bridge distribution

Strategies for getting more bridge addresses (/blog/strategies-getting-more-bridge-addresses)

Posted May 13th, 2011 by arma

We need more bridges. When I first envisioned the bridge address arms race, I said "We need to get lots of bridges, so the adversary can't learn them all." That was the wrong statement to make. In retrospect, the correct statement was: "We need the rate of new bridge addresses to exceed the rate that the adversary can block them."

For background, bridge relays (https://www.torproject.org/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. We deployed them several years ago (https://svn.torproject.org/svn/projects/design-paper/blocking.html) in anticipation of the upcoming arms race, and they worked great in their first showing in 2009 (https://blog.torproject.org/blog/picturing-tor-censorship-china) . But since then, China has learned and blocked (https://metrics.torproject.org/users.html?graph=bridge-users&start=2010-01-01&end=2011-04-09&country=cn#bridge-users) most of the bridges we give out through public (https and gmail) distribution channels.

One piece of the puzzle is smarter bridge distribution mechanisms (https://blog.torproject.org/blog/bridge-distribution-strategies) (plus see this post (http://archives.seul.org/or/dev/Dec-2009/msg00000.html) for more thoughts) — right now we're getting 8000 mails a day from gmail asking for bridges from a pool of less than a thousand (https://metrics.torproject.org/network.html#networksize) . The distribution strategy that works best right now is ad hoc distribution via social connections. But even if we come up with brilliant new distribution mechanisms, we simply need more addresses to work with. How can we get them? Here are five strategies.

Approach one: Make it easy to become a bridge using the Vidalia interface. This approach was our first try at getting more bridges: click on "Sharing", then "Help censored users reach the Tor network". Easy to do, and lots of people have done it. But lots here is thousands, not hundreds of thousands. Nobody knows that they should click it or why.

Approach two: Bridge-by-default bundles. People who want to help out can now simply download and run our bridge-by-default (https://blog.torproject.org/blog/windows-bridge-default-bundle) bundle, and poof they're a bridge. There's a lot of flexibility here. For example, we could provide a customized bridge-by-default bundle for a Chinese human rights NGO that publishes your bridge address directly to them; then they give out the bridge addresses from their volunteers through their own social network. I think this strategy will be most effective when combined with targeted advocacy, that is, after a given community is convinced that they want to help and want to know how they can best help.

Approach three: Fast, stable, reachable Tor clients auto-promote themselves. Tor clients can monitor their own stability, performance, and reachability, and the best clients can opt to become bridges automatically. We can tune the thresholds ("how fast, how stable") in the directory consensus, to tailor how many clients promote themselves in response to demand. Read the proposal (https://gitweb.torproject.org/torspec.git/blob/HEAD:/proposals/175-automatic-node-promotion.txt) for more details. In theory this approach could provide us with many tens of thousands of bridges in a wide array of locations — and we're drawing from a pool of people who already have other reasons to download Tor. Downsides include a) there's quite a bit of coding work remaining before we can launch it, b) there are certainly situations where we shouldn't turn a Tor user into a bridge, which

means we need to sort out some smart way to interact with the user and get permission, and c) these users don't actually change addresses as often as we might want, so we're still in the "gets lots of bridges" mindset rather than the "increase the rate of new bridge addresses" mindset.

Approach four: Redirect entire address blocks into the Tor network. There's no reason the bridge and its address need to run in the same location, and it's really addresses that are the critical resource here. If we get our friends at various ISPs to redirect some spare /16 networks our way, we'd have a lot more addresses to play with, and more importantly, we can control the churn of these addresses. Past experience with some censors shows that they work hard to unblock addresses that are no longer acting as proxies. If we light up only a tiny fraction of the IP space at a time, how long until they block all of it? How much does the presence of other services on the address block make them hesitate? I want to find out. The end game here is for Comcast to give us a few random IP addresses from each of their /24 networks. All the code on the Tor bridge side already works here, so the next steps are a) figure out how to teach an ISP's router to redirect some of its address space to us, and then b) sign up some people who have a good social network of users who need bridges, and get them to play that arms race more actively.

Approach five: More generic proxies. Originally I had thought that the extra layer of encryption and authentication from a bridge was a crucial piece of keeping the user (call her Alice) safe. But I'm increasingly thinking that the security properties she gets from a Tor circuit (anonymity/privacy) can be separated from the security properties she gets from the bridge (reachability, and optionally obfuscation). That is, as long as Alice can fetch the signed Tor network consensus (https://torproject.org/docs/faq#KeyManagement) and verify the keys of each of the public relays in her path, it doesn't matter so much that the bridge gets to see her traffic. Attacks by the bridge are no more effective than attacks by a local network adversary, which by design is not much. Now, this conclusion is premature — adding a bridge into the picture means there's a new observation point in addition to the local network adversary, and observation points are exactly what the attacker needs to correlate traffic flows and break Tor's anonymity, But on the flip side, right now bridges already get to act as these observational points, and the extra layer of encryption they add doesn't seem to help Alice any. So it's too early to say (https://trac.torproject.org/projects/tor/ticket/2764) that a socks or https proxy is just as safe as a bridge (assuming you use a full Tor circuit in either case), but I'm optimistic that these more generic proxies have a role to play.

If we go this route, then rather than needing volunteers to run a whole Tor (which is especially cumbersome because it needs libraries like OpenSSL), people could run socks proxies on a much broader set of platforms. For example, they should be easy to add into Orbot (https://www.torproject.org/docs/android) (our Tor package for Android) or into Seattle (https://seattle.cs.washington.edu/html/) (an overlay network by UW researchers that restricts applications to a safe subset of Python). We could even imagine setting up a website where volunteers visit a given page and it runs a Flash or Java applet socks proxy, lending their address to the bridge pool while their browser is open. There are some gotchas to work through, such as a) needing to sign the applets so they have the required network permissions, b) figuring out how to get around the fact that it seems hard to allow connections from the Internet to a flash plugin, and c) needing to program the socks proxy with a Tor bridge or relay address so the user doesn't have to ask for it (after all, socks handshakes are unencrypted and it wouldn't do to let the adversary watch Alice ask for an IP address that's known to be associated with Tor). This 'flash proxy' idea was developed in collaboration with Dan Boneh at Stanford, and they are currently designing and building it — stay tuned.