DetecTOR: Detecting Tor Abuse Traffic in real time

Jiahua Zhang
University of Iowa

Zain Khan
University of Iowa

Hammas Bin Tanveer
University of Iowa

ABSTRACT

The rampant use of Tor as a tool to attack internet services while maintaining anonymity, has led to severe differential treatment for Tor users by the online service providers who are affected by this abuse. In this study, we aim to detect such abusive traffic in real time while maintaining user anonymity by looking at flow characteristics of traffic which have been approved by the Tor ethics board. We extract these characteristics from traffic logs we collect in order to train a classifier which can detect attack traffic.

First we show that malicious and regular traffic are distinguishable using these characteristics for a baseline case (where Tor is not considered), indicating that malicious traffic has fundamentally different flow characteristics from regular traffic. Then, we show that this same detection is possible for the same traffic sent through Tor.

KEYWORDS

Networks, Privacy and Security, Tor, Abuse Mitigation

ACM Reference Format:
Jiahua Zhang, Zain Khan, and Hammas Bin Tanveer. 2019. DetecTOR: Detecting Tor Abuse Traffic in real time. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

In recent times, global internet censorship has been on a constant rise with newer countries partaking and deploying more sophisticated censorship techniques than ever, for example the great China Firewall [24] [11]. This coupled with the persecution faced by sensitive minorities and groups involved in sensitive reporting [5] has led to the widespread adoption of anonymity networks, especially Tor [19].
Tor has nearly 2 million daily users and is used the most in countries which are known for widespread internet censorship such as Iran [17]. Although Tor is such a vital tool for millions of individuals in these countries, whether used for whistle-blowing [14], anonymous journalism, or other important reasons, it is also used for malicious reasons. According to Cloudflare, almost 94% of all traffic going through Tor exit nodes to the Cloudflare servers is related to attacks like vulnerability scanning and 18% of world's total spam email is sent over Tor. [18]. Further studies suggest that almost 10% of all Tor traffic has resulted in triggering IDS alerts referencing serious attacks like Denial of service and spam [16].
This increase in the usage of Tor for malicious reasons has resulted in the degradation of its reputation leading to serious differential treatment, like CAPTCHAs or complete blocking of websites, for Tor users. Tor users face some sort of discrimination on up to 20% of Alexa top 500 websites and 7% of the blacklists that websites use to distinguish between normal and malicious traffic now proactively block IP addresses linked to Tor exit nodes. [21]. This has resulted in the degradation of the utility of Tor due to a differentiated experience of users accessing the web with and without Tor. Despite the reputation that Tor has built over the years, big internet players like Cloudflare have come to terms with the importance of Tor and have recently signed a deal with the anonymity network which would result in reducing differential treatment for Tor users on sites which use Cloudflare to serve their content [20]. This is high time when an improving reputation of Tor coupled with less malicious traffic from the network be used to raise awareness about the merits of Tor thereby increasing the acceptance.
In this paper, we aim to detect and block abuse traffic through Tor in real time without deanonymizing the user resulting in less malicious traffic exiting the Tor nodes, enhancing the Tor network's reputation and consequently providing comparable user experience to users accessing the internet without Tor.
Key findings - We find that using only flow based characteristics is sufficient for distinguishing regular Tor traffic from malign Tor network. Furthermore, this can also be achieved by looking only at the first 50 to 100 packets, so that it is possible for an exit relay to identify such attack traffic in real time. We also show that some classifiers perform this job better than others and which features are most important in identifying this abusive traffic.

Conference'17, July 2017, Washington, DC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

2 BACKGROUND AND RELATED WORK

Tor

Tor is often discriminated against due to the nature of the anonymity network. Tor is a low-latency anonymous communication network and one of the main aims of Tor is to provide unlinkability between two communicating entities or even between multiple connections to and from a single user [6]. Tor uses a distributed overlay network run by volunteers coupled with onion routing [9] to providing anonymity to its users using TCP services like web browsing.
When a user decides to use Tor for communicating, he/she makes an n-hop circuit to the destination, n being equal to 3 in the default case, to the destination; first hop being the guard relay, second hop being the middle relay and the last hop being the exit relay. Once the circuit is successfully established, every outgoing request that a client makes is observed as being originated from the last hop in the circuit, namely the exit relay as shown in figure 1.

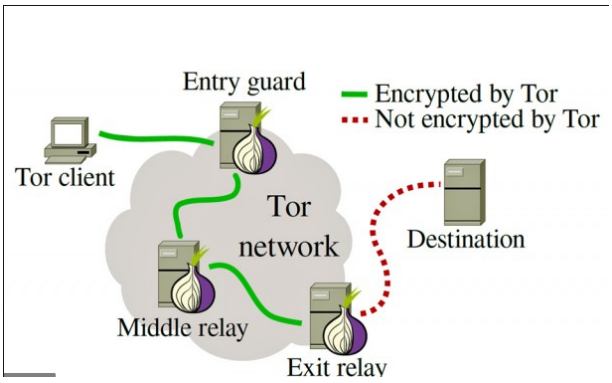


Figure 1: An example Tor circuit

An exit relay can be shared by many clients at once and hence comes the idea of shared reputation of Tor users. Even if one of the clients, out of potentially thousands, misuses the circuit, the reputation of all other clients using the same exit relay is tarnished. This resulting in differential treatment, like CAPTCHAs or even outright blocking for regular Tor users [21].
As the traffic is observed to be coming from the tor exit relay, the volunteers running these exits get into trouble for different kinds of malicious traffic exiting their relays [12]. Tor has provided some sort of authority to exit relay operators as to what sorts of traffic can exit their relays in the form of exit policies [23]. In this paper, we aim to increase the authority these exit relay operators have by defining ways in which malicious abusive traffic can be detected on Tor exit nodes and later be blocked.

Denial of Service Attacks against Tor

While there are many forms of abusive traffic, a significant portion of the traffic are DoS attacks [10]. Additionally, some other malicious behaviors, such as port scanning, generate traffic similar to that of DoS attacks. As such, we dedicate a discussion to DoS attacks as these are the primary attacks we expect to be able to detect. There are many different methods for carrying out a DoS attack. The most common method of attack occurs when an attacker floods a network server with traffic. In this type of DoS attack, the attacker sends several requests to the target server, overloading it with traffic [4]. In reference to Tor, this can take up multiple forms whether it is flooding the whole network so that the relays are not able to communicate with each other efficiently or whether Tor is used to conduct DoS attacks on servers outside the Tor network. There has been a lot of studies previously on attacks launched against the Tor network as a whole.
Borisov et al. [3] showed that if an attacker couples a good amount of relays with an ability to launch DoS attacks on good relays, clients would be forced to use relays used by these attackers which would place the adversary in a much better place to launch traffic correlation attacks [13] [22].
Barbera et al. [2] discuss a new form of DoS attack which is particular to the way that Tor creates its circuits. In this attack, a flood of CREATE cells are sent to the target relay which the relay tries to decrypt using expensive cryptographic functions and ends up becoming unusable for the network, overloading other relays in the process.
There has also been a lot of debate about the merits of Tor specially after it has been know to be used for a lot of DoS attacks on regular services, disrupting them substantially [7].

Traffic Analysis on Tor

Our goal of identifying malicious traffic going through Tor is similar to previous research on traffic analysis on Tor. Dingledine et al. [6] describe certain characteristics of Tor's traffic that make it difficult for adversaries to conduct attacks like traffic analysis. Tor does not depend on any sort of packet mixing, padding or shaping and top of that, many TCP streams, of many different clients, can use a single circuit. Furthermore, each Onion router makes a TLS connection with the next and each Onion router communicates using standard 512 bytes cells, through which all TOR traffic passes, making all Tor traffic look alike.
Having those properties can make traffic analysis a non-trivial task, as only properties that are now useful for any sort of traffic analysis are timing related properties and that is what researchers have used.
Lashkari et al. [15] take into consideration timing based factors and try to distinguish between real traffic and TOR based

traffic using the same applications like Voice over IP, web browsing, video streaming etc. They figure out that using strictly timing based features, they can, to some extent, distinguish TOR traffic from real traffic. Our study differs from there's by the fact that we are using timing based features too but to distinguish between Tor traffic of different kinds, normal and malicious.

3 METHODOLOGY

Setup

We look for normal and malicious data for two different situations: without Tor and with Tor. By classifying the normal data against the malicious data without Tor, we obtain a baseline accuracy and find the important features identifying malicious traffic. We then repeat the classification for the normal and malicious traffic with Tor to test our hypothesis that malicious traffic is still identifiable in Tor.

We test the ability for our classifiers to detect attacks in real time by training classifiers on partial flows. A flow consists of a sequence of packets with the same source IP, destination IP, source port, destination port, and protocol. For a partial flow, we consider only the first n packets of each flow. We test the values n = 10, 50, 100, and 150.

Collecting Data

We looked for network traffic in the form of PCAP files. One source of DoS traffic that we used was the 2017 CIC DoS dataset. This dataset consists of 24 hours of network traffic containing four different types of slow DoS attacks to 10 web servers [8].

For comparison, we looked at normal traffic routed through the private Tor network, to obtain a baseline. This data was collected from traffic captures on Wireshark's website.

Extracting Features

We used the CICFlowMeter tool to parse the PCAP files, generates bidirectional flows, and extract flow level features from the PCAPs [1]. In addition, the CICFlowMeter tools computes additional time related features for the flows, such as the average time between two packets in the flow.

Important Features

The data that we look at contains 80 network traffic features extracted from using the CICFlowMeter tool [1]. The important features we looked at were the following:

- **Total Bwd packets** - Total packets in the backward direction
- **Bwd Packet Length Mean** - Mean size of packet in backward direction
- **Subflow Fwd Packets** - The average number of packets in a sub flow in the forward direction
- **Fwd Packet Length Mean** - Mean size of packet in forward direction
- **Fwd IAT Min** - Minimum time between two packets sent in the forward direction
- **Idle Min** - Minimum time a flow was idle before becoming active
- **Init Win bytes forward** - The total number of bytes sent in initial window in the forward direction
- **Flow IAT Min** - Minimum time between two packets sent in the flow

Classification Methods

In order to achieve a robust detection method, we looked at multiple classifiers with varied decision boundaries. This includes a Gaussian Naive Bayes, a Support Vector Machine (SVM), a Decision Tree, and a Mutlilayer Perceptron (MLP). These were all implemented using the Scikit-learn Python library for classification.

Although Naive Bayes is widely used in classification tasks and can achieve outstanding results, it assumes that features are independent of one another. This is clearly not the case for this kind of data, as some of the features being considered are means, minimums, and maximums (such as Fwd Packet Length Mean, Fwd Packet Length Min, and Fwd Packet Length Max). It is clear that the mean is not independent of the other two. Network traffic is also inherently independent of all of its features, as bytes sent per second is related to the total number of bytes sent in an interval and the time between sending packets. Due to these discrepancies, we did not have high hopes for this classifier.

The SVM can make complex boundaries and we had high expectations for it given the copious amount of features it could make said decision boundaries with. Used as a binary classifier, we felt that any underlying pattern of data that separated malicious and regular traffic would be found out with this.

Decision Trees are akin to SVM's in that they choose to split where the maximum instances of each class can be separated. Therefore, splitting by features until all the data is separated as well it can be can provide insight into the features that decide malicious versus regular traffic.

Finally, MLP had to be included due to the performance of neural networks in classification tasks. Since there is a hidden layer between the input and the output, the Universal Approximation Theorem states that this neural network should be able to learn any continuous function that would define if data is malicious or regular.

An LSTM deep learning classifier was considered due to its impressive performance in time series classification tasks, although this kind of classifier necessitates much more data than the other models. This model was replaced by the MLP mentioned above due to the similar non-linear boundary

curves we can achieve with neural networks and deep learning networks while not being a massive model that required order of magnitude of more data in order to run well. A preliminary run with this kind of model on an equally sized dataset produced inferior results to the MLP classifier and we chose to not continue with it in further evaluations.

Metrics

To measure the performance of our classifiers we will consider the following metrics:

- `Accuracy`
- `F1-Score`

Given the confusion matrix for a classifier with True Positive (TP), False Positive (FP), True Negative (TN), and False Negative (FN), these metrics are defined as follows:

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN}$$

$$Precision = \frac{TP}{TP + FP}$$

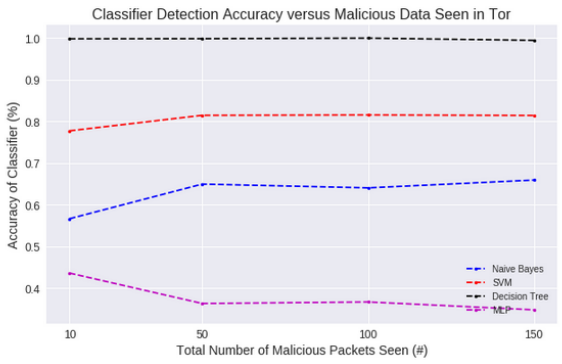
$$Recall = \frac{TP}{TP + FN}$$

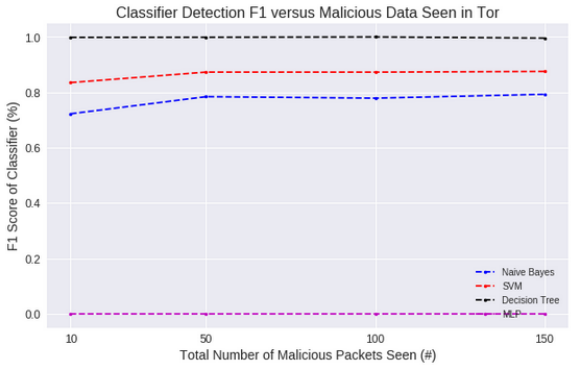
$$F1 - Score = 2 * \frac{Precision * Recall}{Precision + Recall}$$

4 EVALUATION

Baseline

In order to support our hypothesis that detection of malicious traffic versus regular browsing traffic can be done in Tor, we first took a look at regular traffic versus abuse traffic in general (without Tor). The data put into the models includes sample Wireshark internet browsing traffic as the good data and a combination of HTTP Flood and Slowloris DoS attacks for the malicious traffic. Despite the completely different nature of the two DoS attacks we used to train the models, we can see that they are fundamentally different from regular network traffic patterns.





We can tell that it is possible to differentiate DoS traffic from regular traffic given that our classifiers are not simply guessing and are significantly above 50 percent. There is an exception with the Multilayer Perceptron which is behaving unexpectedly terribly. We postulate that this is due to a lack of training data and the weights have not stabilized. Given more data we know that neural networks are able to approximate functions well, but the flow splitting procedure we used resulted in less data than when we were not splitting, causing this bad result. Looking further into the decision tree and plotting its decision criteria yields the following tree. The stellar performance was not expected, and is indicative of over-fitting.

This is the decision tree (where class 0 is regular and class 1 is malicious):

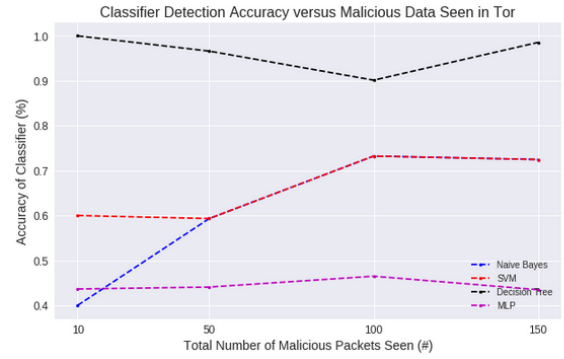
```
|--- Init Bwd Win Byts <= 242
|   |--- Init Bwd Win Byts <= 27
|   |   |--- class: 0.0
|   |--- Init Bwd Win Byts >  27
|   |   |--- Flow Byts/s <= 3334
|   |   |   |--- Bwd Pkt Len Max <= 483
|   |   |   |   |--- Flow IAT Min <= 8014
|   |   |   |   |   |--- Bwd IAT Min <= 4
|   |   |   |   |   |   |--- Bwd Seg Size Avg <= 28
|   |   |   |   |   |   |   |--- class: 1.0
|   |   |   |   |   |   |--- Bwd Seg Size Avg >  28
|   |   |   |   |   |   |   |--- class: 0.0
|   |   |   |   |   |--- Bwd IAT Min >  4
|   |   |   |   |   |   |--- class: 1.0
|   |   |   |   |--- Flow IAT Min >  8014
|   |   |   |   |   |--- class: 0.0
|   |   |   |--- Bwd Pkt Len Max >  483
|   |   |   |   |--- class: 0.0
|   |   |--- Flow Byts/s >  3334
|   |   |   |--- class: 0.0
```

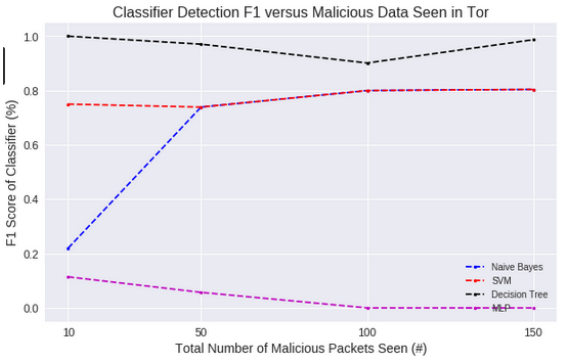
```
|--- Init Bwd Win Byts >  242
|   |--- Idle Min <= 1415124346732544
|   |   |--- class: 0.0
|   |--- Idle Min >  1415124346732544
|   |   |--- class: 1.0
```

As one would expect, features that differentiate the traffic between regular and malicious include total number of bytes being sent per second (as seen by Flow Byts/s), but other features are important in distinguishing the two, such as Flow IAT Min, which is the minimum time between two packets sent in the flow of traffic. The features that the decision tree viewed as important can be clearly seen to differentiate DoS attacks (both high volume HTTP floods and low volume Slow Loris attacks alike) and regular browsing traffic and give importance to the features that define the two despite the apparent over-fitting visible from its results (despite pruning efforts).

Tor

Now we look at the same type of traffic, but with data collected through Tor network.





As before, we see that abuse traffic has different characteristics, making it distinguishable from regular traffic from the perspective of an exit node. Looking at the decision tree, we can see what factors are most important for determining whether traffic is good or malicious. As before, in the decision tree, class 0 represents regular traffic and class 1 represents malicious.

```
|--- Idle Max <= 1369039146319872
|   |--- Bwd Pkts/s <= 60662
|   |   |--- Init Bwd Win Byts <= 64888
|   |   |   |--- Init Bwd Win Byts <= 3196
|   |   |   |   |--- Init Bwd Win Byts <= 8
|   |   |   |   |   |--- class: 1.0
|   |   |   |   |--- Init Bwd Win Byts >  8
|   |   |   |   |   |--- class: 0.0
|   |   |   |--- Init Bwd Win Byts >  3196
|   |   |   |   |--- class: 1.0
|   |   |--- Init Bwd Win Byts >  64888
|   |   |   |--- class: 0.0
|   |--- Bwd Pkts/s >  60662
|   |   |--- Flow IAT Max <= 4
|   |   |   |--- class: 1.0
|   |   |--- Flow IAT Max >  4
|   |   |   |--- class: 0.0
|--- Idle Max >  1369039146319872
|   |--- class: 0.0
```

We see that `Flow IAT Max` is a significant feature in classifying between good and malicious traffic, very similar to the `Flow IAT Min` in the first classifier. However, we also note that we have 85 data points for good data and 435 data points for bad data. This imbalance in the data suggests that these results should be taken with a grain of salt, as a classifier simply predicting malicious will be accurate most of the time. Although, we do obtain accuracy greater than a ZeroR classifier that would predict the majority class with our classifiers. We see greater returns in classifier performance after seeing more data as one would expect, although the graph of the data seems to say that 100 packets is the point where accuracy peaks (as seen by the SVM and Naive Bayes) without incurring the cost of looking at too much data. Seeing as flows can be hundreds of packets long, this seems to be a fraction that exit relays might be able tolerate before attacks are fully realized.

5 CONCLUSION AND FUTURE WORK

In this work, we looked at abuse traffic in order to determine how accurately this can be detected via machine learning methods. We found that a Naive Bayes classifier can achieve 60% accuracy after seeing 50 packets in a flow and 70% accuracy after seeing 100 packets. While we restricted the abuse

attacks we looked at to DoS and other typical botnet behavior, we found that these attacks have significant differences from normal traffic and that these differences are noticeable to an exit relay. Looking at the features identified by a decision tree, we found that minimum and maximum time between packets sent is an important feature for this classification.

For future work, we would like to make a modified version of Tor which observes the traffic passing through it in real time and makes a decision based on the traffic it sees whether the traffic is malicious or not. Our study currently is only focused on specific abuse traffic in our limited dataset; in future it can be extended to other abuse such as network layer attacks, spam, and port scanning. Ways of blocking traffic after recognizing abusive traffic without deanonymizing any user using the specific exit can also be a future area of research. We would also like to deploy an exit relay with our classifier and run a measurement study so we can see how well it performs out in the wild.

REFERENCES

[1] ahlashkari. 2019. *CICFlowmeter*. https://github.com/ahlashkari/CICFlowMeter

[2] Marco Valerio Barbera, Vasileios P Kemerlis, Vasilis Pappas, and Angelos D Keromytis. 2013. CellFlood: Attacking Tor onion routers on the cheap. In *European Symposium on Research in Computer Security*. Springer, 664–681.

[3] Nikita Borisov, George Danezis, Prateek Mittal, and Parisa Tabriz. 2007. Denial of service or denial of security?. In *Proceedings of the 14th ACM conference on Computer and communications security*. ACM, 92–102.

[4] CNN. 2015. *Understanding Denial-of-Service Attacks.* https://www.us-cert.gov/ncas/tips/ST04-015

[5] CNN. 2018. *Journalist jailed by Iran talks about how Anthony Bourdain changed his life.* https://money.cnn.com/2018/06/10/media/anthony-bourdain-jason-rezaian/index.html

[6] Roger Dingledine, Nick Mathewson, and Paul Syverson. 2004. *Tor: The second-generation onion router.* Technical Report. Naval Research Lab Washington DC.

[7] Doug Drinkwater. 2015. *Hackers route via Tor for stealthy 'slow-death' DoS attacks.* https://www.scmagazineuk.com/hackers-route-via-tor-stealthy-slow-death-dos-attacks/article/1478453

[8] Canadian Institute for Cybersecurity. [n.d.]. *CIC DoS dataset (2017).* https://www.unb.ca/cic/datasets/dos-dataset.html

[9] David Goldschlag, Michael Reed, and Paul Syverson. 1999. *Onion routing for anonymous and private internet connections.* Technical Report. NAVAL RESEARCH LAB WASHINGTON DC CENTER FOR HIGH ASSURANCE COMPUTING SYSTEMS ....

[10] Katerina Goseva-Popstojanova, Goce Anastasovski, Ana Dimitrijevikj, Risto Pantev, and Brandon Miller. 2014. Characterization and classification of malicious Web traffic. *Computers Security* 42 (2014), 92–115.

[11] Guardian. 2018. *The great firewall of China: Xi Jinping's internet shutdown.* https://www.theguardian.com/news/2018/jun/29/the-great-firewall-of-china-xi-jinpings-internet-shutdown

[12] Taylor Hatmaker. 2017. *Tor node operator arrested in Russia will be held on terrorism charges until June trial.* https://techcrunch.com/2017/04/24/dmitry-bogatov-tor-russia/

[13] Aaron Johnson, Chris Wacek, Rob Jansen, Micah Sherr, and Paul Syverson. 2013. Users get routed: Traffic correlation on Tor by realistic adversaries. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*. ACM, 337–348.

[14] Adrienne LaFrance. 2012. *The Tor Project helps journalists and whistleblowers go online without leaving a trace.* https://www.niemanlab.org/2012/06/the-tor-project-helps-journalists-and-whistleblowers-go-online-without-leaving-a-trace/

[15] Arash Habibi Lashkari, Gerard Draper-Gil, Mohammad Saiful Islam Mamun, and Ali A Ghorbani. 2017. Characterization of Tor Traffic using Time based Features.. In *ICISSP*. 253–262.

[16] Zhen Ling, Junzhou Luo, Kui Wu, Wei Yu, and Xinwen Fu. 2014. TorWard: Discovery of malicious traffic over Tor. In *IEEE INFOCOM 2014-IEEE Conference on Computer Communications*. IEEE, 1402–1410.

[17] 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) (LNCS)*. Springer.

[18] Matthew Prince. 2016. *The Trouble with Tor.* https://blog.cloudflare.com/the-trouble-with-tor/

[19] Tor Project. 2019. *The Tor Project | Anonymity Online.* https://www.torproject.org/

[20] Mahrud Sayrafi. 2018. *Introducing the Cloudflare Onion Service.* https://blog.cloudflare.com/cloudflare-onion-service/amp/

[21] Rachee Singh, Rishab Nithyanand, Sadia Afroz, Paul Pearce, Michael Carl Tschantz, Phillipa Gill, and Vern Paxson. 2017. Characterizing the nature and dynamics of Tor exit blocking. In *26th {USENIX} Security Symposium ({USENIX} Security 17)*. 325–341.

[22] Paul Syverson, Gene Tsudik, Michael Reed, and Carl Landwehr. 2001. Towards an analysis of onion routing security. In *Designing Privacy Enhancing Technologies*. Springer, 96–114.

[23] TOR. 2018. *Reduced Exit Policy.* https://trac.torproject.org/projects/tor/wiki/doc/ReducedExitPolicy

[24] Jonathan L Zittrain, Robert Faris, Helmi Noman, Justin Clark, Casey Tilton, and Ryan Morrison-Westphal. 2017. The shifting landscape of global internet censorship. *Berkman Klein Center Research Publication* 2017-4 (2017), 17–38.