A case study on DDoS attacks against Tor relays

Tobias Höller
Johannes Kepler University
Linz, Austria
hoeller@ins.jku.at

René Mayrhofer
Johannes Kepler University
Linz, Austria
rm@ins.jku.at

ABSTRACT

Being the victim of DDoS attacks is an experience shared by many Tor relay operators. Despite the prevalence of this type of attack, the experiences and lessons learned after such attacks are rarely discussed publicly. This work provides a detailed description of a DDoS attack against two Tor relays operated by the authors. By sharing experiences on how an attack was analyzed after it happened and what mitigation mechanisms would have been capable of stopping it, this work tries to support a discussion on guidelines for relay operators on how to properly and securely run their relays. In addition to that, the included attack analysis investigates why the attack took place in the first time, what the attackers were trying to achieve, the amount of resources they had to expend and how the attack actually worked. Hopefully, this information will be useful in future discussions on how to make the Tor network as a whole more resilient against this kind of attack.

KEYWORDS

Tor, DDoS, Networks, Anonymity

1 INTRODUCTION

Distributed Denial of Service (DDoS) attacks — overloading a single system with a large amount of requests from many other systems — are common occurrences in today's Internet. Anonymity networks like Tor are impacted by this kind of attack in many different ways. First, attackers can launch their attacks via the network to conceal their identity. This approach is somewhat limited because the Tor network has limited exit bandwidth, so attacks launched in this way only work against systems that are relatively easy to overload. Second, attackers can target systems within the Tor network to disrupt individual services or the entire network. This kind of attack is especially problematic because most common defense mechanisms cannot be applied. Dropping traffic early is infeasible because every node only knows the next hop, not the ultimate destination so they cannot make any decisions about which packets to keep and which to drop. Similarly, just blocking the entire network an attack is coming from is either not possible because their IP addresses are hidden or not desirable because this would allow attackers to prevent other users on their network from connecting to Tor.

Attackers are obviously also aware of this problem and have a history of abusing it. A fairly recent type of attack started in July of 2022 [3, 6], where attackers began to overload regular relays by flooding them with large amounts of traffic/connections. The attacks have slowed down after several months but they did not stop completely and the motivations of the attackers have never been fully understood.

This work focuses on a recent (2024-02-14) DDoS attack targeting some of the Tor relays operated by the authors. Since other relay operators reported similar attacks at the same time[1], we believe that a thorough investigation of such attacks has value for the entire Tor community. Our contribution consists of documenting our process of investigating this attack, the useful information sources we identified, the estimated effort required to execute this attack and its consequences for the Tor network we hope to narrow down the potential motivations of the attackers. Hopefully, this information can be augmented by other Tor relay operators to extend our understanding of DDoS attacks against the Tor network. Without this understanding, it is impossible to have a productive debate on effective countermeasures against this kind of attack.

2 ATTACK DESCRIPTION

Understanding this attack would have been impossible without logging information. This analysis is based on three information sources:

(1) The system log of the underlying Linux OS,
(2) data logged via Tor's metrics port, and
(3) network flow data collected at the perimeter firewall of our university.

Note: We also considered the log written by the Tor process itself as a potential information source, but at log level INFO the Tor process did not log any information related to the attack.

2.1 Targeted Environment

The targeted relays:

relay1 = `C5BACF269EFDC7FE91356F2A98ADDAA49BD82EC`, and
relay2 = `746BE16C31791C9A7D15461F734F0C017DDC55EE`

are part of a family consisting of 50 relays in total operated for a research project on the hidden service directory (HSDir). Since those relays are running to join the HSDir, they do not advertise enough bandwidth to qualify as Guard nodes and they do not volunteer to become Exit nodes either. In order to qualify for the HSDir the relays advertise a bandwidth of 105 KiB and try to keep their uptime above 24 hours.

2.2 What happened

Based on the available information sources, it was possible to reconstruct significant parts of the attack. An overview over the most important aspects is provided in Table 1. Relay1 came under attack at 14:26:38 CET when the first attacking host opened a connection

Free and Open Communications on the Internet 2024 (2), 64–67

[1] https://flokinet.social/@frelsisbaratta/111974050182432141

Table 1: Overview: Attack Metrics

Metric	Relay1	Relay2
Packets Received	99 M	36 M
Bytes Received	4,7 GiB	1,8 GiB
TCP Sessions	338.351	251.920
Packets/s	790	474
Packets/session	315	176
Attack Start	14:26:38	14:54:33
Attack End	15:20:55	15:22:50

to the target relays OR port. The attack continuously ramped up over 16 minutes until a total of 46 hosts were flooding the relay with 338.351 TCP connections sending in total about 4.7 GiB of traffic over about an hour. While the attackers did manage to temporarily saturate our network link, this only happened during short spikes and should not have disrupted our relay. As a matter of fact, some other relays running on the same physical machine did not experience any disruptions during the attack indicating that network saturation was not the intention of this attack.

Nevertheless, the attack was successful as the targeted relay restarted at 14:45:13 CET after being killed by the Linux out-of-memory killer (OOM-Killer). Unfortunately, we have no concrete data on our relay's memory consumption, but considering the available memory and the OOM-Killers configuration the victim relay had to allocate more than 4 GiB of memory before this event could happen. The attack ended abruptly at 15:20:27 CET when all 45 attacking hosts stopped creating new connections within just 19 seconds.

The attack on relay2 started at 14:54:34 CET, while the first attack was still ongoing. The same 46 hosts responsible for attacking relay1 started connecting to relay2 as well. What was different however was the ramp up time, which reduced from 16 minutes to only 12 seconds. This combined attack managed to quickly exhaust our systems resources again causing both relay2 and relay1 to reboot at 15:00:10 and 15:00:57 CET. Attacks against relay2 continued until 15:22:34 when attacks ended within just 17 seconds.

After 15:22:51 no further connection attempts from the attacking hosts were logged, indicating that the hosts were either disabled or moved on to other targets.

3 ATTACK ANALYSIS

This section presents which conclusions we could make about the attack and what information enabled us to do so:

3.1 Attacker Identification

The first task was to identify the hosts responsible for overloading our system. In this regard the fact that both relay1 and relay2 are only used as middle nodes in Tor circuits is really useful because this means that incoming TCP connections should mostly come from other relays in a currently valid consensus. The only exception would be Bridges who can also work as guard nodes and are not included in the consensus. Therefore we filtered the list of hosts with active connections to our relays by two criteria:

(1) No IP addresses included in the Tor consensus.
(2) No IP addresses creating less than 1000 TCP connections per hour.

These criteria resulted in the final list of 46 hosts whose congruent timing behavior discussed in section 2 strongly indicates that they were all operated by the same entity. Another noteworthy point of congruency is that all 46 relays seemed to be hosted at Linode[2] based on their reverse DNS entries.

3.2 Attacker Intention

With the attacking hosts identified, the next question to answer is if this attack was intentional or unintentional. Since the number of attacking hosts was relatively low, it is possible that this was not an attack at all, but instead an experiment gone wrong or a configuration error in a cloud deployment. Without any information about the actual data sent by the attacking hosts, it is difficult to provide a definite answer to this question, but there are some relevant indicators. First, the network flow shows that the communication appeared valid to the relays. On average (mean) about 250 packets were exchanged for every TCP connection. If the attackers were not at least trying to complete a valid handshake, the relays would have dropped the connection without accepting so many incoming packets. This observation combined with the fact that our relays were sending responses to the attackers leads us to conclude that the attackers were using the Tor protocol and therefore intentionally targeting Tor relays. We cannot definitively rule out that this attack was the result of a failed Tor experiment, but the fact that the individual(s) responsible for it have not reached out to the Tor network or affected relay operators strongly supports the assumption that this attack was intentional.

3.3 Attacker Motivation

Working with the assumption that the attack was executed intentionally, the main question to answer is what attackers were trying to achieve. There are many possible goals attackers could be trying to achieve by launching DoS attacks. For this analysis we considered the following potential motivations:

(1) Worsening the user experience of Tor users
(2) Selectively de-anonymizing Tor users
(3) Disabling specific onion services
(4) Manipulating the hidden service directory

3.3.1 Worsening user experience. This option can be dismissed with very high confidence. Attackers trying to disrupt the Tor network would focus their attention on guard and exit relays as they are usually the limiting factor for traffic within the Tor network. The fact that both attacks were stopped after a relatively short time also supports this conclusion. The attackers were apparently content with forcing a reboot of the relay, they had no interest in keeping it continuously unable to partake in the Tor network. This leads us to conclude that the reboot of the system was what the attack was intended to do.

3.3.2 Selective de-anonymization. Rebooting a relay stops all currently active circuits forcing other clients to rebuild their circuits with other middle nodes. If an attacker controls the guard node of

[2]https://www.linode.com/

a specific Tor user, they could be trying to reboot the middle nodes chosen by their target until they select a middle node also under the control of the attacker. The delayed attack against relay2 could be a hint for this being the case, but at the same time it seems more likely that an attacker going for this goal would cease the attack against relay1 immediately after the victim builds a new circuit. Without more information on which and how many other relays were targeted by the attackers, it is impossible to rule this out but we do not consider this option very likely.

3.3.3 *Disabling specific onion services.*

Disrupting active circuits also stops a relay from acting as introduction point for an onion service. The attackers might have intended to disrupt access to an onion service by attacking its introduction points. This type of attack would circumvent the DDoS protections introduced by Tor in 2023 [11] but it requires an attacker to constantly adapt to the currently used introduction points to remain effective. The fact that two of our 50 relays were targeted by the attackers makes this scenario quite unlikely but since onion services can pick up to 20 introduction points and many onion services are mirrored and available with multiple different onion addresses, it cannot be ruled out. Again, without additional information about which other relays were attacked, this theory can neither be confirmed nor dismissed.

3.3.4 *Manipulating the hidden service directory.*

Another consequence of forcing a reboot is changing the uptime of a relay, which is relevant for the assignment of two different flags: The HSDir flag is only assigned to relays with an uptime of more than 96 hours and the Guard flag where the uptime must be higher than the median uptime of other familiar routers. Forcing a reboot effectively removes both of these flags from the targeted relays for at least 96 hours. In our case, both of our relays lost their HSDir flag because of this attack. In theory, an attacker could have used this to influence which HSDir relay is responsible for different service descriptors. However, the HSDir is highly redundant and relays joining and leaving the HSDir with every new consensus is to be expected, so the potential value gained by attackers would be quite limited. At most, they could estimate how many users an onion service has and deny access to a fraction of its users.

3.4 Attack Method

Based on the assumption that the memory exhaustion and consequent reboot of our Tor relays was intended by the attacker, the question remains how the attackers managed to increase a relays memory consumption so significantly. While our data provides no answer to this question, we can make a few observations worth noting. First, the DDOS metrics reported by the Tor metrics port did not show anything indicating an attack, so the attack was not using a well-known method.

Second, the number of active circuits tracked via the Tor metrics port did not increase on our attacked relays. While there is a mechanism within Tor that mitigates DDoS attacks [9] by limiting the amount of TCP connections a single address is allowed to make (currently set to 50 connections), at least 2300 (46 hosts * 50 connections) additional connections should have been logged. A potential explanation for this could be that the attack created incomplete connections which already required the Tor process to allocate memory but were not yet counted as connections by Tor. This had the negative side effect that the attack is invisible to everyone relying on the information provided by the Tor process itself.

Third, all attacking hosts across all TCP connections consistently kept a packet size of exactly 50 bytes. This is noteworthy because Tor exchanges all messages in *cells* which have a fixed size of 514 bytes, so our attackers were not transmitting complete cells, but at most fractions of cells. The low Bytes-per-packet rate becomes even more significant when taking into account that mandatory IP [1] and TCP [2] headers already use up 40 of the available 50 bytes reducing the effective payload left for the Tor client to at most 10 bytes. At this rate transmitting just a single Tor cell would require at least 52 different TCP packets. This renders any form of communication extremely inefficient because the majority of network traffic is used for metadata not for actually relevant payloads. It seems unlikely that attackers preparing a targeted DDoS attack against multiple Tor relays would use a configuration that slows their systems down, unless the low packet size is a necessary component of the attack.

This leads us to theorize that the Tor client exhibits performance issues when dealing with very small packets, possibly containing highly fragmented cells. If the inbuilt DoS defense in Tor had worked as intended, the vast majority of these connections should have been dropped immediately. Instead, the attackers were able to send on average more than 100 packets per connection. Either the small packet size just allowed the attackers to send this many packets before the connection was torn down or the attackers actually managed to craft a payload that delayed the teardown of a connection and were therefore able to send so many packets. Unfortunately, we have no data on how the payload sent by the attackers looked like — as this information was not logged by our Tor relays — but the fact that our relays were sometimes responding to the attacking hosts leads us to speculate that they were actually sending valid packets, but in a highly fragmented way to delay the Tor process' decision whether to drop the connection or not. The memory exhaustion experienced by our relays indicates to us that either the connections were never dropped and consumed memory despite not resulting in active circuits or the connections were dropped but not without freeing all the memory allocated for them in the first place, resulting in a continuous increase of memory consumption. More research is needed to find out what the actual cause for the exploding memory consumption is. Based on our analysis, the attack seems to be surprisingly efficient. Considering that only 4.7 GiB of data were sent to relay1, and the relays memory consumption was beyond 4 GiB before it rebooted for the first time, it seems like the attackers could allocate almost 1 byte of memory on our relay for every byte they transferred over the network.

3.5 Attack Costs

Another question our analysis can answer is about the financial capabilities required to execute this attack. As discussed in section 3 all 46 attacking hosts were hosted at Linode and their pricing model is public[3]. Assuming that the attackers worked with simple virtual machines with dedicated CPU's, they attackers had to spend at least 0.054 $ per hour. The attack against our relay (46 attacking hosts for 56 minutes) cost the attackers less than 2.5 $. However, since our logs show that the 46 relays were attacking multiple relays in parallel, the actual costs per attack were probably even lower.

4 LESSONS LEARNED

With our current understanding of the attack we feel confident to say that properly configured firewall rules that deny too many concurrent connections from other hosts would have stopped this attack. There are community curated lists of firewall rules for relay operators [4, 6, 10] making it easy to implement this and make Tor relays more resilient against attacks. Unfortunately, applying these rules is not an official recommendation by the Tor project for relay operators, so a lot of relay operators are likely unaware of this option leaving a large part of the Tor network susceptible to this kind of attack. Integrating counter-DDoS measures like the deployment of firewall rules directly into the standard Tor relay installation process would significantly reduce the number of vulnerable relays within the Tor network.

As a prerequisite for a discussion about effective counter-DDoS measures, we need a better understanding of the reasons behind these attacks. The flow data collected about our relay proved extremely valuable for our analysis, but is again most likely not available to most other Tor relays. In contrast, the data logged via Tor's metrics port did not even show that an attack was taking place. Consequently, our relays never reported themselves as overloaded to the Tor network, making this attack invisible to observers relying on the information gathered by the Tor network [7, 8]. This is frustrating because it makes it impossible to accurately gauge which or how many other relays were targeted by the attackers and as discussed in section 3.2 this information is critical to narrow down what an attacker might have been trying to achieve with their attacks. However, collecting more information about attacks on relays without compromising the privacy of Tor users is not trivial and requires further work.

A final lesson we would like to share is that DDoS attacks against Tor relays also have a negative impact on how organizations view the operation of Tor relays. Our research institution was previously supportive of our operation of Tor relays, but the attack on February 14th had side effects that impacted other parts of our organization—overloading our perimeter firewall and disabling Internet access for everyone— and preventing such issues by discontinuing the operation of Tor relays was one of the solutions immediately suggested. If the operation of Tor relays becomes to associated with being targeted by DDoS attacks, recent efforts to get more organizations to support the Tor network [5] are unlikely to succeed. Making Tor relays in general more resilient against such attacks would be the easiest way to reduce the amount of DDoS attacks attempted against running relays and reduce the risk of side effects of these attacks impacting the organizations responsible for running them.

[3]https://www.linode.com/de/pricing/

ACKNOWLEDGEMENTS

Special thanks go to Stefan Kronawithleitner and the entire networks department of JKU for providing us with the flow data essential to this paper. This work has been carried out within the scope of Digidow, the Christian Doppler Laboratory for Private Digital Authentication in the Physical World. We gratefully acknowledge financial support by the Austrian Federal Ministry of Labour and Economy, the National Foundation for Research, Technology and Development, the Christian Doppler Research Association, 3 Banken IT GmbH, ekey biometric systems GmbH, Kepler Universitätsklinikum GmbH, NXP Semiconductors Austria GmbH & Co KG, and Österreichische Staatsdruckerei GmbH.

REFERENCES

[1] 1981. Internet Protocol. RFC 791. https://doi.org/10.17487/RFC0791
[2] 1981. Transmission Control Protocol. RFC 793. https://doi.org/10.17487/RFC0793
[3] Roger Dingledine. 2022. *reports that relays not obeying DoSConnectionMaxConcurrentCount.* The Tor Project. https://gitlab.torproject.org/tpo/core/tor/-/issues/40636
[4] Enkindu-6. 2022. *tor-ddos.* Tor Tor community. Retrieved 2024-04-06 from https://github.com/Enkidu-6/tor-ddos
[5] Thorin Klosowski. 2024. *Announcing the Tor University Challenge.* Electronic Frontier Foundation. https://www.eff.org/de/deeplinks/2023/08/announcing-tor-university-challenge
[6] Georg Koppen. 2022. *Provide a recommended set of iptables/nftables rules to help in case of DoS attacks.* The Tor Project. https://gitlab.torproject.org/tpo/community/support/-/issues/40093
[7] Karsten Loesing, Steven J. Murdoch, and Roger Dingledine. 2010. A Case Study on Measuring Statistical Data in the Tor Anonymity Network. In *Proceedings of the Workshop on Ethics in Computer Security Research (WECSR 2010)* (Tenerife, Canary Islands, Spain) *(LNCS)*. Springer.
[8] The Tor Project. 2010. *Tor Metrics.* Tor Tor Project. Retrieved 2024-04-06 from https://metrics.torproject.org/
[9] The Tor Project. 2018. *Tor network parameters.* Retrieved 2024-04-12 from https://spec.torproject.org/param-spec.html#dos
[10] toralf. 2022. *torutils.* Tor Tor community. Retrieved 2024-04-06 from https://github.com/toralf/torutils
[11] Pavel Zoneff. 2023. *Introducing Proof-of-Work Defense for Onion Services.* The Tor Project. Retrieved 2024-04-05 from https://blog.torproject.org/introducing-proof-of-work-defense-for-onion-services/