Securing Anonymous Communication Channels under the Selective DoS Attack

Anupam Das and Nikita Borisov

University of Illinois at Urbana Champaign, USA
{das17,nikita}@illinois.edu

Abstract. Anonymous communication systems are subject to selective denial-of-service (DoS) attacks. Selective DoS attacks lower anonymity as they force paths to be rebuilt multiple times to ensure delivery, which increases the opportunity for more attack. We present a detection algorithm that filters out compromised communication channels for one of the most widely used anonymity networks, Tor. Our detection algorithm uses two levels of probing to filter out potentially compromised tunnels. We probabilistically analyze our detection algorithm and show its robustness against selective DoS attacks through simulation. We also analyze the overhead of our algorithm and show that we can achieve better security guarantee than the conventional Tor path selection algorithm, while adding only approximately 5% bandwidth overhead to the Tor network. Finally, we validate our design with experiments using the live Tor network.

Keywords: Anonymity, Tor network, denial of service (DoS) attack.

1 Introduction

Anonymous communication was first introduced in 1981 with Chaum's seminal paper on "Untraceable electronic mail, return addresses, and digital pseudonyms" [14]. Since then, many researchers have concentrated on building, analyzing and attacking anonymous communication systems such as Tor [18], I2P [4], Freenet [3]. In this paper we concentrate on Tor [18], one of the most widely used low-latency anonymity networks, which conceals users' identities and activities from surveillance and traffic analysis. Tor provides confidentiality and privacy to users of various types ranging from ordinary individuals to business personnel, journalists, government employees and even military personnel [8]. Currently, Tor has over 3000 relays all around the world and it is used by hundreds of thousands of people every day [7, 19, 21].

Users' identities, however, can become exposed when multiple relays are compromised. By default, Tor uses three relays and an attacker who can gain control of the entry and exit relays is capable of compromising user identity using timing analysis [20, 25]. Moreover, malicious nodes can perform a selective denial-of-service (DoS) attack [12, 13] where malicious relays drop circuits if they cannot compromise them. This increases the probability of such a path being built and as a result lowers anonymity. The selective DoS attack is particularly useful for an attacker with moderate resources; as one potential example, the Dutch ministry of Justice and Security proposed passing a law which would enable the law enforcement office to launch any form of attack (i.e., selective DoS could one form of attack) on any system in order to gather evidence

[1]. So some form of mechanism is needed to ensure secure path construction in the presence of compromised/controlled relays.

Danner et al. [15] showed that it is possible to identify relays mounting selective DoS using exhaustive probing. The intent is to periodically carry out these probes and blacklist the misbehaving relays; however, the total number of probes required is prohibitive–3 times the size of the network at a minimum, and many more (typically retrying each probe 10 times) to account for non-malicious failures. So, their approach seems practical for a centralized design, but we wanted to create a local mechanism to defend against selective DoS. By using probabilistic inference, we can make do with orders of magnitude fewer probes and thus our approach is practical to be executed at each individual client. We perform simulations and real world experiments to show the effectiveness of our detection mechanism.

2 Background

2.1 Tor Network

Tor [18] is an anonymous communication network that allows users to make TCP connections to Internet sites without revealing their identity to the destination or third-party observers. We briefly explain the main components of Tor relevant to this work. To initiate an anonymous TCP connection, a Tor user constructs a *circuit* (also known as a tunnel or path) comprised of several Tor *relays*. The relays form a forwarding chain that sends traffic from the user to the destination, and vice versa. Circuits typically involve three relays: the *entry*, *middle*, and *exit*. The traffic contents are protected by a layered encryption scheme [24], where each relay peels off a layer while forwarding. As a result, any individual router cannot reconstruct the whole circuit path and link the source to the destination. The relays in a circuit are chosen following specific constraints [17]. Each user selects a small, fixed number (currently 3) of entry relays that are used for all circuits. These relays are called *guard relays* [23, 28]; their use is designed to defend from the predecessor attack [29]. To choose the exit relay, the user picks from among those relays that have an exit policy compatible with the desired destination. After these constraints, the relays for each position are chosen randomly, weighted by their bandwidth [1].

Tor aims to provide low-latency traffic forwarding for its users. As a result, as traffic is forwarded along a circuit, timing patterns remain discernible, and an attacker who observes two different relays can use timing analysis to determine whether they are participating in the same circuit [20, 25, 27, 30]. So, to link a Tor user to a destination, it suffices to observe the entry and the exit relays of a circuit. Standard security analysis of Tor [18, 27] shows that if t is the fraction of relays that are observed, an adversary will be able to violate anonymity on t^2 of all of the circuits. Note that, due to bandwidth-weighted path selection in Tor, t is best thought of as the fraction of total Tor *bandwidth*

[1] Guard and/or exit relays are underweighted when chosen as middle node to improve the overall balancing of load, however, these details are not key to our discussion.

that belongs to relays under observation[2]. The security of Tor, therefore, relies on the assumption that a typical adversary will not be able to observe a significant fraction of Tor relays. The easiest way to observe relay traffic is to run your own relays as there is no barrier to entry other than Internet connectivity and sufficient bandwidth.

2.2 Selective Denial of Service in Tor

An adversary who controls a Tor relay can perform a number of active attacks to increase the odds of compromise [12, 13]. One approach, which is the focus of this work, is *selective denial of service* [13]. A compromised relay that participates in a circuit can easily check whether both the entry and exit relays are under observation. If this is not the case, the relay can "break" the circuit by refusing to forward traffic. This will cause the user to reformulate a new circuit for the connection, giving the adversary another chance to compromise the circuit. A simple analysis shows that this increases the overall fraction of compromised circuits to $\frac{t^2}{t^2+(1-t)^3} > t^2$, because only circuits with compromised entry and exit relays (t^2) or circuits with no compromised relays ($(1-t)^3$) will be functional, and out of those t^2 will be compromised. E.g., if $t = 0.2$, selective DoS increases the fraction of compromised circuits by 81.25%. The use of guard nodes changes the analysis somewhat; compromised guards can amplify the effect of selective DoS. Bauer et al. [11] showed that deploying a moderate number of inexpensive[3] middle-only relays can boost the effect of selective DoS attack.

2.3 Threat Model

In our threat model we assume that a small fraction (typically 20%) of the Tor relays are compromised and for each user g (where $g \in \{0, 1/3, 2/3, 1\}$) fraction of the guard nodes are compromised. Compromised relays carry out selective DoS attack, however they may choose to perform probabilistic dropping where a compromised relay terminates a certain fraction of all circuits that it cannot compromise. Finally, we assume that probes are indistinguishable from real user traffic[4].

3 Detection Algorithm

Our algorithm is built on the fundamental assumption of the Tor security model that a relatively small fraction of all relays are compromised. The algorithm works in two phases and runs periodically. Table 1 summarizes the different parameters used for our detection algorithm.

[2] To be more precise, the correct fraction would be $t_g \cdot t_e$, where t_g and t_e are the fractions of the guard and exit bandwidth under observation, respectively. For simplicity of presentation, we will assume $t_g = t_e = t_m = t$ in the rest of the paper.

[3] Middle-only nodes do not have to fulfill stronger commitments (e.g., minimum bandwidth, minimum uptime, legal issues related to exit policies) that guard and exit nodes have to fulfill.

[4] To mask probes from actual user traffic we propose downloading popular web pages listed by Alexa [5]. More discussion is available in our technical report [16].

Table 1: Parameters Used

Setting	Parameter	Description
Environmental	t	Fraction of relays compromised
	g	Fraction of compromised guards per user
	d	Random drop rate by compromised nodes
Tunable	N	# of working Tor circuits created in 1st phase
	K	# of probes used per circuit in 2nd phase
	θ	Threshold for classifying circuit

3.1 First Phase

Under active use, Tor will switch to a new circuit every 10 minutes, meaning that we need 6 non-compromised circuits every hour. So in the first phase of our detection algorithm we iteratively generate a random Tor circuit and test its functionality by retrieving a random web file through the circuit. If it fails we discard the circuit and try a new circuit. We stop when we have N (we can calculate the value of N using equation: $N = \left\lceil 6 \times \frac{gt+(1-g)(1-t)^2}{(1-g)(1-t)^2} \right\rceil$. Considering worst case scenario we can set the value of N to 10) working circuits. If an adversary is carrying out selective DoS attack then after the first phase we should have a set of circuits of form either CXC or HHH, where C denotes a compromised relay, H denotes an honest one, and X is a relay of any type.

3.2 Second Phase

In the second phase, we examine each of the circuits passing the first phase (we will call these circuits as *potential* circuits) as follows:

- We randomly pick $K(1 \leq K < N)$ other circuits (we will call them as *candidate* circuits) out of the list of potential circuits.
- For each of the K *candidate* circuits, we change the exit relay of the *potential* circuit being evaluated with the exit relay of the candidate circuit and choose a random middle relay from the candidate set. We then test the functionality of the new circuit by performing a web retrieval through it. If, out of these K probes, θ or more succeed, we consider the evaluated circuit to be honest; otherwise, we consider it to be compromised.

Note that under selective DoS, if we change the exit relay of a compromised circuit with that of an honest circuit, we will get a circuit where the entry is compromised while the exit is honest and hence the file retrieval should fail. On the other hand, if both the evaluated and candidate circuits are honest or compromised, the probe will succeed. We expect more success for an honest circuit, since most of the potential circuits are honest; we use θ as a threshold for distinguishing between the two circuit types. At the end of the second phase, we will have some number of potentially honest circuits. This collection of circuits is then used for making real anonymous connections.

4 Security Analysis

4.1 False Error Rates

We first evaluate the false-negative (FN) and false-positive (FP) errors of our algorithm under selective DoS strategy. False-negative (FN) error, i.e., the fraction of compromised circuits that pass our detection algorithm, depends upon the number of compromised ($n(\mathrm{CXC})$) and honest ($n(\mathrm{HHH})$) circuits that pass the first phase (we can compute such probabilities using Binomial distributions). Now, a false-negative error occurs when a compromised circuit is paired with at least θ other compromised candidate circuits in the second phase. So we can use hypergeometric distribution to calculate $\Pr(\mathrm{FN})$ (similarly we can compute $\Pr(\mathrm{FP})$). Detailed derivation of these false errors can be found in our technical report [16]. Transient network failure can directly influence the success rate of our probing, so it can affect both FN and FP errors. We discuss the impact of such failures in our technical report [16] as well.

4.2 Tuning Parameters

Security vs Overhead: Our detection algorithm has two tunable parameters (K, θ) (see Table 1 for description). For tuning purpose, we introduce two evaluation metrics: *security* (ψ) and *overhead* (η). We then tune K and θ in terms of these evaluation metrics. We define *security* as the probability of not choosing a compromised circuit for actual usage and *overhead* as the expected number of probes required for each usable circuit (by usable circuits we refer to circuits that are used for actual client traffic). We define ψ and η using the following functions (both of these metrics are approximations):

$$\psi = 1 - \frac{gt \times \Pr(\mathrm{FN})}{gt \times \Pr(\mathrm{FN}) + (1-g)(1-t)^2 \times (1 - \Pr(\mathrm{FP}))} \tag{1}$$

$$\eta = \frac{1 + \left[gt + (1-g)(1-t)^2\right] \times K}{gt \times \Pr(\mathrm{FN}) + (1-g)(1-t)^2 \times (1 - \Pr(\mathrm{FP}))} \tag{2}$$

Detailed derivations of these metrics can be found in our technical report [16]. We can then look at the distribution of ψ vs η for different values of (K, θ) and choose a (K, θ) pair that achieves satisfactory security guarantee at the cost of reasonable overhead.

5 Experimental Evaluation

5.1 Simulation Results

We implemented a simulator in C++ that emulates the basic functionality of Tor circuit construction and selective DoS attacks. We collected real Tor node information from the Tor network status page [10] and randomly tagged 20% ($t = 0.2$) of the bandwidth to be controlled by a compromised entity. To analyze the robustness and effectiveness of our detection algorithm we vary g ($0 \leq g \leq 1$) and d ($0 \leq d \leq 1$) in our simulations. Here, 100% drop rate refers to selective DoS and 0% drop means no dropping at all. Based on empirical results from our technical report [16], we set $K = 3$ and $\theta = 2$ in all the simulations. All simulation results are averaged over 100 runs with 95% confidence interval.

Robustness: First, we will look at the robustness of our detection algorithm in filtering out compromised circuits. For this purpose we evaluate the probability of selecting compromised circuits, $\Pr(\text{CXC})$[5], against different drop rates, d.

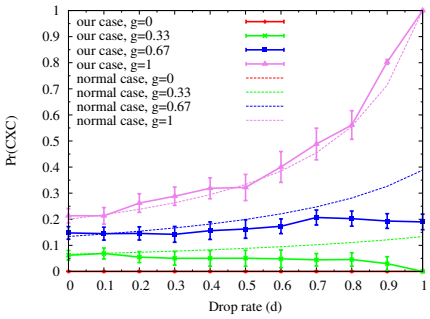

Fig. 1: Probability of selecting compromised circuits (CXC) for different drop rates d. In general $\Pr(\text{CXC})$ decreases as d rises compared to the conventional Tor network.

From Figure 1, we see that as drop rate d increases, the probability of selecting compromised circuits for our approach lowers compared to the conventional Tor circuit construction policy (indicated by the dashed lines). The main reason behind the decrease of $\Pr(\text{CXC})$ lies on the fact that as compromised nodes start to perform aggressive dropping, the pool of available circuits after the first phase quickly converges to the set $\{\text{CXC}, \text{HHH}\}$. This in turn lowers a compromised circuit's chance of selecting other compromised circuits as candidates in the second phase because honest circuits dominate over compromised circuits for $t = 0.2$.

5.2 Real-World Experiment

We carried out real-world experiments by introducing our own relays into the Tor network, all of which acted as compromised nodes. For this purpose we used 11 Emulab [2] machines, 10 of which were configured to act as Tor relays with a minimum bandwidth capacity of 20Kbps. Note that all our nodes belonged to the same /16 subnet, meaning that no user would (by default) choose two of our nodes in the same circuit. Moreover, to prevent other users from using our nodes as exit nodes, we configured our relays with a fixed exit policy (allowing connection to only specific destinations). All these measures were taken to respect the privacy of Tor users. To implement selective DoS, we take an approach similar to the one described by Bauer et al. [12]. We modified Tor source code version-*tor-0.2.2.35* and implemented our detection algorithm in the client side in Python (we used the Python version of Tor-Controller [6]).

Robustness: We first query the Tor directory server to retrieve a list of all available Tor relays and then consider only those routers which are flagged as *running*, *stable* and *valid*, because we want our routers to be alive and running during our experiments. We selected 40 Tor nodes (3 guards, 19 exits and 18 relays) at random with probability proportional to their bandwidth and added our own 10 nodes to this set to get a total of 50 nodes. Then we ran our experiments on this small set of Tor nodes where nodes were selected randomly. This choice results in about 20% of the nodes being compromised. To emulate user traffic, we retrieve random web files 100–300 KB in size. We set $K = 3, \theta = 2$ for our experiments. Table 2 summarizes our findings.

[5] $\Pr(\text{CXC}) = n(\text{CXC}) / [n(\text{HHH}) + n(\text{CXC}) + (1-d)n(\text{Others})]$

Overhead: Let us now estimate what kind of bandwidth overhead our mechanism would inflict on the real Tor network. Now on average a single usable circuit requires approximately 4 probes (one in the 1st phase and 3 in the 2nd phase). And since we are proposing to use popular web sites as probing destinations we can approximate the average probe size to be 300KB [26]. So the total traffic used by a single user every one hour is $(6 \times 3 \times 300 \times 4)KB\approx$ 21MB. Now, Tor's bandwidth capacity was found to be 3.21GB/s [7] during the month of September 2012. If we allow 5% of the bandwidth to be used for our detection algorithm then we can support approximately 28 000 simultaneous users per hour (i.e.,$\approx 672\,000$ user attempts daily which is comparable to $619\,696$, the peak number of daily Tor users for October 2012 [7]).

g	FN	FP	ψ	Security in Current Tor
0	0.0	0.0664	1.0	1.0
1/3	0.0	0.178	1.0	0.867
2/3	0.133	0.283	0.843	0.612
1	1.0	0.0	0.0	0.0

Table 2: Results from the Tor Network

From Table 2 we see that as g increases the security assurance provided by both our approach and the conventional Tor network goes down. However, for $g = 1/3, 2/3$ our approach shows significant improvement in filtering out compromised circuits.

6 Related Work

Borisov et al. [13] first showed that carrying out selective DoS could benefit an adversary to increase its chance of compromising anonymity for both high and low-latency anonymous communication systems. In fact, it was pointed out that with 20% compromised nodes in Salsa [22], the selective DoS attack results in 19.14% compromised tunnels compared to the conventional security analysis of 6.82% compromised tunnels.

Later on Danner et al. [15] proposed a detection algorithm for selective DoS attack on Tor. Their algorithm basically probes each individual Tor node in the network and they prove that this requires $O(n)$ probes to detect all compromised nodes in a Tor network comprising of n participants. For circuits of length 3, their algorithm requires $3n$ probes; however to handle transient network failures they propose to repeat each probe 10 times. Clearly, their approach seems only practical for a centralized design, but ours is a local mechanism to defend against selective DoS.

Recently, Mike Perry proposed a client-side accounting mechanism that tracks the circuit failure rate of each guard node used by a client [9]. The goal is to avoid malicious guards that deliberately fail circuits extending to non-colluding exit nodes. We take a more proactive approach to finding malicious circuits through probing instead of tracking actual circuit usage.

7 Conclusion

Anonymous communication systems like Tor are vulnerable to selective DoS attacks that considerably lower anonymity. Such attacks however, can be detected through probing. Our detection algorithm probes communication channels to filter out potentially compromised ones with high probability. We also show that adaptive adversaries

who choose to deny service probabilistically do not benefit from adopting such a strategy. Our experimental results demonstrate that our detection algorithm can effectively defend users against selective DoS attack.

References

[1] Dutch government proposes cyberattacks against... everyone., https://www.eff.org/deeplinks/2012/10/dutch-government-proposes-cyberattacks-against-everyone
[2] Emulab., https://www.emulab.net
[3] Freenet., https://freenetproject.org/
[4] I2p., http://www.i2p2.de/
[5] Top sites on the web. http://www.alexa.com/topsites
[6] Tor controller., https://svn.torproject.org/svn/blossom/trunk/TorCtl.py
[7] Tor metrics portal:. https://metrics.torproject.org/
[8] Tor project., https://www.torproject.org/
[9] Tor proposal-209, https://gitweb.torproject.org/user/mikeperry/torspec.git/blob/path-bias-tuning:/proposals/209-path-bias-tuning.txt
[10] Torstatus., http://torstatus.blutmagie.de/index.php
[11] Bauer, K., Juen, J., Borisov, N., Grunwald, D., Sicker, D., Mccoy, D.: On the optimal path length for Tor. In: 3rd Workshop on Hot Topics in Privacy Enhancing Technologies (2010)
[12] Bauer, K., McCoy, D., Grunwald, D., Kohno, T., Sicker, D.: Low-resource routing attacks against Tor. In: ACM Workshop on Privacy in Electronic Society. pp. 11–20 (2007)
[13] Borisov, N., Danezis, G., Mittal, P., Tabriz, P.: Denial of service or denial of security? In: 14th ACM conference on Computer and Communications Security. pp. 92–102 (2007)
[14] Chaum, D.L.: Untraceable electronic mail, return addresses, and digital pseudonyms. In: Communications of the ACM. vol. 24, pp. 84–90 (1981)
[15] Danner, N., Krizanc, D., Liberatore, M.: Detecting denial of service attacks in Tor. In: Financial Cryptography and Data Security. pp. 273–284 (2009)
[16] Das, A., Borisov, N.: Securing Tor tunnels under the selective-DoS attack. http://arxiv.org/abs/1107.3863
[17] Dingledine, R., Mathewson, N.: Tor path specification. https://gitweb.torproject.org/torspec.git/blob/HEAD:/path-spec.txt
[18] Dingledine, R., Mathewson, N., Syverson, P.: Tor: The second-generation onion router. In: 13th USENIX Security Symposium. pp. 303–320 (2004)
[19] Hahn, S., Loesing, K.: Privacy-preserving ways to estimate the number of Tor users (2010), https://metrics.torproject.org/papers/countingusers-2010-11-30.pdf
[20] Levine, B.N., Reiter, M.K., Wang, C., Wright, M.: Timing attacks in low-latency mix systems. In: Financial Cryptography and Data Security. pp. 251–265 (2004)
[21] Loesing, K.: Measuring the Tor network: Evaluation of client requests to the directories. Tech. rep. (2009), https://metrics.torproject.org/papers/directory-requests-2009-06-25.pdf
[22] Nambiar, A., Wright, M.: Salsa: a structured approach to large-scale anonymity. In: 13th ACM conference on Computer and Communications Security. pp. 17–26 (2006)
[23] Overlier, L., Syverson, P.: Locating hidden servers. In: IEEE Symposium on Security and Privacy. pp. 100–114 (2006)
[24] Reed, M., Syverson, P., Goldschlag, D.: Anonymous connections and onion routing. In: IEEE Journal on Selected Areas in Communications. vol. 16, pp. 482–494 (1998)
[25] Shmatikov, V., Wang, M.H.: Timing analysis in low-latency mix networks: Attacks and defenses. In: European Symposium on Research in Computer Security. pp. 18–33 (2006)
[26] Sreeram Ramachandran, G.: Web metrics: Size and number of resources. https://developers.google.com/speed/articles/web-metrics
[27] Syverson, P., Tsudik, G., Reed, M., Landwehr, C.: Towards an analysis of onion routing security. In: International Workshop on Design Issues in Anonymity and Unobservability. pp. 96–114 (2001)
[28] Wright, M., Adler, M., Levine, B.N., Shields, C.: Defending anonymous communications against passive logging attacks. In: IEEE Symposium on Security and Privacy. pp. 28–41 (2003)
[29] Wright, M.K., Adler, M., Levine, B.N., Shields, C.: An analysis of the degradation of anonymous protocols. In: Network and Distributed System Security Symposium (2002)
[30] Zhu, Y., Fu, X., Graham, B., Bettati, R., Zhao, W.: On flow correlation attacks and countermeasures in mix networks. In: Workshop on Privacy Enhancing Technologies. pp. 207–225 (2004)