# Research problems: Ten ways to discover Tor bridges

Posted October 31st, 2011 by arma

While we're exploring smarter ways of getting more bridge addresses (https://blog.torproject.org/blog/strategies-getting-more-bridge-addresses), 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 (https://www.torproject.org/docs/bridges) (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 (https://bridges.torproject.org/) 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 (https://www.torproject.org/docs/bridges#FindingMore) 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.

**#2: Run a non-guard non-exit relay and look for connections from non-relays.**

Normal clients use guard nodes (https://www.torproject.org/docs/faq#EntryGuards) 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 (http://www.cs.uml.edu/~xinwenfu/paper/Bridge.pdf) 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 (http://freehaven.net/anonbib/#babel) 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 (https://gitweb.torproject.org/torspec.git/blob/HEAD:/proposals/188-bridge-guards.txt) for details.

[Edit: 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!]

**#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" (https://trac.torproject.org/projects/tor/ticket/4115), "bridges didn't use a CREATE_FAST cell for the first hop of their own circuits like clients would have" (https://trac.torproject.org/projects/tor/ticket/4124), "bridges didn't reject CREATE and CREATE_FAST cells on connections they had initiated like clients would have" (http://archives.seul.org/tor/commits/Oct-2011/msg01037.html), and "bridges distinguish themselves in their NETINFO cell" (https://trac.torproject.org/projects/tor/ticket/4348).

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: 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 in Membership-concealing overlay networks (http://freehaven.net/anonbib/#DBLP:conf/ccs/VassermanJTHK09) and by Jon McLachlan in On the risks of serving whenever you surf: Vulnerabilities in Tor's blocking resistance design (http://freehaven.net/anonbib/#wpes09-bridge-attack), 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" (http://archives.seul.org/tor/relays/Apr-2010/msg00078.html) 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 (http://freehaven.net/anonbib/#tor-blocking).

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 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 (https://trac.torproject.org/projects/tor/ticket/4185) 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 (http://dl.dropbox.com/u/37735/index.html) that tunnels your traffic to the Tor relay once she's presented the right password. Or Tor could handle that authentication itself (https://gitweb.torproject.org/torspec.git/blob/HEAD:/proposals/187-allow-client-auth.txt). BridgeSPA: Improving Tor Bridges with Single Packet Authorization (http://freehaven.net/anonbib/#wpes11-bridgespa) 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 (http://freehaven.net/anonbib/#usenix11-telex), Decoy Routing (http://freehaven.net/anonbib/#foci11-decoy), or Cirripede (http://freehaven.net/anonbib/#ccs2011-cirripede). 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 (https://svn.torproject.org/svn/projects/design-paper/blocking.html#tth_sEc5.2) aggregates the list of bridges and periodically sends it to the bridgedb service (https://gitweb.torproject.org/bridgedb.git/tree) 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.

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 (https://svn.torproject.org/svn/projects/design-paper/blocking.html#subsec:geoip). More on that in a future blog post.

**#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 (https://blog.torproject.org/blog/iran-blocks-tor-tor-releases-same-day-fix) 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 (https://gitweb.torproject.org/torspec.git/blob/HEAD:/proposals/179-TLS-cert-and-parameter-normalization.txt), 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 (https://gitweb.torproject.org/torspec.git/blob/HEAD:/proposals/180-pluggable-transport.txt), like obfsproxy (https://gitweb.torproject.org/obfsproxy.git/blob/HEAD:/doc/tor-obfs-howto.txt), 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 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 (http://en.wikipedia.org/wiki/Clandestine_cell_system), 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 (https://gitweb.torproject.org/bridgedb.git/blob/HEAD:/bridge-db-spec.txt) 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 (http://en.wikipedia.org/wiki/Consistent_hashing)), and the answers are the k successors of that point in the ring (using DHT (http://en.wikipedia.org/wiki/Chord_%28DHT%29) terminology).

Dan Boneh suggested an alternate approach where we do keyed hashes (http://en.wikipedia.org/wiki/HMAC) 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?**

## Comment viewing options

Threaded list - expanded | Date - oldest first | 300 comments per page | Save settings

Select your preferred way to display the comments and click "Save settings" to activate your changes.

On October 31st, 2011 Anonymous said:

11. convince tor dev to open list ip addresses. install back door. use gun, kidnap, torture. all skill the US has for world. see hushmail. see jap. see hidemyass.

On October 31st, 2011 arma said:

Ah. Yes, I count these operational security concerns as part of #7. The suggested "fixes" in that section apply to many of the issues you raise as well. More suggested fixes would be great.

So far our faq entry (https://www.torproject.org/docs/faq#Backdoor) mentioning smart lawyers has been excellent at dissuading legal attempts. And we're happy that nobody has tried the extralegal approaches.

During the Q&A session for my 23c3 talk (https://media.torproject.org/video/23C3-1444-en-tor_and_china.m4v) in 2006 introducing the bridge design, a nice man with a thick Russian accent raised the following question: "Then they'll kill you." I stick with my answer at the time: "My main goal is to make sure that the software exists, and at that point maybe I'm not the top target anymore." That's the great thing about open source, open designs, and open communities.

On November 1st, 2011 Anonymous said:

why would the tor developers program a backdoor into tor? I don't get it. This wouldn't make tor more secure would it? How?

On November 2nd, 2011 Anonymous said:

WTH .... there is a backdoor at the TorButton... 0_o

http://pastebin.com/hquN9kg5

3) We secretly contacted our friends at The Mozilla Foundation™, Developers of Firefox™, for them to authorize a developer signer certificate for "The Honey Pawt", a TorButton that we Anon created to funnel all ORIGINATING traffic to our forensic logger

A group of bad guys got help from Mozilla Foundantion to put a backdoor at the TorButton. They say it was used only for 24 hours, but ..... is that real? Will we be able to trust at Tor again?

On November 2nd, 2011 arma said:

No backdoor in Torbutton. What happened here (as far as I can tell) is that some person wrote a Firefox extension and convinced some other people to install it. Actual Torbutton was nowhere in the picture.

Have we been mentioning that you should always check the PGP signatures of Tor software before you install it?

On November 3rd, 2011 Anonymous said:

This Firefox extension had a trusted signature from Mozilla Foundation.

On November 4th, 2011 arma said:
Yeah? Can you show us a copy?

On November 13th, 2011 Anonymous said:
The Mozilla Foundation doesn't sign Firefox extensions...

On November 1st, 2011 Anonymous said:
Just how would you be able to discover tor bridges by "see hushmail"?

On November 6th, 2011 Anonymous said:
I think he referred to the fact that Hushmail gave out data.

On November 1st, 2011 Anonymous said:
Could be part of 1....

Convince bridge operators or bridge users to tell 'em which addresses the provide/use. e.g. by making them believe that they are censored users, too.

On November 1st, 2011 Anonymous said:
Sounds like several of these attacks could be mitigated by running a relay and a bridge on the same box on two different IPs and using the EntryNodes configuration option on the bridge to ensure the local relay is always the first hop. Shame about the wasted processing passing packets back and forth between the two Tor instances though. It'd be nice if at some point Tor when running as a relay supported an option to act as a bridge on a different IP as well, just never originating connections from that IP. Even better if it supported acting as a bridge on multiple IPs.

-Pascal

On November 6th, 2011 Anonymous said:
We defiantly need an according feature/config option!

On November 2nd, 2011 Anonymous said:
چگونه نرم افزار تور را دانلود کنیم؟

On December 3rd, 2011 Anonymous said:
دوست عزیز اینجا اگه فارسي بنویسي کسي جوابتو نمیده
از اینجا دانلود کن:

https://www.torproject.org/dist/torbrowser/tor-browser-2.2.34-3_en-US.exe