Know Your Victim: Tor Browser Setting Identification via Network Traffic Analysis

Chun-Ming Chang
National Taiwan University
Taipei, Taiwan
r08922004@csie.ntu.edu.tw

Hsu-Chun Hsiao
National Taiwan University
Taipei, Taiwan
hchsiao@csie.ntu.edu.tw

Timothy Lynar
University of New South Wales
Canberra, Australia
t.lynar@adfa.edu.au

Tatsuya Mori
Waseda University
NICT/RIKEN AIP
Tokyo, Japan
mori@seclab.jp

ABSTRACT

Network traffic analysis (NTA) is widely researched to fingerprint users' behavior by analyzing network traffic with machine learning algorithms. It has introduced new lines of de-anonymizing attacks [1] in the Tor network, inclusive of Website Fingerprinting (WF) and Hidden Service Fingerprinting (HSF). Previous work [4] observed that the Tor browser version may affect network traffic and claimed that having identical browsing settings between the users and adversaries is one of the challenges in WF and HSF. Based on this observation, we propose a NTA method to identify users' browser settings in the Tor network. We confirm that browser settings have notable impacts on network traffic and create a classifier to identify the browser settings. The classifier can establish over 99% accuracy under the closed-world assumption. The open-world assumption results indicate classification success except for one security setting option. Last, we provide our observations and insights through feature analysis and changelog inspection.

CCS CONCEPTS

• **Networks** → **Web protocol security**; **Network privacy and anonymity**; *Security protocols.*

KEYWORDS

anonymity, privacy, network traffic analysis, Tor

ACM Reference Format:
Chun-Ming Chang, Hsu-Chun Hsiao, Timothy Lynar, and Tatsuya Mori. 2022. Know Your Victim: Tor Browser Setting Identification via Network Traffic Analysis. In *Companion Proceedings of the Web Conference 2022 (WWW '22 Companion), April 25–29, 2022, Virtual Event, Lyon, France.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3487553.3524244

1 INTRODUCTION

Tor is a well-known censorship circumvention application and has gained popularity with more than two million daily users since 2018. One of its features is to prevent adversaries from identifying the websites users visit. Researchers have proposed NTA-based de-anonymizying attacks including WF [3] and HSF [6]. To conduct these attacks, local passive adversaries such as the governments and the Internet Service Providers (ISP) eavesdrop network traffic sent between the users and Tor guard nodes, extract traffic features, and identify the websites users visit with classifiers. WF and HSF leverage the fact that each website's traffic pattern varies, so the classifiers can identify the website given the traffic features.

Although previous studies have demonstrated high WF and HSF accuracy [3, 6], several assumptions such as network condition and browser settings [4] limit their threats in the real world. Different network condition and browser settings will influence traffic features and result in mis-classification. Juarez et al. [4] conducted the cross-setting examination on the Tor browser version. They collected website traffic using Tor browser versions 2 and 3 and trained a WF classifier on the dataset collected by one Tor browser version and tested its accuracy on the other. The results show that website-classification accuracy declines drastically if Tor browser version inconsistency exists between training and testing dataset, implying that the browser settings used by the adversaries to collect the dataset should be identical to the users' browser settings. However, no further explanation on this observation is provided, and no other browser settings such as security settings are examined.

Based on this observation, we propose the NTA-based browser setting identification. We hypothesize that the browser settings have notable effect on network traffic such that the adversaries can identify users' browser settings merely through NTA in the Tor network. NTA-based browser setting identification is not only a possible way to tackle the browser setting challenge in WF and HSF but also an alternative to fingerprint users' browser setting without JavaScript execution [5]. In this preliminary research, we investigate Tor browser version and security setting[1] to explain how these factors affect the network traffic patterns. Although Tor browser enables auto-update by default, some users tend to disable this feature due to fear of breaking existing features and settings, inadequate update details, and the annoyance update prompt [2].

We systematically address the following questions:

RQ1: Will Tor browser settings affect network traffic?
RQ2: Can Tor browser settings be identified from network traffic?
RQ3: Why do different Tor browser settings have distinguishable traffic features?

To answer RQ1, we conduct cross-setting examination to verify if network traffic differences originated from browser settings are distinctive enough to cause mis-classification. We find that when the training and testing network traffic datasets are collected with different browser settings, the website classification accuracy drops

[1] we refer to Tor browser version and security settings as browser settings.

WWW '22 Companion, April 25–29, 2022, Virtual Event, Lyon, France

ACM ISBN 978-1-4503-9130-6/22/04…$15.00
https://doi.org/10.1145/3487553.3524244

more than 50%. To answer RQ2, we create browser-setting classifiers to identify the Tor browser version and security setting given the collected network traffic. Because users change their browser settings infrequently, the classifier can make multiple predictions and combine the results to identify browser settings. Under the closed-world assumption, the version classifier requires seven predictions to achieve 99% accuracy, and the security setting classifier requires 19 predictions. Under the open-world assumption, the classifiers can successfully identify the browser version but fail to identify the safer level security setting. To answer RQ3, we analyze the features from the browser-setting classifiers in RQ2 and record the changelog and updates to explain our observations.

In this research, we examine Tor browser version and security setting and publish our code and dataset on Github[2]. Future research can apply a similar approach to analyze other browser settings or the combination of multiple settings. Also, more sophisticated learning algorithms and feature engineering can be conducted to improve the accuracy of browser setting identification.

2 BACKGROUND AND RELATED WORK

Tor Tor is a low-latency anonymity network consisting of about 6,000 relays around the world. Generally, users install a Tor Browser Bundle (TBB) to access the Tor network. A TBB comprises a Firefox browser and a Tor proxy. Tor introduces several mechanisms to ensure anonymity, including layers of encryptions, data segmentation, and randomized pipelining. Also, Tor browser supports three levels of security setting that limits the browser's feature to a certain level. Table 4 lists the security levels and corresponding changes.

Network Traffic Analysis (NTA) Targeting Tor Current NTA research in Tor mostly focuses on WF attacks [3] and HSF [6]. In both attacks, local and passive adversaries predict the websites or hidden services users visit through classifiers. Each Web page and hidden service yields different network traffic patterns, so the adversaries can fingerprint each web page and service based on the network traffic features. HSF specifically aims at fingerprinting hidden services, and WF attack aims at fingerprinting the web pages. Common traffic features [7] are packet flow statistics (e.g., number of packets) and time-based packet features (e.g., packet flow duration). However, NTA is more challenging in the Tor network because of the privacy enhancements that conceal traffic features.

To evaluate a classifier, prior work in WF often considers two assumptions based on the attacker's knowledge, the closed-world assumption and the open-world assumption. The closed-world assumption assumes the attacker knows the exact set of websites the users may visit. In this case, the attacker can train a website classifier within this set of websites. The training and testing datasets comprise traffic instances from an identical website set. On the other hand, the open-world assumption assumes a more probable condition in the real world: the attacker knows only a subset of websites the users may visit. Thus, some websites visited by the user are excluded from the training dataset.

[2]https://github.com/csienslab/torbrowser-nta

3 METHODOLOGY

3.1 Crawler Implementation

The crawler is implemented with an open-source library, tbselenium. We follow prior work's crawling approach [8]. The crawler visits website list in a Round-Robin fashion, and a tcpdump process is forked to collect the network traffic during each visit. To avoid browser cache and using fixed circuit, we rebuild the Tor circuit after visiting every website once in a round and restart browser after every visit. We set up multiple crawlers, each with a specific browser setting, and crawl the website list in parallel. To isolate each crawling process, the crawler is executed in the docker container.

3.2 Data Collection

Prior work [8, 9] did not collect traffic dataset using multiple browser settings, so we collect our own datasets to compare traffic features across browser settings. The datasets are collected under closed-world and open-world assumptions, and for each assumption we collect version dataset and security setting dataset. The version dataset contains four TBB versions, and security setting dataset includes three security levels. As we only examine one variable at a time, we only consider standard security level in version dataset and TBB version 10 in security setting dataset. Thus, we end up having 14 datasets in total. Table 1 summarizes our datasets.[3]

Closed-world assumption The set of websites the user visits are known to the attackers. Prior work selects top 100 websites from popular website lists [9], we select the top 200 websites from the Tranco list [10] to be the closed-world dataset. To reduce bias caused by network instability, we crawl each website 250 times over a month. We obtain 180 websites after removing the outliers and failed websites.

Open-world assumption The set of websites the user visits are partially known or unknown to the attackers. We consider the worst case and assume the website set are entirely unknown to the attackers. Thus, the training and testing datasets are disjoint. The classifier is required to classify traffic of unseen websites to the correct browser settings. The open-world dataset contains top 7,000 websites from the Tranco list. We collect one traffic instance for each examined browser setting.

Table 1: Traffic dataset

Dataset	Settings	Assumption	# of sites	# of instances per site
Version	7, 8, 9, 10	Closed-world	180	250×4
		Open-world	7,000	1×4
Security setting	standard safer, safest	Closed-world	180	250×3
		Open-world	7,000	1×3

3.3 Feature Selection and Model Training

We combine the features used in previous WF research [3, 7]. These features can be grouped into three categories. Table 2 summarizes the 117 features used in our study.

We use random forest for classification and describe the training and testing data under the two different assumptions. Under the closed-world assumption, we split the closed-world dataset into a training dataset and a testing dataset. Under the open-world assumption, we train the classifiers using the closed-world dataset and validate its accuracy using the open-world dataset.

[3]The safer security level comprises only 85 web pages due to lots of page load timeouts. We will discuss this in Section 5.3.

Table 2: Feature Set Summary

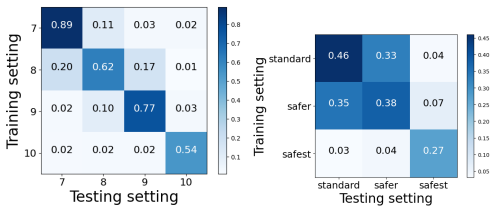
Category	Feature Type	Description	No.
Cell feature	Packet Count	Number and ratio of incoming and outgoing packets	9
	Packet Order [3]	For each packet, we record total number of packets sent or received before it	4
	Packet Concentrate [3]	Group Packet into chunks and extract features	8
Time feature	Packet Timestamp	Packet timestamp and packet per second	15
	Interval Time	Interval time between packet	21
	Wavelet Transform	Wavelet transform features	30
Flow feature	Flow Count	Number of traffic flows	5
	Flow Duration	Duration of traffic flow	19
	Outgoing Packet Burst	consecutive outgoing packets	6

4 EVALUATION

Using the collected datasets, we answer our research questions by conducting the cross-setting examination (RQ1, §4.1) and evaluating setting classifications (RQ2, §4.2).

4.1 Cross-setting Examination (RQ1)

We hypothesize that existing WF models cannot accurately classify traffic instances to the corresponding websites when the training and testing datasets are collected using inconsistent browser settings. To validate this hypothesis, we perform the cross-setting examination with a random forest model under the closed-world assumption. We train a WF classifier with one browser setting dataset and test its accuracy on another dataset. Figure 1 summarizes the results. When the training and testing datasets are collected with inconsistent settings, the WF attack accuracy drops significantly. The result confirms that browser settings influence traffic features.



(a) TBB version **(b) Security setting**

Figure 1: cross-setting examination

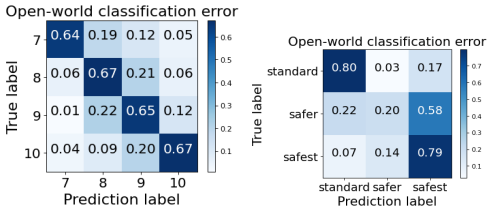
4.2 Version and setting classifier (RQ2)

To evaluate RQ2, we create random-forest-based classifiers to classify browser settings. The classifiers will take the traffic instance as input and predict the TBB version along with the security setting used to collect it. We consider two scenarios in the classification:

Closed-world Result We split the closed-world dataset into training and testing and train the classifiers using the training data. The version classifier achieves an 89% accuracy on the testing data, and the setting classifier achieves 77% accuracy on the testing data. The accuracy rate is sufficiently high in practice because a user will change browser settings infrequently, and the attacker can assume that consecutive n traffic instances have identical browser settings. The attacker can apply the classifiers on these instances and perform majority votes on the result to improve the overall accuracy. As n increases, the accuracy will grow close to 100%. The majority vote equation is $P = 1 - \sum_{i=0}^{\frac{n}{2}} C_{n-i}^{n}(1-p)^i p^i$.

The overall success rate, P, indicates the probability of getting more than $n/2$ correct predictions after n predictions. Suppose the single-prediction success rate is p. If p is higher than 0.5, the overall success rate will grow as n increases. Moreover, if p is closer to 1, P will converge to 1 more quickly. When $p = 0.89$ (the success rate in our experiment under the closed-world assumption for version classification), we can reach a 99% accuracy in seven predictions.

Open-world Result We train the version classifier and setting classifier with the closed-world dataset and test their accuracy with the open-world dataset. The version classifier reach 65% accuracy. The attacker can repeatedly apply the version classifier and reach 99% accuracy after 59 predictions. However, the accuracy for setting classifier only reach 60%, which means the security setting classifier cannot reach 99% accuracy within 100 predictions. Note that this work aims at evaluating the feasibility of setting identification through NTA and explain the observations, we do not invest in feature engineering or ML model optimization and only perform classification with random forest. We expect the accuracy can be improved further via sophisticated ML models and extra data-processing efforts. Moreover, through the confusion matrix in figure 2, we find that prediction errors in safer level accounts for most of the classification errors. Though current setting classifier cannot accurately distinguish three security levels, it can accurately distinguish the standard security level from others. The classification errors in the safer security level may be partially due to the disable of JIT compiler and will be discussed in Section 5.3.



(a) version classification **(b) setting classification**

Figure 2: Open-world setting classification results

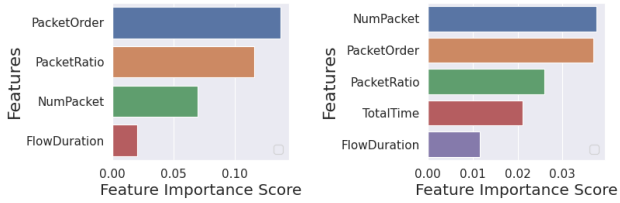
5 ANALYSIS (RQ3)

This section discuss why browser settings affect traffic features via feature analysis and changelog inspection.

5.1 Feature Analysis

We quantify features' influence by extracting the informative features from the version and security setting classifiers in Section 4.2 with scikit-learn library. The feature importance is calculated by normalizing each feature's gini index. Figure 3 lists the informative features whose normalized gini index is larger than 0.01.

The results indicate that cell features are more influential than time or flow features. Traffic features concerning the alteration in packet orders or quantity traffic features have apparent differences among browser settings. One possible explanation is that the packet arrival time and flow duration are mostly dependent on Tor's network condition. For the security setting classification, traffic features tend to be less informative. The most informative feature, which is the number of packets, accounts for 0.03 of importance. We suppose that the total time feature is relatively more informative in setting classifier is because the JIT compiler is disable in safer and safest security levels, which increase page load time.



(a) version classifier **(b) Security setting classifier**

Figure 3: Classifier's feature importance

5.2 Changelog Inspection

To uncover the root cause, we manually scrutinize the changelog of Tor proxy and Firefox and identify updates that may affect network traffic. Table 3 shows the summary of changelog inspection.

Table 3: Important updates for each TBB version

TBB version	Feature Updates		
	Component	Update	Type
7.5.2 - 8.0.6	Firefox	E10s development (ver. 52ESR)	*O*
		safe browsing update (ver. 56)	*S*
	Tor	bandwidth-limit refactor (ver. 0.3.4.1)	*O*
8.0.6 - 9.0.2	Firefox	TLS1.3 by default (ver. 61ESR)	*S*
		block ftp sub-resources (ver. 61)	*S*
		media encoding algorithm (ver. 67)	*O*
	Tor	circuit padding negotiation (ver. 0.4.0.4)	*O*
9.0.2 - 10.5.2	Firefox	disable TLS1.0, 1.1	*S*
		cache trusted Web PKI CA	*S*
		Flash Plugin content permission	*S*
		Service Worker and Push API enabled	*O*
	Tor	flow control (ver. 0.4.1.2)	*O*

S: Security policy update, *O*: Browser optimization

Security Policy Update Several security updates disable outdated cryptography algorithms. Furthermore, Firefox has set TLS1.3 by default since version 61 and has introduced several tracking protection configurations. Changes in the content-blocking policy may as well increase or decrease the amount of transmitted user information and influence traffic flows.

Browser Optimization Performance improvement may also affect the number of packets and traffic flow duration. For instance, Mozilla launched Electrolysis (E10s) to production in 2018. E10s was a multi-process system for improving the performance of web page loading. Firefox can better handle large chunks of web server responses in a shorter time, thereby altering the traffic features such as traffic bursts or numbers of packets in a fixed duration.

5.3 Security Setting Inspection

To compare the changes among security levels, we record the pref.js differences using diff command while setting security levels and study the firefox browser source code. Table 4 lists the safer and safest security levels and corresponding changes. One notable difference is the JIT compiler. JIT compiler effectively compiles JavaScript code to machine code, thus enhancing the JavaScript execution and page load performance. However, the JIT compiler introduces plenty of vulnerabilities and is disabled in the safer and safest security levels. Because disabling the JIT compiler extends the page load time, lots of page load timeout occurs when we collect the traffic with safer security level. We suppose this change attributes to the less traffic data and inaccuracy in safer security level.

Table 4: Disabled features in safer and safest security level

Configurations	Security level
javascript disabled on non https sites	*s*
javascript JIT compiler optimization disabled	*s*, δ
graphite, opentypesvg, and web assembly disabled	*s*, δ
media, audio, and webGL are click to play	*s*, δ
frame, fetch, webGL, object API, svg disabled	δ
javascript disabled on all sites	δ

s: safer level, δ: safest level

6 CONCLUSION

This study systematically investigates the effects of Tor browser settings on network traffic and discusses the root cause. We confirm that browser settings generate distinguishable traffic features and the browser setting identification is feasible when the attacker can continuously eavesdrop on traffic. Existing browser setting identification approach known as browser fingerprinting [5] require the users to visit attacker-controlled URLs. Our approach allows the adversaries to identify the settings merely through network traffic. This approach is likely to remain effective in the future because the traffic differences result from security and performance updates.

To reach the ultimate goal of NTA-based browser setting identification, an interesting future direction is to examine other browser settings or even the combination of multiple settings. Current approach examines one setting at a time, and it will be more practical to consider the effect of multiple settings on network traffic instead of a single setting. Another direction is to improve the setting classification accuracy by incorporating state-of-the-art learning algorithms and feature engineering. Random forest is useful for feature analysis but does not optimize the classification accuracy. Because we prioritize the feature analysis over accuracy, we do not utilize any deep learning model for classification.

ACKNOWLEDGMENTS

This research was supported in part by the Ministry of Science and Technology of Taiwan under grants MOST 109-2636-E-002-021 and 110-2628-E-002-002.

REFERENCES

[1] 2020. Attacks on Tor. https://github.com/Attacks-on-Tor/Attacks-on-Tor.
[2] 2020. How to shut firefox up about updates. https://support.mozilla.org/en-US/questions/1289766.
[3] George Danezis Jamie Hayes. 2016. k-fingerprinting: a Robust ScalableWebsite Fingerprinting Technique. In *Proceedings of Usenix Security Symposium.*
[4] Marc Juarez, Sadia Afroz, Gunes Acar, Claudia Diaz, and Rachel Greenstadt. 2014. A Critical Evaluation of Website Fingerprinting Attacks. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security.*
[5] Daniel Gruss Michael Schwarz, Florian Lackner. 2019. JavaScript Template Attacks: Automatically Inferring Host Information for Targeted Exploits. In *Proceedings of Network and Distributed System Security Symposium.*
[6] Rebekah Overdorf, Marc Juarez, Gunes Acar, Rachel Greenstadt, and Claudia Diaz. 2017. How Unique is Your .Onion? An Analysis of the Fingerprintability of Tor Onion Services. In *Proceedings of the ACM SIGSAC CCS.*
[7] Nicholas Hopper Shuai Li, Huajun Guo. 2018. Measuring Information Leakage in Website Fingerprinting Attacks and Defenses. In *Proceedings on ACM CCS.*
[8] Ian Goldberg Tao Wang. 2013. Improved Website Fingerprinting on Tor. In *Proceedings of ACM workshop on Workshop on privacy in the electronic society.*
[9] Rishab Nithyanand Tao Wang, University of Waterloo; Xiang Cai and Rob Johnson. 2014. Effective Attacks and Provable Defenses for Website Fingerprinting. In *Proceedings of Network and Distributed System Security Symposium.*
[10] Samaneh Tajalizadehkhoob Maciej Korczyński Wouter Joosen Victor Le Pochat, Tom Van Goethem. 2019. Tranco: A Research-Oriented Top Sites Ranking Hardened Against Manipulation. In *Proceedings of Network and Distributed System Security Symposium.*