Punching Down: Deflation Attacks on the Tor Network

Ran Tao
Georgetown University
Washington, DC, USA

Ryan Wails
Stealth Software Technologies, Inc.*
Los Angeles, CA, USA

Cameron Daly
Georgetown University
Washington, DC, USA

James Trombo
Georgetown University
Washington, DC, USA

Micah Sherr
Georgetown University
Washington, DC, USA

Benjamin E. Ujcich
Georgetown University
Washington, DC, USA

Abstract

Tor is an anonymity network with relays run by thousands of volunteers across the world. Load-balancing among these relays depends on bandwidth measurement carried out by directory authorities. Prior work has explored attacks in which the attacker prioritizes measurement traffic to cause an inflation of weights (i.e., the probability of being selected) assigned to malicious relays.

We introduce a related but novel *deflation attack* in which exit relays operated by an adversary cause a decrease of weights assigned to other benign relays by throttling bandwidth measurements. We implement our attack and evaluate it using the Shadow simulator. We show that deflation attacks can have devastating impacts on both Tor's performance (e.g., a 5.5× slowdown using two malicious exits) and anonymity (e.g., increasing susceptibility to traffic correlation attacks by 4.1×). We then propose and evaluate an easy-to-deploy solution that completely thwarts the deflation attack.

CCS Concepts

• **Networks → Network privacy and anonymity**.

Keywords

Tor, anonymous communication, load balancing

ACM Reference Format:
Ran Tao, Ryan Wails, Cameron Daly, James Trombo, Micah Sherr, and Benjamin E. Ujcich. 2026. Punching Down: Deflation Attacks on the Tor Network. In *25th Workshop on Privacy in the Electronic Society (WPES '26), November 15–19, 2026, The Hague, Netherlands.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3847192.3847374

1 Introduction

Tor is an anonymity overlay network consisting of thousands of volunteer-run *relays* (routing nodes) distributed globally. To access the Internet anonymously, a *client* (the user's software) constructs a *circuit*, a multi-hop path through the overlay network, by selecting three random relays through which to route their layered-encrypted traffic [6, 21]. Given that volunteer-operated relays exhibit a wide variance in available bandwidth [3], this selection process is weighted by each relay's measured capacity to optimize network load balancing [25].

Accurate relay bandwidth measurement is critical to Tor's performance and anonymity properties. Measuring this capacity falls to *directory authorities* (or *dirauths*), a small set of trusted servers that maintain the definitive state of the network and publish hourly network consensus files. Several dirauths run the Simple Bandwidth Scanner (sbws) [15, 27] to actively probe and report relay capacities.

Because relays can easily distinguish active measurement probes from standard client traffic, malicious relay operators can exploit this visibility. Specifically, attackers can mount a selective denial-of-service (DoS) attack in which they prioritize measurement traffic while throttling normal client traffic to artificially inflate their reported bandwidth [13]. This *weight inflation* degrades the Tor network's load balancing and overall performance.

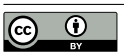
Attackers can achieve this inflation at no additional bandwidth cost by reallocating their existing capacity. Inflated bandwidth measurements increase the probability that malicious relays are selected by clients, thereby facilitating traffic analysis and deanonymization attacks that compromise user privacy [1, 14, 16].

Deflation attacks. In this paper, we introduce a novel class of adversary behavior: the *deflation attack*. In such an attack, a malicious exit relay (the final node in a circuit) detects when it has been selected as the second hop in an active measurement circuit designed to evaluate a preceding relay. Upon detection, the malicious relay throttles the measurement traffic, causing the target relay's measured throughput to appear artificially low (i.e., deflated).

Similar to inflating one's own bandwidth, deflating the measured bandwidth of honest competitors increases the relative selection probability of the malicious relays. As a result, such an attack degrades load balancing, reduces network performance, and increases deanonymization risks. We argue that deflation attacks are harder to detect than inflation attacks: inflation requires throttling active clients, who would notice the performance drop, whereas deflation targets only measurement traffic and causes no performance drop visible to clients. As such, measurements misattribute the low throughput to a bottleneck at the victim relay.

Contributions. We evaluate deflation attacks in Shadow [9, 10] simulations on generated networks proportionally scaled down to 1% of the real Tor network. The simulations show that deflation attacks decrease overall network performance (e.g., by 5.5× with two malicious exits) and cause the malicious relays to be selected more often (e.g., increasing the adversary's ability to perform traffic correlation attacks by 4.1×). We also design and implement a

*Wails was affiliated with Georgetown University when this research was conducted.



This work is licensed under a Creative Commons Attribution 4.0 International License.
WPES '26, The Hague, Netherlands
© 2026 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-3026-9/2026/11
https://doi.org/10.1145/3847192.3847374

defense against deflation attacks in which a trusted exit with sufficient bandwidth is dedicated as the permanent second hop in all measurement circuits. The simulations show that this defense completely defeats deflation attacks.

2 Background

Although Tor is mostly decentralized, it relies on nine directory authorities (dirauths) to publish consensus files that list information about each Tor relay (such as its IP address, bandwidth weight, relay flags, *etc.*) and the status of the whole network [7]. Five of the dirauths currently run sbws to perform relay bandwidth measurement. sbws exchanges traffic with websites dedicated to the measurement through a circuit that includes the measured relay. It then computes the throughput as the number of bytes transferred divided by the elapsed time. Because only exit relays (*i.e.*, relays with the exit flag) can route traffic outside of the Tor network, the measurement circuit is a two-hop circuit where the first hop is the measured relay and the second *helper* hop is selected among the exits.[1] To prevent the measurement from being bottlenecked, the second hop ideally needs higher bandwidth than the first [15, 27]. We expand on the measurement circuit in §3.

The IP addresses of the dirauths are public information. The destination web servers (with which sbws exchanges arbitrary data) are not intentionally hidden. They are easily fingerprintable [13, 26, 27], and their IP addresses are assumed to be public information. Therefore, the two-hop circuit means that the measured relay (*i.e.*, the one at the first hop) can always recognize that the traffic is exchanged with one of the dirauths, and the helper relay (the second hop) can always recognize that the traffic is exchanged with one of the destination web servers. Both relays, therefore, can recognize when they are being measured.

This is what makes inflation attacks possible: when a malicious relay recognizes that it is being measured, it executes a "selective DoS" attack [13] in which it throttles or drops normal client traffic to prioritize measurement traffic, making itself appear faster during measurement. Since clients' relay selection is weighted by relays' measured bandwidth, such inflation increases the probability of malicious relays being used by clients. This can degrade load-balancing and the overall performance of the Tor network. It can also lead to the deanonymization of more Tor clients, since the adversary can trivially identify a client via traffic analysis if both the guard and exit relay of the client's circuit are malicious [14, 16].

3 The Deflation Attack

In this section, we present the complement to the inflation attack—the *deflation* attack. In an inflation attack, the measured relay prioritizes sbws traffic to increase its bandwidth weight; in the deflation attack, the *helper* relay *throttles* measurement traffic to *decrease* the measured relay's bandwidth weight.

Threat Model. To effectively run the attack, the adversary is assumed to control the fastest exit in the Tor network. This can be achieved by running two[2] new exit relays on machines with 10 Gbps links, as the current fastest exit relays advertise about 1 Gbps bandwidth capacities. On a popular cloud hosting provider, Hetzner, it would cost about €800 per month to operate these relays [19]. Note that, by definition, there is always a fastest exit relay and hence someone is always in the privileged position to run this attack.

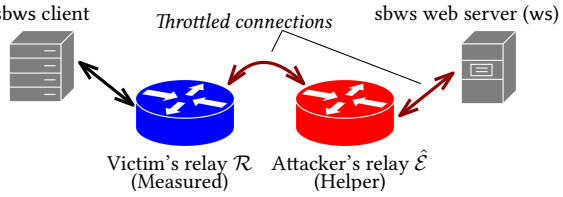


Figure 1: Schematic diagram of the deflation attack.

Deflation. A sbws two-hop circuit can be described as follows: sbws $\rightleftarrows$ $\mathcal{R}$ $\Longleftrightarrow$ $\mathcal{E}$ $\rightleftarrows$ ws, where sbws is the sbws client, $\mathcal{R}$ is the relay being measured, $\mathcal{E}$ is a randomly-chosen helper exit relay, and ws is the measurement web server. When a malicious exit is chosen by sbws, then, through its vantage point on the path sbws $\rightleftarrows$ $\mathcal{R}$ $\Longleftrightarrow$ $\hat{\mathcal{E}}$ $\rightleftarrows$ ws, it can throttle the measurement by introducing delays (in our case, via a token bucket). This setup is shown in Figure 1. Crucially, sbws cannot distinguish between a slow relay $\mathcal{R}$ and a malicious exit relay $\hat{\mathcal{E}}$ because it can only observe end-to-end performance; however, since sbws selects the exit relay $\hat{\mathcal{E}}$ because it has ample bandwidth, sbws always attributes the bottleneck to the measured relay. As a result, $\mathcal{R}'s$ bandwidth will be significantly underestimated by sbws.

Helper selection. For two reasons, to perform the deflation attack, the malicious relays must be selected for the helper position with high probability. First, and perhaps obviously, the malicious relay can deflate a victim's measured bandwidth only when it is selected as the helper. Second, there are currently five authorities running sbws (see §2). To arrive at a single bandwidth estimate for each relay, the median measurement is used, so a malicious relay must be present on the majority of the authorities' measurement circuits.

As mentioned previously, helper relays are selected randomly by sbws, which is seemingly problematic for the attack—there are thousands of relays in the Tor network. However, to avoid bottlenecked measurements, sbws makes the reasonable choice to enforce that helper is roughly 2× faster than the target relay (according to the bandwidth weights at the time of measurement): to select a helper for a relay with bandwidth B, the logic sbws uses is shown in the pseudocode below (original source code is given in Appendix A).

```
for w in {2, 1.75, 1.5, 1.25, 1} do
    S ← {exit relays with consensus weight ≥ w · B}
    if S = ∅ continue.
    return a uniformly random relay from S
end for
return The fastest exit relay          ▷ (fall back if no faster relay exists)
```

Hence, the fastest exit relays are chosen at disproportionally high rates to help measure other fast relays in the network. To demonstrate this effect, for a Tor consensus from July 16, 2026, we simulated a scenario wherein an adversary inserts two malicious exit relays, both provisioned with 2× the bandwidth of the largest exit in the

[1] The sbws documentation states the measured relay can be in either position, but we find this to be incorrect, as it deviates from what we see in the sbws source code.

[2] We assume the adversary runs *two* fast relays so that, as they are eventually measured by sbws prior to the attack, they can be used to measure each other; a single fastest relay would be bottlenecked during measurement by the (benign) second fastest.

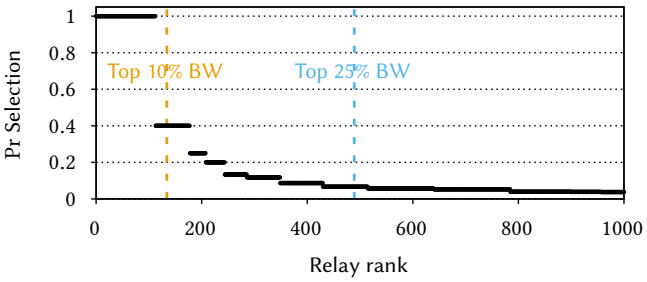


Figure 2: Probability of the fast malicious relays being selected as the helper node by sbws for the first 1000 fastest relays in the Tor network. 10% of Tor's bandwidth is provided by its fastest 134 relays; 25% by its fastest 489 relays.

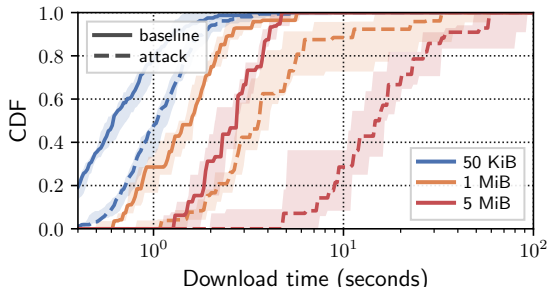


Figure 3: Distributions of download time (log scale) for 50 KiB, 1 MiB, and 5 MiB transfers through Tor.

network. Figure 2 shows the probability of either malicious exit being selected as the helper for the first 1000 fastest relays in the Tor network. *The fastest 112 relays, which represent nearly 10% of Tor's bandwidth, are measured using a malicious helper with 100% probability.* That is, a large fraction of Tor's fastest relays may be reliably deflated by the adversary.

Effect. The damage caused by the deflation attack is similar to the inflation attack. First, Tor's performance may be degraded: because victim relays have deflated consensus weights, they will be chosen less frequently and underutilized. If many relays are attacked, the overall load-balance and performance of the Tor network can be negatively impacted. Second, an adversary who also operates guard relays can target honest exit and guard relays with high consensus weights. By deflating honest relays, the adversary increases the probability that malicious guard and exit relays will be chosen. When positioned as the first and last hops in a circuit, the adversary can perform traffic correlation attacks [14, 16], breaking anonymity. We demonstrate both effects in the next section.

4 Implementation and Evaluation

We implement the deflation attack by modifying Tor 0.4.9.6 to apply its existing token-bucket rate-limiting algorithm to targeted (victim) relays. As our 274-line patch constitutes deployable attack code, we will restrict its distribution to research groups and the Tor Project (under a free license) rather than release it publicly.

Simulation setup. We use a 1%-scale Tor network (relative to the live network), as configured by Shadow's tornettools utility [11, 23]. This yields 79 relays, of which 46 are guards, 22 are exits, and 25 are middles; 14 have both the GUARD and EXIT flags. All simulations use Shadow 3.3.0 and are performed across a local cluster at our institution and Jetstream2 large-memory nodes [2, 8, 17]. As workloads, clients configured via tornettools perform 50 KiB, 1 MiB, and 5 MiB downloads, with load calibrated to approximate that of the live Tor network (i.e., `load_scale=2`). Additionally, simulations use three directory authorities, each of which runs its own sbws instance and uses a dedicated webserver.

We repeat each experiment five times. Error bars and shading in the figures depict the range of measurements across these runs.

Adversary configuration. Unless otherwise specified, we target our attacks at the top n non-malicious relays by consensus weight that have either the GUARD or EXIT flags. The adversary runs a few fast exits (usually two) and four mid-tier (closest to the median weight) guards. The adversary's guards are effectively honest-but-curious and collude with the adversary's exits to deanonymize connections—they otherwise do not misbehave.

As metrics, we consider client download time and the end-to-end compromise rate, where the latter is the fraction of Tor circuits in which both the guard and exit are operated by the adversary. To measure the efficacy of the attack, we consider a "baseline" network that is topologically identical to our attack configuration, but in which all participants behave correctly (i.e., the attack is disabled). Because Tor computes a relay's consensus weight as the median of the three measured bandwidths reported by the sbws instances, a deflation attack is possible only when malicious exits are chosen by the sbws instance in a majority ($\geq$ 2 of 3) of cases.

4.1 Network Performance

Rather than directly attacking the network's actual capacity (e.g., as in bandwidth denial-of-service), the deflation attack targets the network's load balancing. In effect, deflation causes relays' bandwidths to be mislabeled by sbws, resulting in poor resource utilization as high-bandwidth relays are miscored and thus underutilized.

Figure 3 shows the effect of the deflation attack on Tor's overall network performance when the adversary controls the two highest-bandwidth exit relays. The figure plots the cumulative distribution of download times for clients' 50 KiB, 1 MiB, and 5 MiB transfers, with (dashed lines) and without (solid lines) the attack. With the attack enabled, median client download times increase 1.7× for small 50 KiB transfers and 5.5× for 5 MiB transfers. This corresponds to a median increase in 5 MiB download time from 2.7s to 15.1s. The tail also increases significantly—for example, the 99th percentile download time for 5 MiB files increases by 18.8×.

4.2 Deanonymization

Tor is vulnerable to traffic-correlation attacks: an adversary who observes both sides of a circuit can trivially discern the sender and the receiver; even without controlling the middle relay, it can perform traffic analysis to confirm that both observations belong to the same circuit [14, 16].

Deflation allows an adversary to improve its observability of Tor traffic by degrading relays it does not control, increasing the probability that relays it *does* control are chosen in the guard and exit positions. We evaluate the effect of deflation on anonymity

	Compromised circuits	Amplification
Baseline	1.6% [1.1%–1.8%]	–
Attack	6.6% [6.1%–9.9%]	4.1×

Table 1: Percentage of Tor circuits an adversary controlling the top two exits and four mid-tier guards can deanonymize (median over five runs; min/max in brackets).

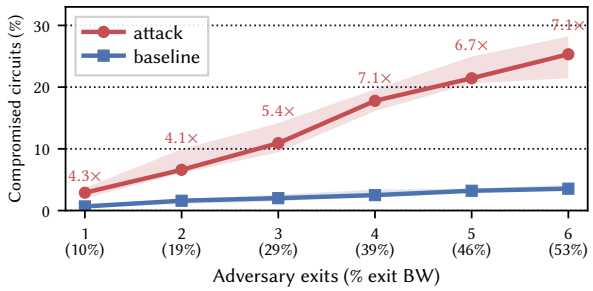


Figure 4: Percentage of compromised circuits (y-axis) as a function of the number of top exit relays (by bandwidth) controlled by an adversary.

using two malicious exits (as before) and four adversary-controlled guards. The guards are mid-tier and chosen to be those closest to the median, as measured by guard weight.

Table 1 shows the percentage of compromised circuits without and with the attack. Without the attack, the adversary deanonymizes 1.6% of circuits. By deflating other exits and guards, it increases its observability by 4.1× and deanonymizes 6.6% of circuits.

We also consider how the attack scales with the adversary's control over exit bandwidth. Figure 4 shows the percentage of compromised circuits as a function of the number of exit relays operated by the adversary; the percentage of adversary-controlled exit bandwidth is shown in parentheses. The number of malicious guards is held at four. With larger adversary coalitions, both the compromise rate and the "amplification factor" (the compromise rate relative to that achieved without the attack) increase. For example, if the adversary controls the top three exit relays, its ability to compromise Tor circuits increases by 5.4× (from 2.0% to 10.9%, at the median).

5 Defense

We defend against the deflation attack by assigning each sbws instance a dedicated, unlisted exit relay. The exit relay enforces a restrictive exit policy that permits connections only to the sbws measurement server. sbws instances construct measurement circuits using their dedicated relay as the second-hop. Consequently, a malicious public exit cannot occupy the second hop, eliminating the attack opportunity. There is no anonymity-set or load-balancing distortion because the dedicated relays carry no user traffic.

We evaluate the defense using the same Shadow configuration as in §4, with four malicious exits. Figure 5 shows the defense's efficacy. In an undefended network, the adversary supplies the second-hop for 89.6% of victim measurements (range 87.5%–91.5%), causing victim bandwidth to be underestimated by 27× (range 25×–35×)

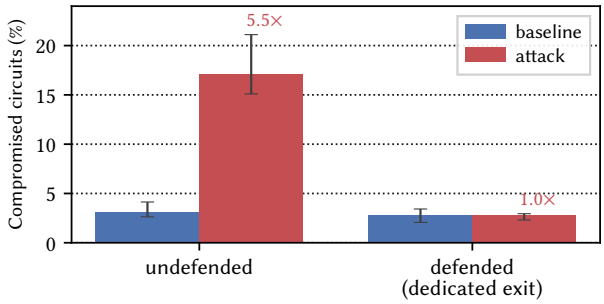


Figure 5: Percentage of end-to-end compromised circuits (y-axis) with and without the dedicated-exit defense.

and amplifying end-to-end compromise 5.5× (3.1% → 17.1%). With the defense, *no* measurement uses a malicious second-hop (0% in all runs), victim measured bandwidth is essentially unchanged (1.1×), and the compromised-circuit rate is similar to the baseline (1.0×).

The defense requires each sbws instance to host one exit and a web server. This should incur no additional cost if the instance has symmetric bandwidth. Crucially, the defense does not require changes to Tor—only a minimal sbws patch that pins measurements to the dedicated relay; hence, directory authorities can deploy it without a Tor release or relay upgrades.

6 Related Work

Johnson et al. were among the first to demonstrate attacks against TorFlow [18], the functionally similar predecessor to sbws [13]. They simulated a malicious exit that not only prioritizes measurement traffic (the selective DoS attack), but also exploits TorFlow's trust in self-reported bandwidth, falsely reporting arbitrarily high bandwidth to dirauths. The combined attacks achieved a weight inflation of 177× for their malicious relay. Sendner et al. enhanced the selective DoS attack by placing their simulated malicious relays on the same physical host or behind the same network gateway, provided that the host or gateway owns multiple public IP addresses [20]. Once the host or gateway recognizes measurement traffic, it dynamically reallocates all bandwidth to the measured relay, so that each relay appears to have the total available bandwidth rather than the fraction it actually has. Jansen et al. also explored DoS attacks in Tor, with TorFlow instances among the targets [12]. Their attacks use the more traditional strategy of flooding. Others have proposed improvements to TorFlow and sbws, as well as alternative systems [4, 5, 13, 22, 24]. To the best of our knowledge, we are the first to propose the deflation attack.

7 Conclusion

This paper introduces a novel deflation attack in Tor that allows malicious exits to decrease other benign relays' weights in the consensus by throttling sbws measurement traffic. We evaluate the deflation attack in Shadow and demonstrate that it degrades both performance and anonymity. We conclude by proposing a simple and lightweight solution that dedicates trusted exits as permanent second hops for all sbws measurement circuits, eliminating the means for the deflation attack.

Acknowledgments

This work was supported by the Georgetown University Callahan Family Chair Fund, Farr Faculty Award, and Fritz Family Fellowship. The findings and opinions expressed in this paper are those of the authors and do not necessarily reflect those of the funding organizations.

References

[1] K. Bauer, D. McCoy, D. Grunwald, T. Kohno, and D. Sicker. 2007. Low-resource Routing Attacks Against Tor. In *ACM Workshop on Privacy in the Electronic Society (WPES)*.

[2] Timothy J. Boerner, Stephen Deems, Thomas R. Furlani, Shelley L. Knuth, and John Towns. 2023. Access: advancing innovation: NSF's advanced cyberinfrastructure coordination ecosystem: services & support. In *Practice and Experience in Advanced Research Computing 2023: Computing for the Common Good* (PEARC '23). Association for Computing Machinery, New York, NY, USA, 173–176. ISBN: 9781450399852. doi:10.1145/3569951.3597559.

[3] [n. d.] Consensus health. Accessed: 2026-07-13. The Tor Project. https://consensus-health.torproject.org/.

[4] Hussein Darir, Nikita Borisov, and Geir E. Dullerud. 2024. DiProber: estimating relays capacities in underloaded anonymous communication networks. In *American Control Conference (ACC 2024)*. IEEE, 462–467. doi:10.23919/ACC60939.2024.10644851.

[5] Hussein Darir, Hussein Sibai, Chin-Yu Cheng, Nikita Borisov, Geir E. Dullerud, and Sayan Mitra. 2022. MLEFlow: learning from history to improve load balancing in Tor. *Proceedings on Privacy Enhancing Technologies (PoPETs)*, 2022, 1, 75–104. doi:10.2478/POPETS-2022-0005.

[6] Roger Dingledine, Nick Mathewson, and Paul F. Syverson. 2004. Tor: the second-generation onion router. In *Proceedings of the 13th USENIX Security Symposium*. USENIX, 303–320. http://www.usenix.org/publications/library/proceedings/sec04/tech/dingledine.html.

[7] [n. d.] Directory authorities. Accessed: 2026-07-13. The Tor Project. https://community.torproject.org/relay/governance/policies-and-proposals/directory-authority/.

[8] David Y. Hancock et al. 2021. Jetstream2: accelerating cloud computing via jetstream. In *Practice and Experience in Advanced Research Computing 2021: Evolution Across All Dimensions* (PEARC '21). Association for Computing Machinery, Boston, MA, USA. ISBN: 9781450382922. doi:10.1145/3437359.3465565.

[9] Rob Jansen and Nicholas Hopper. 2012. Shadow: running Tor in a box for accurate and efficient experimentation. In *19th Annual Network and Distributed System Security Symposium NDSS*. The Internet Society. https://www.ndss-symposium.org/ndss2012/shadow-running-tor-box-accurate-and-efficient-experimentation.

[10] Rob Jansen, Jim Newsome, and Ryan Wails. 2022. Co-opting Linux processes for high-performance network simulation. In *2022 USENIX Annual Technical Conference (USENIX ATC 22)*, 327–350.

[11] Rob Jansen, Justin Tracey, and Ian Goldberg. 2021. Once is never enough: foundations for sound statistical inference in Tor network experimentation. In *30th USENIX Security Symposium, USENIX Security 2021*. USENIX Association, 3415–3432. https://www.usenix.org/conference/usenixsecurity21/presentation/jansen.

[12] Rob Jansen, Tavish Vaidya, and Micah Sherr. 2019. Point break: A study of bandwidth denial-of-service attacks against Tor. In *28th USENIX Security Symposium (USENIX Security 2019)*. USENIX Association, 1823–1840. https://www.usenix.org/conference/usenixsecurity19/presentation/jansen.

[13] Aaron Johnson, Rob Jansen, Nicholas Hopper, Aaron Segal, and Paul Syverson. 2017. PeerFlow: secure load balancing in Tor. *Proceedings on Privacy Enhancing Technologies (PoPETs)*, 2017, 2, 74–94. doi:10.1515/POPETS-2017-0017.

[14] Aaron Johnson, Chris Wacek, Rob Jansen, Micah Sherr, and Paul F. Syverson. 2013. Users get routed: traffic correlation on Tor by realistic adversaries. In *2013 ACM SIGSAC Conference on Computer and Communications Security CCS'13*. ACM, 337–348. doi:10.1145/2508859.2516651.

[15] juga. [n. d.] How bandwidth scanners monitor the Tor network. Accessed: 2026-07-13. The Tor Project. https://blog.torproject.org/how-bandwidth-scanners-monitor-tor-network/.

[16] Steven J Murdoch and George Danezis. 2005. Low-cost traffic analysis of Tor. In *2005 IEEE Symposium on Security and Privacy (S&P'05)*. IEEE, 183–195.

[17] Michael Norman et al. 2021. Cloudbank: managed services to simplify cloud access for computer science research and education. In *Practice and Experience in Advanced Research Computing 2021: Evolution Across All Dimensions* (PEARC '21). Association for Computing Machinery, Boston, MA, USA. ISBN: 9781450382922. doi:10.1145/3437359.3465586.

[18] Mike Perry. 2009. Torflow: Tor Network Analysis. The Tor Project, 1–14. https://research.torproject.org/techreports/torflow-2009-08-07.pdf.

[19] 2026. Price dedicated server addons. Hetzner. https://docs.hetzner.com/robot/dedicated-server/dedicated-server-hardware/price-server-addons/.

[20] Christoph Sendner, Jasper Stang, Alexandra Dmitrienko, Raveen Wijewickrama, and Murtuza Jadliwala. TorMult: introducing a novel Tor bandwidth inflation attack. (2023). arXiv: 2307.08550. doi:10.48550/ARXIV.2307.08550.

[21] Robin Snader and Nikita Borisov. 2008. A tune-up for tor: improving security and performance in the tor network. In *Proceedings of the Network and Distributed System Security Symposium, NDSS*. The Internet Society. https://www.ndss-symposium.org/ndss2008/a-tune-up-for-tor-improving-security-and-performance-in-the-tor-network/.

[22] Robin Snader and Nikita Borisov. 2009. EigenSpeed: secure peer-to-peer bandwidth evaluation. In *Proceedings of the 8th International Conference on Peer-to-peer Systems (IPTPS'09)*. USENIX, 9. eprint: http://www.iptps.org/papers-2009/snader.pdf.

[23] [n. d.] Tornettools. Accessed: 2026-07-16. The Tor Project. https://github.com/shadow/tornettools.

[24] Matthew Traudt, Rob Jansen, and Aaron Johnson. 2021. FlashFlow: a secure speed test for Tor. In *41st IEEE International Conference on Distributed Computing Systems (ICDCS 2021)*. IEEE, 381–391. doi:10.1109/ICDCS51616.2021.00044.

[25] Chris Wacek, Henry Tan, Kevin S. Bauer, and Micah Sherr. 2013. An empirical evaluation of relay selection in Tor. In *20th Annual Network and Distributed System Security Symposium (NDSS)*. The Internet Society. eprint: https://www.ndss-symposium.org/ndss2013/empirical-evaluation-relay-selection-tor.

[26] [n. d.] Web server for bandwidth scanners. Accessed: 2026-07-13. The Tor Project. https://gitlab.torproject.org/tpo/network-health/bwscanner_webserver.

[27] [n. d.] Welcome to simple bandwidth scanner's documentation! Accessed: 2026-07-13. The Tor Project. https://tpo.pages.torproject.net/network-health/sbws/.

Appendices

A sbws Source Code

Listing 1 shows the helper relay selection routine from sbws v2.2.0 (`sbws/core/scanner.py:294`). Comments and unreachable code were removed, and a few explanatory comments have been added.

```
def _pick_ideal_second_hop(relay, rl, relay_as_entry, candidates):
    min_relay_bw = rl.exit_min_bw() # <- Set elsewhere to be roughly the 25th
                                    #    percentile of Tor bandwidth values

    for min_bw_factor in [2, 1.75, 1.5, 1.25, 1]:
        min_bw = relay.consensus_bandwidth * min_bw_factor

        if min_bw < min_relay_bw:  # <- These lines just ensure the selected
            min_bw = min_relay_bw  #    relay isn't too small

        new_candidates = stem_utils.only_relays_with_bandwidth(
            candidates, min_bw=min_bw
        )

        if len(new_candidates) > 0:
            chosen = rng.choice(new_candidates)
            return chosen

    candidates = sorted(
        candidates, key=lambda r: r.consensus_bandwidth, reverse=True
    )
    chosen = candidates[0]
    return chosen
```

Listing 1: sbws's helper selection algorithm.