Better guard rotation parameters

Roger Dingledine
arma@torproject.org

Tor Tech Report 2011-08-001
August 20, 2011

Tor's entry guard design[1] protects users in a variety of ways.

First, they protect against the "predecessor attack" [3]: if you choose new relays for each circuit, eventually an attacker who runs a few relays will be your first and last hop. With entry guards, the risk of end-to-end correlation for any given circuit is the same, but the cumulative risk for all your circuits over time is capped.

Second, they help to protect against the "denial of service as denial of anonymity" attack [1], where an attacker who runs quite a few relays fails any circuit that he's a part of and that he can't win against, forcing Alice (the user) to generate more circuits and thus increasing the overall chance that the attacker wins. Entry guards greatly reduce the risk, since Alice will never choose outside of a few nodes for her first hop.

Third, entry guards raise the startup cost to an adversary who runs relays in order to trace users. Without entry guards, the attacker can sign up some relays and immediately start having chances to observe Alice's circuits. With them, new adversarial relays won't have the Guard flag so won't be chosen as the first hop of any circuit; and even once they earn the Guard flag, users who have already chosen guards won't switch away from their current guards for quite a while.

But how long exactly? The first research question here examines vulnerability due to natural churn of entry guards. Consider an adversary who runs one entry guard with advertised capacity C. Alice maintains an ordered list of guards (chosen at random, weighted by advertised speed, from the whole list of possible guards). For each circuit, she chooses her first hop (uniformly at random) from the first G guards from her list that are currently running (appending a new guard to the list as needed, but going back to earlier guards if they reappear). How long until Alice adds the adversary's node to her guard list? You can use the uptime data from the directory archives[2], either to build a model for guard churn or just to use the churn data directly. Assuming $G = 3$, how do the results vary by C? What's the variance?

Research question two: consider intentional churn due to load balancing too. Alice actually discards each guard between 30 and 60 days (uniformly chosen) after she first picks it. This intentional turnover prevents long-running guards from accumulating more and more users and getting overloaded. Another way of looking at it is that it shifts load to new guards so

[1] https://www.torproject.org/docs/faq#EntryGuards
[2] https://metrics.torproject.org/data.html

we can make better use of them. How much does this additional churn factor impact your answer from step one above? Or asked a different way, what fraction of Alice's vulnerability to the adversary's entry guard comes from natural churn, and what fraction from the proactive expiration? How does the answer change for different expiry intervals (e.g. between 10 and 30 days, or between 60 and 90)?

Research question three: how do these answers change as we vary G? Right now we choose from among the first three guards, to reduce the variance of expected user performance—if we always picked the first guard on the list, and some users picked a low-capacity or highly-congested relay, none of that user's circuits would perform well. That said, if choosing $G = 1$ is a huge win for security, we should work on other ways to reduce variance.

Research question four: how would these answers change if we make the cutoffs for getting the Guard flag more liberal, and/or change how we choose what nodes become guards? After all, Tor's anonymity is based on the diversity of entry and exit points [2], and while it may be tough to get around exit relay scarcity, my theory is that our artificial entry point scarcity (because our requirements are overly strict) is needlessly hurting the anonymity Tor can offer.

Our current algorithm for guard selection has three requirements:

1. The relay needs to have first appeared longer ago than 12.5% of the relays, or 8 days ago, whichever is shorter.

2. The relay needs to advertise at least the median bandwidth in the network, or 250KB/s, whichever is smaller.

3. The relay needs to have at least the median weighted-fractional-uptime of relays in the network, or 98% WFU, whichever is smaller. (For WFU, the clock starts ticking when we first hear about the relay; we track the percentage of that time the relay has been up, discounting values by 95% every 12 hours.

Today's guard cutoffs in practice are "was first sighted at least 8 days ago, advertises 100KB/s of bandwidth, and has 98% WFU."

Consider two relays, A and B. Relay A first appeared 30 days ago, disappeared for a week, and has been consistently up since then. It has a WFU (after decay, give or take a fencepost) of 786460 seconds / 824195 = 95.4%, so it's not a guard. Relay B appeared two weeks ago and has been up since then. Its WFU is 658517 / 658517 = 100%, so it gets the Guard flag—even though it has less uptime, and even less weighted uptime. Based on the answers to the first three research questions, which relay would serve Alice best as a guard?

The big-picture tradeoff to explore is: what algorithm should we use to assign Guard flags such that a) we assign the flag to as many relays as possible, yet b) we minimize the chance that Alice will use the adversary's node as a guard?

References

[1] Nikita Borisov, George Danezis, Prateek Mittal, and Parisa Tabriz. Denial of service or denial of security? How attacks on reliability can compromise anonymity. In *Proceedings of CCS 2007*, October 2007. http://freehaven.net/anonbib/#ccs07-doa.

[2] Roger Dingledine. Measuring the safety of the Tor network. Technical Report 2011-02-001, The Tor Project, February 2011. https://research.torproject.org/techreports/measuring-safety-tor-network.pdf.

[3] Matthew Wright, Micah Adler, Brian Neil Levine, and Clay Shields. The Predecessor Attack: An Analysis of a Threat to Anonymous Communications Systems. *ACM Transactions on Information and System Security (TISSEC)*, 4(7):489–522, November 2004. http://freehaven.net/anonbib/#Wright:2004.