D. Reinhardt, H. Langweg, B.C. Witt, M. Fischer et al. (Hrsg.): Sicherheit 2020,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Göttingen 2020 29

Analyzing PeerFlow – A Bandwidth Estimation System for Untrustworthy Environments

Asya Mitseva[1], Thomas Engel[2], Andriy Panchenko[3]

Abstract: Tor is the most popular low-latency anonymization network comprising over 7,000 nodes, mostly run by volunteers. To balance user traffic over the diverse resource capabilities of those nodes, Tor users choose nodes in proportion to their available bandwidth. However, since self-reported bandwidth values are not trustworthy and active measurements put undesired load on the network, an amount of research looks for alternatives. Recently, a new bandwidth measurement system, PeerFlow, has been inroduced aiming to solve the Tor bandwidth estimation problem. In this work, we perform a practical analysis of PeerFlow. We propose a set of strategies for the practical realization of probation periods in PeerFlow and show that many Tor nodes fail to recover to their normal state after one measuring period. We also demonstrate that low-bandwidth adversaries gain significantly higher bandwidth estimates, exceeding the theoretically defined security boundaries of PeerFlow.

Keywords: Privacy; Anonymous Communication; Onion Routing; Tor; Security; Performance

1 Introduction

In the age of mass surveillance and censorship, users rely on anonymization techniques to exercise their right to freedom of expression and to freely access information. Currently, Tor [DMS04, DM18b] is the most popular anonymization network designed for low-latency applications, e.g., web browsing. To hide the identity (i.e., IP address) of a user, the user traffic in Tor is sent through a virtual tunnel, i.e., *circuit*, over three nodes, called *onion relays* (ORs). Depending on their position in the circuit, the ORs are known as *entry*, *middle*, and *exit*. A set of trusted ORs, called *directory authorities* (DAs), periodically distribute a list of available ORs together with their identities and a status description in the form of a *consensus* document [bib18]. The Tor users select ORs for circuits probabilistically according to the ORs' consensus bandwidth, availability, and exit policy to a given target [DM18a]. The users negotiate a separate symmetric key with each OR in a circuit and utilize these keys to encrypt user data in multiple layers [DM18b]. While forwarding the user data, each OR removes (or adds, depending on the direction) a layer of encryption. Thus, no OR in the circuit knows both the user and its destination at the same time.

[1] University of Luxembourg, Computer Science and Communications Research Unit, 6 Avenue de la Fonte, L-4364 Esch-sur-Alzette, Luxembourg asya.mitseva@uni.lu
[2] University of Luxembourg, Computer Science and Communications Research Unit, 6 Avenue de la Fonte, L-4364 Esch-sur-Alzette, Luxembourg thomas.engel@uni.lu
[3] Brandenburg University of Technology, Chair of IT Security, Konrad-Wachsmann-Allee 5, 03046 Cottbus, Germany andriy.panchenko@b-tu.de

doi:10.18420/sicherheit2020_02

Nowadays, Tor comprises over 7,000 ORs run by volunteers and carries terabytes of traffic for more than two million daily users [Met19]. To balance the large user traffic load over the diverse resource capabilities of the ORs, Tor entirely relied on self-reported bandwidth values in its original specification. However, malicious ORs could easily lie about their resources by advertising significantly higher bandwidth than they actually have and, thus, attract a larger fraction of traffic to be routed through them [BMG 07]. In response, Tor hardened its bandwidth estimation method by combining both self-reported values and external measurements. Nevertheless, this method remains an attractive target for adversaries aiming to compromise user connections. If an attacker can strategically run ORs and give the impression that these ORs have a large amount of available bandwidth, he can dramatically increase the likelihood of these ORs to be selected for Tor connections more often than they would have been chosen when advertising their actual capacity. As a result, these malicious ORs may either frequently appear as entry ORs in user connections, which allows them to conduct website fingerprinting [PLZ 16, SIJW18], or be chosen for both entry and exit ORs by users and, thus, execute traffic correlation [BMG 07, JWJ 13]. Some Tor users may even create Tor connections entirely comprising malicious ORs, which leads to a trivial deanonymization of these communications.

To limit the threat of attackers trying to cheat the bandwidth estimation method, Tor currently conducts periodic bandwidth measurements of all ORs executed by the DAs [LPG11, Per09]. However, previous studies [BPW13, JJH 17] showed that these measurements are easily identifiable and prone to attacks (i.e., detected measurement probes can be prioritized). The authors demonstrated that malicious ORs can make the bandwidth estimation method to believe that they have up to 177 times higher bandwidth than their actual one. Until recently, EigenSpeed, an opportunistic bandwidth estimation system proposed by Snader and Borisov [SB09], was considered as the main alternative of the current Tor bandwidth measurement system. EigenSpeed relies on passive measurements between ORs collected while communicating with each other that are reported to the DAs. The DAs employ Principal Component Analysis to compute bandwidth estimates for each OR. However, Johnson et al. [JJH 17] have recently shown fundamental flaws in this approach, which allow malicious ORs to either get a large number of honest ORs kicked out of Tor or obtain up to 28 times higher bandwidth than they actually have. In response, the authors presented the most recent proposal, PeerFlow, aiming to solve the Tor bandwidth estimation problem.

In this paper, we introduce the analysis of PeerFlow. In particular, *our contributions* are as follows: (*i*) We implement and practically evaluate the impact of probation phases (methods applied when the observed bandwidth is lower than the expected/believed one to give ORs a chance to prove that they are able to transmit traffic at a higher rate) on PeerFlow's performance. For this, we consider several strategies for the practical realization of probation periods. (*ii*) We propose a set of novel attacks against PeerFlow, which allow a low-bandwidth attacker to gain significantly higher bandwidth estimates than the theoretical security boundaries defined by the authors of PeerFlow. We show that the variable duration of measurement periods in PeerFlow introduces fundamental flaws in the approach.

2 State-of-the-art Bandwidth Estimation Systems for Tor

To limit the use of self-reported bandwidth values, Tor currently applies an active bandwidth-scanning system, TorFlow [LPG11, Per09], managed by the DAs. Each DA willing to measure ORs' bandwidth operates four bandwidth scanners. Each of these scanners is responsible for a quarter of all currently available ORs ordered by their capacity from the most recent consensus. Each bandwidth scanner further divides its list of ORs into groups, called *slices*, of ORs of similar bandwidth and uses a Tor client to create two-hop circuits via different ORs in a slice. Based on the total bandwidth of a slice, the Tor client downloads a file of a predefined size from the same, publicly-known server. This process is repeated several times as long as a sufficient quantity of measurements is collected. Each bandwidth scanner computes the average bandwidth of an OR, measured on circuits in a slice containing this OR. Once per hour, each measuring DA aggregates these values and estimates the amount of available bandwidth of each OR. This is achieved by multiplying the self-reported bandwidth of the given OR by the ratio of the measured bandwidth of that OR to the average network bandwidth. This data is then used to produce the consensus bandwidth values of ORs.

Despite its main goal to make Tor resistant to bandwidth manipulation, previous research [BPW13, JJH 17] showed that TorFlow remains vulnerable to common attacks. As the self-reported bandwidth is a multiplier in the TorFlow's bandwidth estimation formula, malicious ORs can still announce significantly higher bogus bandwidth than their actual one and influence on their consensus bandwidth values (as shown in [BMG 07]). Even worse, an adversary's OR can easily identify measurement circuits due to their specific length and the publicly-known IP addresses of the measuring nodes. Consequently, it can provide more bandwidth to those circuits while suppressing all others. This, in turn, gives the impression that this OR has high capacity by using a meager amount of resources.

Bauer et al. [BMG 07] proposed the use of a proactive distributed probing of nodes, e.g., by applying anonymous auditing [SNDW06]. Snader et al. [SB08] suggested that ORs collect bandwidth statistics for other ORs during an interaction with them and announce these statistics to the DAs. Instead of measuring node's bandwidth capacity, several authors [SBL09, PR09] proposed that Tor users create paths based on link characteristics, e.g., latency, jitter, loss, to reduce the likelihood of manipulation.

To address the limitations of TorFlow, Snader et al. [SB09] suggested a bandwidth estimation system, EigenSpeed, based on passive measurements. In EigenSpeed, each OR records the speed of each connection with another OR. The collected data is weighted with the current exponentially-weighted moving average of the connections and reported to the DAs. EigenSpeed was considered to provide secure bandwidth estimation [SB11] and was suggested for deployment in Tor before Johnson et al. [JJH 17] revealed major flaws of the system. To overcome the discovered drawbacks, the authors proposed another bandwidth

Bandwidth scanners belonging to a single DA use the same Tor client running on the DA itself [LPG11].
https://trac.torproject.org/projects/tor/ticket/5464

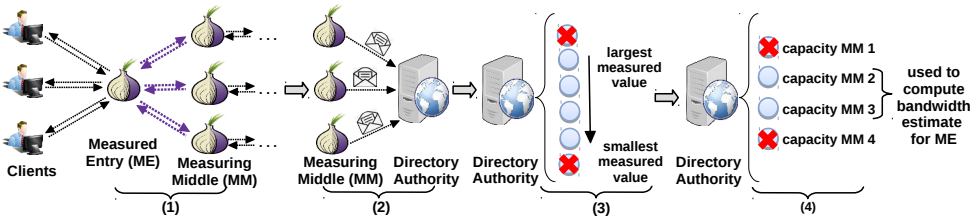


Fig. 1: A brief overview of the PeerFlow operation.

measurement system, PeerFlow, relying on a bandwidth-weight voting mechanism to compute consensus bandwidth of ORs. Contrary to previous work that mainly focus on developing alternative bandwidth estimation systems for Tor, in our work we introduce the first analysis of PeerFlow.

3 Operation of PeerFlow

PeerFlow [JJH 17] relies on a bandwidth-weight voting mechanism to calculate the consensus bandwidth of ORs. Figure 1 illustrates the main steps from the operation of PeerFlow. Based on the idea of EigenSpeed, PeerFlow utilizes passive measurements collected by the ORs while communicating between each other. To this end, each OR counts the number of bytes sent to or received from another OR and adds a random noise to each measurement to prevent information leakage (step 1). This data is periodically collected by each OR and reported to the DAs (step 2). The DAs, in turn, weigh each measurement with the probability of choosing the corresponding measuring OR in a given circuit position (i.e., whether the measuring OR can be selected for an entry, middle or exit node in that circuit). Then, the DAs sort the data gathered for each measured OR according to its possible circuit position (step 3). The largest and smallest values are discarded. Next, the DAs assign to each measurement a voting weight proportional to the *current* capacity of the corresponding measuring OR (i.e., the OR that reported this measurement) and take into account only those statistics whose voting weights are positive and fall into a predefined interval (step 4). The current OR's capacity is estimated based on the traffic that the OR was able to transfer in its last measurement period. Finally, the DAs aggregate the selected statistics to compute the total traffic r_t relayed by an OR r. If r_t is significantly lower than the amount of traffic that the OR r is *expected* to relay (based on history) and its self-reported bandwidth is significantly larger than r_t, the relay enters a *probation* state whereas its consensus bandwidth is not decreased yet. The expected amount of traffic transmitted by an OR is calculated based on the median of inferred bytes transferred by the set of ORs that can be

According to the Tor specification, only ORs which are able to fulfill a set of predefined requirements (e.g., availability, exit policy, etc.) can be chosen for an entry or exit position in a circuit. Based on OR's available bandwidth, the DAs further calculate *consensus weights* used to balance traffic load across different circuit positions that a single OR can be selected for. We omit further details about the selection of an OR for a given circuit position due to space constraints and refer the reader to [DM18b] for more information.

selected for the same position in the last measurement period. If in the next measurement period the OR r does not manage to prove that it is able to transmit traffic at a given rate (by exchanging dummy traffic with a set of measuring ORs), the DAs reduce the consensus bandwidth of this OR. As the authors of [JJH 17] do not specify how ORs are informed when they enter a probation state, in Section 5 we discuss possible methods used by the ORs to detect their probation state. The duration of measurement periods is variable for each OR and depends on how quickly the OR can exchange an amount of user traffic with each measuring OR. The goal of this variability of the measurement periods is to guarantee that the overall random noise added to the reported measurements is relatively small and, thus, ensure accurate bandwidth estimates. Moreover, new ORs joining Tor are selected for middle position only and their measurements are not considered for a given period of time to prevent manipulation.

4 Experimental Setup

To allow for verifiable results of our analysis, we next present our experimental setup. For our evaluation, we deployed the same Tor network configuration used in [JJH 17] consisting of four DAs, 498 ORs, 7,500 Tor users, and 1,000 web servers as traffic destinations. We simulated the operation of the private Tor network by utilizing Shadow [JH12] – a parallel discrete-event network simulator particularly designed for Tor network experimentation. Like [JJH 17], each experiment was repeated three times to measure variability across runs.

Simulation of dummy traffic: For the evaluation of each strategy used to distribute dummy traffic by ORs in probation, we count the median and maximum number of ORs in probation state in a given second. We also compute *inaccuracy* and *imprecision* to take into account errors in bandwidth weight calculations. The inaccuracy represents the ratio of the true relative bandwidth of an OR to the relative consensus bandwidth of the same OR. The true relative capacity of an OR, in turn, is the ratio of the true bandwidth of that OR to the true total network bandwidth. The relative consensus bandwidth of an OR is the ratio of the consensus bandwidth of that OR to the total network consensus bandwidth. We also calculate the imprecision of the estimated consensus bandwidth of each OR over different consensuses. In the rest of the paper, the inaccuracy is averaged across all ORs and consensuses whereas the imprecision is averaged across all ORs in an experiment. For both metrics, we show the average and standard deviation across the three experiments.

Adversarial model: To analyze different attacks against PeerFlow, we assume an active adversary aiming to manipulate the operation of PeerFlow by giving the impression that it has a high amount of bandwidth by using a meager amount of resources. Similar to the authors of PeerFlow, we focus on an adversary who controls several ORs representing 4% of the total network bandwidth.

5 Probation Periods in PeerFlow

In PeerFlow, an OR enters a probation state when the amount of traffic relayed by this OR in its last measurement period was lower than the amount of traffic that the OR was expected to transmit. During its probation period, the consensus bandwidth of that OR is not decreased yet and the OR has the possibility to prove that it is able to transmit traffic at a given rate by exchanging dummy traffic with a set of (measuring) ORs. The authors of [JJH 17] did not fully develop the probation in their proposal, but simply measured how often ORs that sent little traffic would be put into probation state. However, this is not sufficient to explore the performance of PeerFlow and its resistance to bandwidth manipulations in real settings. For instance, a fraction of the measuring ORs may be malicious and may not cooperate with honest ORs in probation. Here, we analyze different strategies for the practical realization of probation periods in PeerFlow and reveal their influence on the Tor performance.

To implement the probation in PeerFlow, we assume that all ORs can estimate the set of currently measuring ORs in Tor (i.e., ORs with positive probability of choosing them in a given circuit position and positive voting weights). This is a realistic assumption as the voting weights of the ORs used to determine the set of measuring ORs are proportional to the consensus bandwidth of these ORs. We also assume that all ORs know the start- and end-time of their measurement periods and when they enter the probation state. To accomplish this in reality, the ORs can keep track of which measuring ORs they communicated with (and the bytes exchanged with them) and can roughly estimate when their measurement period finishes and the amount of traffic that would be observed by the DAs. Once an OR enters the probation state, it starts exchanging dummy traffic with measuring ORs to prove that it is capable to transmit traffic at a given rate [JJH 17]. Since all ORs know their maximum bandwidth advertised to the network and can keep information about the number of bytes exchanged with other ORs, the ORs in probation are also able to calculate the amount of free bandwidth that can be used for dummy traffic with measuring ORs. The measuring ORs, in turn, send the collected dummy reports to the DAs. Contrary to [JJH 17], in our work the DAs do not automatically (regardless of the measurement results) recover ORs in probation in the next measurement period. If an OR in probation does not manage to recover in the next measurement period, the DAs decrease its capacity to a fraction ϵ_{dec} over its last consensus bandwidth. As suggested in [JJH 17], we used $\epsilon_{dec} = 0.25$ to allow a natural variation in traffic amounts and at the same time to avoid too much underperformance of ORs. We also implemented the update of voting weights of the measuring ORs in the PeerFlow prototype based on the design presented in [JJH 17]. The practical realization of voting updates is important for our work as the number of inferred bytes computed from reports of a measuring OR relies on the voting weight of this OR. The larger the voting weight of a measuring OR is, the larger is the final inferred estimate of the measured OR.

Based on what amount of dummy traffic is distributed among different measuring ORs, we study several strategies for the realization of dummy traffic generated by ORs in probation. The first strategy, called *dummy for all measuring ORs*, assumes that an OR in probation knows the set of all measuring ORs for a given circuit position in the current measurement

Tab. 1: Weight errors, median, and maximum number of relays in probation.

PeerFlow	Inaccuracy		Imprecision		ORs in probation	
	Avg.	Std. Dev.	Avg.	Std. Dev.	Median	Max
Baseline	0.229	0.001	0.398	0.0052	0	13
Dummy for all measuring ORs	0.226	0.0048	0.369	0.0067	25	121
Dummy for measuring ORs without connections	0.228	0.0078	0.365	0.011	7	71
Dummy for measuring ORs with connections	0.263	0.0041	0.410	0.0038	7	78
Optimistic strategy	0.233	0.0069	0.369	0.0045	16	46

period. The OR exchanges equal amount of dummy traffic with all of these measuring ORs regardless whether it has some existing exchange with some of them or not. If the OR in probation can be simultaneously selected for two circuit positions, i.e., it acts as entry and middle or exit and middle, we first check if it has enough amount of unused bandwidth to exchange dummy traffic for both positions. If not, we select the set of measuring ORs that contains fewer measuring ORs. Thus, the OR in probation can exchange more dummy bytes with each of the measuring ORs. As shown in Table 1, out of the 498 ORs the median number of ORs in probation is at most 25 and the maximum is at most 121. A possible reason for the significantly high number of ORs in probation compared to the original approach may be that most of these ORs do not have enough capacity to communicate with all measuring ORs. In other words, they exchange a little amount of dummy traffic with each of the measuring ORs, which is not sufficient to influence the final estimation by the DAs. Moreover, as the DAs do not consider the reports from all measuring ORs (i.e., the collected statistics are trimmed before computing the total number of inferred bytes), a part of the real traffic will not be considered by the DAs at all. This, in turn, hinders ORs in probation to recover from the probation state as the remaining (dummy) traffic is not sufficient for that.

Instead of connecting to all measuring ORs for a given circuit position, in the second strategy, called *dummy for measuring ORs without connections*, the OR in probation prefers to contact only those measuring ORs, which it has not communicated with in the current measurement period yet and exchanges an equal amount of dummy traffic with each of them. Here, the median number of ORs in probation reduces to at most 7 and the maximum is at most 71. The obtained results confirm our hypothesis presented in the previous paragraph. Hence, this is a superior strategy for probation period. We also examine the strategy, called *dummy for measuring ORs with connections*, where an OR in probation exchanges an equal amount of additional, dummy traffic only with those measuring ORs it has already communicated with in the current measurement period. I.e., the measuring ORs add the dummy traffic to already existing reports for the ORs in probation and send them to the DAs. The median number of ORs in probation remains 7 and the maximum is at most 78.

We further try to optimize the latter scenario. We consider an *optimistic strategy* where

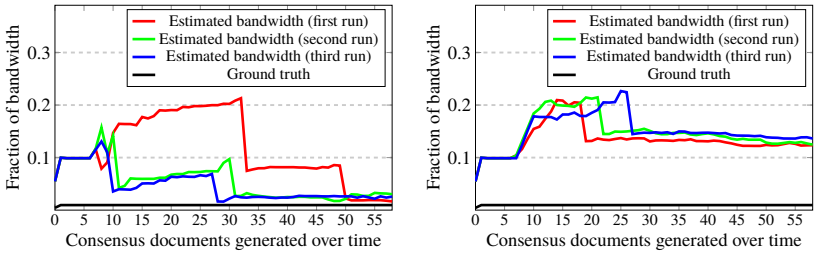
we assume that an OR in probation can correctly filter only those measuring ORs whose statistics will be considered by the DAs after trimming. Having this knowledge would allow ORs to optimally distribute their bandwidth to boost the ranking. The OR in probation sums the number of bytes already reported by those measuring ORs for it and its unused bandwidth in order to compute the average amount of traffic that can be exchanged with each of the non-trimmed ORs in total. The difference between the total average number of bytes and the real inferred bytes is transmitted in the form of dummy traffic and added by the measuring ORs to their statistics reported to the DAs. To the best of our knowledge, this is the best strategy that ORs in probation phase can apply to increase their ranking without tearing down user connections. While this strategy significantly reduces the maximum number of ORs in probation to 46, the median number of ORs in probation increases to 16, as shown in Table 1. Similar to the first strategy, a possible reason for this may also be the fact that a larger number of relays in probation do not manage to recover from the probation state in the next measurement period. Although each of the measuring ORs reports the maximum number of bytes that can be exchanged with the OR in probation on average, the DAs assign different weights to the distinct reports based the voting weight of the measuring OR and the time interval needed to transmit this data.

To sum up, both strategies *dummy for measuring ORs with connections* and *dummy for measuring ORs without connections* achieve the minimum median number of ORs in probation in a given second whereas the *dummy for measuring ORs without connections* reduces the maximum number of ORs in probation to 7 ORs more contrary to *dummy for measuring ORs with connections*. For the maximum number of ORs in probation, the best results are achieved by the *optimistic strategy*. Nevertheless, the set of ORs in probation remains significantly higher than expected by the authors [JJH 17]. In addition, we also observed that most of the ORs falling in probation were not able to recover within a single measurement period. In terms of inaccuracy and imprecision, all strategies behave consistent and similar. Therefore, to optimize median number of ORs in probation periods in PeerFlow, we apply the strategy *dummy for measuring ORs without connections* in the rest of our work.

6 Attacks on PeerFlow

In this section, we propose and analyze three new attacks against PeerFlow. In our evaluation, we assume that the attacker exploits the fact that PeerFlow uses TorFlow measurements to determine the bandwidth capacity of newly bootstrapping ORs. As shown in [JJH 17], a malicious OR can dramatically inflate its consensus bandwidth by providing more bandwidth to measuring TorFlow circuits and suppressing all others. We simulate this when our malicious ORs join in the network and inflate their consensus bandwidth by factor of 10.

Attack 1: We first analyze a simple attack, in which our malicious ORs communicate with other honest ORs as usual, but do not establish connections between each other. At the same time, they send dummy statistics for other malicious ORs with high bandwidth values and artificially decrease the bandwidth values reported for the honest ORs. Figure 2(a)



(a) Attack 1 (b) Attack 2

Fig. 2: Fraction of total consensus bandwidth obtained by PeerFlow for our malicious ORs.

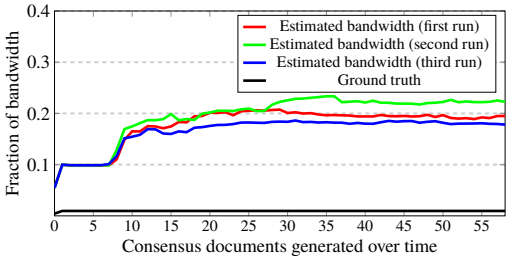


Fig. 3: Fraction of total consensus bandwidth obtained by PeerFlow for our malicious ORs in Attack 3.

shows the fraction of bandwidth estimated by PeerFlow for these ORs versus the fraction of their real bandwidth capacity. After completing the first measurement period and taking into account the first Peerflow measurements, we confirm that the bandwidth inflation of the malicious ORs is limited according to the security boundaries presented in [JJH 17]. Nevertheless, we see that at the beginning of the experiment for a certain period of time our malicious relays manage to gain bandwidth inflation factor of up to 9 (compared to the security boundary of 4.6 for small adversaries defined by the authors).

Attack 2: Although we were able to keep a high bandwidth inflation of the malicious ORs for a given time period, the success of the attack depends on the fact whether malicious ORs are selected as measuring ORs. According to the authors, the DAs will publish the last known bandwidth value of an OR in the consensus as long as the current PeerFlow measurement period does not finish. Based on this knowledge, we try to improve the latter attack by exploiting the fact that the length of the next measuring period of our malicious ORs rely on the number of measuring ORs they communicate with and the amount of traffic they transmitted to those measuring ORs. In other words, if our malicious ORs connect to a restricted number of measuring ORs and transmit very little amount of traffic or they do not connect to them at all, their measuring period will never end. This, in turn, will force the DAs to keep their initial bandwidth estimate obtained by the TorFlow measurements. As

shown in Figure 2(b), we see that in the first half of the experiment the fraction of consensus bandwidth gained by the malicious ORs is approx. 30 times higher than their actual capacity.

Attack 3: In spite of the large bandwidth inflation achieved above, we still observe that some of the malicious ORs are not able to keep their measuring period infinitive. A reason for this is the fact that our malicious ORs try to avoid connections to currently measuring ORs. However, the set of measuring ORs is changing permanently. This means that measurements done during previous connections to other ORs can be considered later by the DAs if these ORs become measuring relays. Therefore, we further improve the latter attack by avoiding malicious ORs to connect to other high-capacity ORs, i.e., whose consensus bandwidth is higher than the real capacity of our malicious ORs. Thus, we try to prevent communications with possible future measuring ORs. Figure 3 shows the obtained results. As it can be seen, we are able to keep the inflated consensus bandwidth of our malicious ORs constantly high.

To sum up, we showed that low-bandwidth adversaries are able to gain significantly higher bandwidth estimates exceeding the theoretically defined security boundaries of PeerFlow. The main reason for this is the fact that PeerFlow completely rely on the already vulnerable TorFlow measurements during the bootstrapping period of new ORs joining Tor. We further demonstrated that the variable duration of measurement periods in PeerFlow introduces fundamental flaws in the approach. In particular, malicious ORs can selectively refuse connections with measuring ORs and make their first PeerFlow measurement period never end. As a result, the inflated bandwidth of the malicious ORs obtained by TorFlow will be kept in the consensus, which, in turn, will attract more Tor users to use these ORs.

7 Conclusion

In this paper, we presented the first analysis of the most recent proposal, PeerFlow, aiming to solve the Tor bandwidth estimation problem. To this end, we implemented and evaluated different strategies for the practical realization of probation periods in PeerFlow and showed that many honest ORs cannot recover to their normal state after one measuring period. A possible solution would be that ORs in probation prioritize the exchange of dummy traffic to recover faster to their normal state. Nevertheless, we argue that the practical realization of probation periods needs further research before adopting PeerFlow into Tor. Thereafter, we showed that low-bandwidth adversaries can gain significantly higher bandwidth estimates exceeding the security boundaries of PeerFlow due to two undesirable issues. First, PeerFlow relies on initial bandwidth measurements prone to bootstrapping attacks. Second, as the measurement periods in PeerFlow have a variable length, malicious ORs can selectively refuse connections with measuring ORs and make their first measurement period never end. A possible solution would be to keep track of ORs with often unsuccessful connection establishments and report them to the DAs. Expect evaluating this, for future work we plan to extend our analysis on PeerFlow to identify whether low-bandwidth adversaries can gain a control over a set of measuring ORs and, thus, further manipulate PeerFlow estimations.

Bibliography

[bib18] Tor Directory Protocol, Version 3. `https://gitweb.torproject.org/torspec.git/tree/dir-spec.txt`, 2018.

[BMG 07] Kevin Bauer, Damon McCoy, Dirk Grunwald, Tadayoshi Kohno, and Douglas Sicker. Low-resource Routing Attacks Against Tor. In *Workshop on Privacy in Electronic Society*, pages 11–20, Alexandria, USA, October 2007. ACM.

[BPW13] Alex Biryukov, Ivan Pustogarov, and Ralf-Philipp Weinmann. Trawling for Tor Hidden Services: Detection, Measurement, Deanonymization. In *Symposium on Security and Privacy*, pages 80–94, Berkeley, USA, May 2013. IEEE.

[DM18a] Roger Dingledine and Nick Mathewson. Tor Path Specification. `https://gitweb.torproject.org/torspec.git/plain/path-spec.txt`, 2018.

[DM18b] Roger Dingledine and Nick Mathewson. Tor Protocol Specification. `https://gitweb.torproject.org/torspec.git/plain/tor-spec.txt`, 2018.

[DMS04] Roger Dingledine, Nick Mathewson, and Paul Syverson. Tor: The Second-generation Onion Router. In *13th conference on USENIX Security Symposium*, San Diego, USA, August 2004. USENIX Association.

[JH12] Rob Jansen and Nicholas Hopper. Shadow: Running Tor in a Box for Accurate and Efficient Experimentation. In *19rd Network and Distributed System Security Symposium*, San Diego, USA, February 2012. Internet Society.

[JJH 17] Aaron Johnson, Rob Jansen, Nicholas Hopper, Aaron Segal, and Paul Syverson. PeerFlow: Secure Load Balancing in Tor. *17th Symposium on Privacy Enhancing Technologies*, pages 74–94, July 2017.

[JWJ 13] Aaron Johnson, Chris Wacek, Rob Jansen, Micah Sherr, and Paul Syverson. Users Get Routed: Traffic Correlation on Tor by Realistic Adversaries. In *20th conference on Computer and Communications Security*, pages 337–348, Berlin, Germany, November 2013. ACM.

[LPG11] Karsten Loesing, Mike Perry, and Aaron Gibson. Bandwidth Scanner specification. `https://github.com/AdrienLE/torflow/blob/master/NetworkScanners/BwAuthority/README.spec.txt`, 2011.

[Met19] Tor Metrics. `https://metrics.torproject.org/`, 2019.

[Per09] Mike Perry. TorFlow: Tor Network Analysis. In *2nd Hot Topics in Privacy Enhancing Technologies*, Seattle, WA, USA, August 2009. Springer.

[PLZ 16] Andriy Panchenko, Fabian Lanze, Andreas Zinnen, Martin Henze, Jan Pennekamp, Klaus Wehrle, and Thomas Engel. Website Fingerprinting at Internet Scale. In *23rd Network and Distributed System Security Symposium*, San Diego, USA, February 2016. Internet Society.

[PR09] Andriy Panchenko and Johannes Renner. Path Selection Metrics for Performance-Improved Onion Routing. In *9th IEEE/IPSJ Symposium on Applications and the Internet*, Seattle, USA, July 2009. IEEE.

[SB08] Robin Snader and Nikita Borisov. A Tune-up for Tor: Improving Security and Performance in the Tor Network. In *16th Network and Distributed System Security Symposium*, San Diego, USA, February 2008. Internet Society.

[SB09] Robin Snader and Nikita Borisov. EigenSpeed: Secure Peer-to-peer Bandwidth Evaluation. In *8th International Conference on Peer-to-peer Systems*, Boston, USA, April 2009. USENIX Association.

[SB11] Robin Snader and Nikita Borisov. Improving Security and Performance in the Tor Network through Tunable Path Selection. *IEEE Transactions on Dependable and Secure Computing*, 8(5):728–741, September–October 2011.

[SBL09] Micah Sherr, Matt Blaze, and Boon Thau Loo. Scalable Link-based Relay Selection for Anonymous Routing. In *9th Symposium on Privacy Enhancing Technologies*, Seattle, USA, August 2009. Springer.

[SIJW18] Payap Sirinam, Mohsen Imani, Marc Juarez, and Matthew Wright. Deep Fingerprinting: Undermining Website Fingerprinting Defenses with Deep Learning. In *25th conference on Computer and Communications Security*, pages 1928–1943, Toronto, Canada, October 2018. ACM.

[SNDW06] Atul Singh, Tsuen-Wan "Johnny" Ngan, Peter Druschel, and Dan S. Wallach. Eclipse Attacks on Overlay Networks: Threats and Defenses. In *25th International Conference on Computer Communications*, Barcelona, Spain, April 2006. IEEE.