/#wpes11-bridgespa.

[8] Eugene Y. Vasserman, Rob Jansen, James Tyra, Nicholas Hopper, and Yongdae Kim. Membership-concealing overlay networks. In Ehab Al-Shaer, Somesh Jha, and Angelos D. Keromytis, editors, *Proceedings of the 2009 ACM Conference on Computer and Communications Security, CCS 2009, Chicago, Illinois, USA, November 9-13, 2009*, pages 390–399. ACM, 2009. http://freehaven.net/anonbib/#DBLP:conf:ccs:VassermanJTHK09.

[9] Eric Wustrow, Scott Wolchok, Ian Goldberg, and J. Alex Halderman. Telex: Anticensorship in the network infrastructure. In *Proceedings of the 20th USENIX Security Symposium*, August 2011. http://freehaven.net/anonbib/#usenix11-telex.