國立臺灣大學電機資訊學院資訊工程研究所

碩士論文

Institute of Computer Science and Information Engineering

College of Electrical Engineering and Computer Science

National Taiwan University

Master Thesis

利用網路封包分析進行洋蔥網路瀏覽器設定偵測

Tor Browser Setting Identification via Network Traffic Analysis

張均銘

Chun-Ming Chang

指導教授: 蕭旭君 博士

Advisor: Hsu-Chun Hsiao Ph.D.

中華民國 111 年 9 月

September, 2022

doi:10.6342/NTU202202954

摘要

隨著網路封包分析技術的進步，一些學者將封包分析的技術運用在洋蔥網路上並提出兩種新攻擊，分別是網站辨識 (Website Fingerprinting, WF) 跟洋蔥網路匿名服務辨識 (Hidden Service Fingerprinting, HSF)。這兩種攻擊能夠讓攻擊者透過監聽使用者到洋蔥網路之間的封包，利用封包傳遞的統計資料和機器學習的演算法來預測使用者瀏覽的網站或洋蔥網路服務。這兩種攻擊對使用者的隱私匿名性造成威脅。

但 WF 和 HSF 目前仍然不構成大威脅，因為諸如瀏覽器的設定，網路狀況，使用者瀏覽習慣，這些原因都會影響到網站的網路封包的統計資料進而影響 WF 和 HSF 的準確率。之前的研究 [27] 觀察到洋蔥網路瀏覽器版本會影響到封包的特徵，並提出提升 WF 和 HSF 準確率的條件就是攻擊者必須跟使用者使用相同的瀏覽器版本。

基於以上的觀察，我們更進一步研究洋蔥網路瀏覽器的設定對於 WF 的影響。我們著重在於瀏覽器版本以及安全性設定這兩項瀏覽器設定進行研究。我們提出了一種基於網路封包分析的方法來辨識使用者的瀏覽器設定的攻擊 (BSF)，並實做了一個分類器透過網路封包特徵來偵測使用者的瀏覽器版本和安全性設定。BSF 的一個特色是，因為使用者不會很頻繁的更動瀏覽器的設定，因此 BSF 能夠透過多次的偵測來提高單一使用者瀏覽器設定的偵測準確度。在封閉世界的假設

doi:10.6342/NTU202202954

中，BSF 分類器能夠在使用者瀏覽七個網站後有高達 99% 機率能夠正確判斷瀏覽器版本，然後在開放世界的假設中則是需要使用者瀏覽 59 個網站後才能夠有 99% 準確率。關於安全性設定，在封閉世界假設中，使用者瀏覽 19 個網站後會有 99% 機率正確判斷安全性設定，在開放世界假設中，單次的安全性設定偵測只有 60% 準確率。最後我們進行封包特徵的分析找到重要的網路封包特徵，和研究洋蔥網路瀏覽器的更新紀錄跟安全性設定對瀏覽器功能影響，並且分析 BSF 預測中，準確率前 10 高跟準確率前 10 低的網站和這些網站的特色。我們發現最高級別的安全設定會把許多瀏覽器送出的請求擋掉，但預設跟第二級別的安全設定中，我們沒辦法從網路封包特徵中看出安全性設定造成的差異。在瀏覽器版本的分析中，我們發現單純從瀏覽器發出的請求類別看不太出不同版本間是否有顯著的差異，但可以從瀏覽網站中平均傳遞跟收到的位元數目看到一些不同版本間微小的差異。

根據我們的研究，我們提供洋蔥網路瀏覽器開發者跟一般的網站開發者一些抵抗 BSF 攻擊的建議。第一個方法是採用目前研究中針對 WF 的防禦方法，因為目前 WF 的防禦研究是隱藏原本網站的封包特徵，這樣的方法也適合拿來做為 BSF 攻擊的防禦機制。另一種方案是網站開發者可以採用一些針對大流量攻擊防護的服務，這些服務通常會產生一些具混淆性的封包跟暫時性的跳轉頁面，而額外的混淆性的封包跟跳轉頁面可以拿來隱藏原有網站的封包特徵，因而無法讓 BSF 的分類器蒐集到足夠的封包特徵和資訊做分類。

關鍵字：匿名性、隱私性、網路封包分析、洋蔥網路、網站特徵追蹤

doi:10.6342/NTU202202954

Abstract

The advance in Network Traffic Analysis (NTA) techniques has introduced new lines of de-anonymization attacks [4] against the Tor network, inclusive of Website Fingerprinting (WF) and Hidden Service Fingerprinting (HSF). These attacks can identify which regular websites or Tor hidden services a Tor user visited by using machine learning algorithms to analyze network traffic, thus undermining the privacy protection provided by Tor. However, WF and HSF are far from practical in the real world [27, 34] because the real-world traffic trace may be affected by not only the visited websites but other factors such as browser settings, network conditions, and users' access patterns. For example, previous work [27] observed that using different Tor browser versions might affect the network traffic and argued that one challenge in constructing an effective WF adversary is to ensure the adversary has the same browser version as the user's.

Inspired by their observation, in this work, we investigate the impact of browser settings, including the browser versions and security levels, on WF in the Tor network. After

doi:10.6342/NTU202202954

confirming that browser settings have substantial impacts on WF, we present a new NTA branch called *Browser Setting Fingerprinting (BSF)* and construct classifiers to identify a user's Tor browser version and security level. Interestingly, unlike WF and HSF, BSF can improve classification accuracy over time because users do not frequently change their browser settings. Our version classifier achieves over 99% accuracy when the user visits more than seven websites without changing the browser setting under the closed-world assumption and 59 under the open-world assumption. Our security-level classifier also achieves over 99% accuracy when the user visits 19 websites without changing settings under the closed-world assumption. However, the security-level classifier achieves 60% accuracy under the open-world assumption.

Last, we conduct an in-depth and comprehensive analysis to identify the most informative features, inspect the changelogs of Tor browsers, and investigate the root cause of the most/least accurate classification results. The safest security setting level significantly influences the number of JavaScript, font, and POST requests, while the standard and safer levels have indistinguishable traffic features. For the Tor browser version, we can only observe little traffic difference from the examined request content types among browser versions, but the average number of bytes sent and received shows the TBB version has observable difference.

doi:10.6342/NTU202202954

Based on our findings, we provide recommendations for browser developers and web developers to defend against BSF, WF, and HSF in general. The first approach is adopting the WF defense. Because WF defense aims at concealing websites' traffic features, it may be effective in BSF defense as well. For the second approach, web developers can adopt a cloud-based DDoS protection service to obfuscate the traffic patterns as it will redirect the visit to a temporary website, and this approach will be particularly useful if the attackers terminate the website visit before the DDoS protection actually redirect the visit to the targeted website because the attackers do not collect the website's traffic patterns in this case.

Keywords: anonymity, privacy, network traffic analysis, Tor, website fingerprinting

doi:10.6342/NTU202202954

doi:10.6342/NTU202202954

Contents

doi:10.6342/NTU202202954

doi:10.6342/NTU202202954

List of Figures

doi:10.6342/NTU202202954

doi:10.6342/NTU202202954

List of Tables

doi:10.6342/NTU202202954

doi:10.6342/NTU202202954

Chapter 1 introduction

Tor [18] is a well-known anonymity network and censorship circumvention application. It has gained popularity with more than two million daily users since 2018 [6]. One of its primary goals is to prevent adversaries from identifying the websites users visit, thereby improving user privacy and censorship resilience. While protected by layered encryption, previous research has demonstrated that adversaries can undermine Tor users' privacy through network traffic analysis (NTA) techniques, which analyze the traffic patterns rather than the packet content. Of particular interest are two NTA-based de-anonymization attacks, namely, website fingerprinting (WF) [24, 40] and hidden service fingerprinting (HSF) [29, 33], which aim to identify the websites or the hidden services a Tor user visited, respectively. These attacks consider local passive adversaries, such as governments or Internet Service Providers (ISP), that eavesdrop on the encrypted network traffic sent between the users and Tor guard nodes (i.e., the first node entering the Tor network). They leverage the fact that each website may have a different traffic pattern (e.g., packet count and the inter-packet time interval), so it is possible to construct classifiers to identify the website given its traffic features.

Although previous studies have demonstrated high WF and HSF accuracy [24, 33], many of them implicitly assume that the adversary and the user have identical browser settings [27], thereby limiting their threats in the real world. Juarez et al. [27] first pointed

doi:10.6342/NTU202202954

out that different Tor browser versions may influence traffic features and result in misclassification through the cross-version examination. To conduct a cross-version examination, they collected websites' network traffic using Tor browser versions 2 and 3, trained a WF classifier on the dataset collected by one Tor browser version, and tested its accuracy on the other. Their results show that website-classification accuracy declines drastically if Tor browser version inconsistency exists between the training and testing traffic dataset, implying that the browser versions used by the adversaries to collect the network traffic should be identical to the users' browser version. However, they did not investigate the root cause behind this misclassification, and thus it is unclear whether such misclassification will persist in the latest Tor versions, such as between Tor version 9 and 10. Moreover, even if we assume all users use the latest version of Tor, the traffic patterns may still be affected by user-configurable browser settings.

Based on the observation of Juarez et al., we hypothesize that Tor browser settings[1] have a notable effect on network traffic such that the adversaries can identify users' browser settings through passively eavesdropping network traffic in the Tor network. In this thesis, we consider two kinds of *browser setting*—Tor browser version and user-configurable security level—and investigate how these two affect the network traffic patterns. Then, we propose a new branch of NTA techniques called Browser Setting Fingerprinting (BSF) and investigate its feasibility. Although the 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 of update prompt [5]. Thus, users may use older versions of Tor. Also, the recent Tor browser support three security levels —standard, safer, and safest—for users who seek to defend against other security threats

[1]We focus on browser settings in this work because the traffic feature differences caused by browser settings have a critical impact on the WF accuracy, and the effect of browser settings on network traffic remains unclear.

doi:10.6342/NTU202202954

such as tracking by web advertisers.

We systematically address the following research 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?

In this work, we collect the traffic dataset to answer RQ1 and RQ2 and the root cause analysis (RCA) dataset to answer RQ3. We implement a crawler to collect our traffic dataset. The crawler visits the websites and executes tcpdump to collect the TCP packets of each website. The traffic dataset consists of a version dataset and a security-level dataset. For the version dataset, the crawler uses four different Tor browser versions to visit the websites and collects the TCP packets, and for the security-level dataset, the crawler installs Tor browser version 10 and visits the website with different security-level configurations to collect the TCP packets of each website. After collecting the raw TCP packet, we extract the traffic features that are commonly used in WF research [24, 30]. The traffic dataset is comprised of the websites' traffic features when visited with different Tor browser versions and security-level configurations. Furthermore, we collect the requests and responses information of each visited website as our RCA dataset. The crawler visits each website and exports the requests and responses listed in the browser network panel by configuring the Tor browser and instFalling the har-export-trigger add-on. We collect the requests and responses information of each website when visiting the website with different Tor browser versions and security-level configurations. The RCA dataset also consists of the version RCA dataset and the setting RCA dataset. The version RCA

doi:10.6342/NTU202202954

dataset includes the request and response data of each website visited with four different Tor browser versions, and the setting RCA dataset contains the websites' request and response data when visited with different browser security levels.

To answer RQ1, we conduct the cross-setting examination to verify if network traffic differences which are originated from browser settings are distinctive enough to cause WF misclassification. Specifically, we examine the four latest Tor browser versions (i.e., 7, 8, 9, and 10) and three security levels (i.e., default, safer, and safest). We find that when the training and testing network traffic datasets are collected with different browser versions, the website classification accuracy drops more than 50% and that the website classification accuracy only obtains 54% in browser version 10. The results of security-level configurations show that the network traffic generated by default and safer settings is less distinguishable and that the website classification accuracy is lower if the security level is configured to the safer or safest options. The results imply that version difference will lead to distinguishable traffic features for most of the websites, but the traffic feature difference caused by security-level configurations are less distinguishable. Besides, the WF achieves low accuracy if the users use the latest browser version (i.e. browser version 10) with high security level (i.e, safer or safest), and it shows updating the browser version to the latest and setting the security level to safer or safest level can defend against WF.

To answer RQ2, we evaluate the BSF feasibility by creating browser-setting classifiers to identify the Tor browser version and security-level configuration given the collected network traffic. An interesting aspect of BSF is that users change their browser settings infrequently, and thus the browser-setting classifiers can make multiple predictions and combine the results to identify the browser settings. Under the closed-world assumption, the version classifier requires seven predictions to achieve 99% accuracy, and the

doi:10.6342/NTU202202954

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 of security configuration. The results show the BSF is possible when the attackers notice the websites users may possibly visit. The attackers can perform multiple classifications and ensemble the results to obtain accuracy that is high enough to apply the BSF in the real world. However, if the attackers have little knowledge about the user-visited websites, the BSF accuracy is too low to become a real-world threat, and more sophisticated machine learning algorithms or feature engineering approaches should be applied in this case.

To answer RQ3, we analyze the features from the browser-setting classifiers in RQ2, record the changelog and browser updates, and investigate the network log of the selected websites under different settings to explain our observations. We observe that cell features play a relatively more important role in BSF than the time features and flow features, and we conclude the browser feature changes in the Tor browser changelog to record possible updates that may influence the websites' network traffic. Furthermore, we visit the websites with different browser settings and analyze the generated requests and responses to perform the root cause analysis. The result shows that the safest-level security configuration will block most of the network requests, while the default and safer security levels do not have observable differences regarding the requests and responses. The browser version analysis shows no observable difference in the requests and responses when the websites are visited with different browser versions. The results also show that websites that generate fewer traffic data tend to obtain low accuracy in security setting classification but high accuracy in browser version classification. More investigation should be conducted to find out the root cause of browser settings' impact on network traffic.

Our findings suggest that BSF can improve existing WF and HSF techniques. More-

doi:10.6342/NTU202202954

over, since BSF is NTA-based, it represents an alternative to fingerprint browsers without JavaScript execution [31]. We also discuss the pitfalls we discover during the data collection. Several websites utilize the DDoS protection that redirects the user's website visit to a temporary web page to prevent the DDoS attack, and the protection will later redirect the website visit to the targeted web page after several seconds. This protection causes the crawler to collect data irrelevant to the targeted web page, thus influencing the WF and BSF accuracy. Future research should take the website's protection into account and remove those websites that adopt the DDoS protection or extend the network traffic collection timeout for those websites. Furthermore, we discuss the real-world impact of BSF by proposing a two-phase classifier that enhances the WF accuracy even if the user's browser settings are unknown. The two-phase classifier performs BSF in the first stage and performs the WF with the specific WF classifier based on the identified browser setting to improve the WF accuracy. Last, we discuss the potential countermeasure against BSF. Current WF defense techniques may be effective defenses against BSF because these techniques conceal the website's network traffic, and more research can be conducted to evaluate the defense effectiveness against BSF. Other defenses such as adopting DDoS protection may be an effective defense against BSF and WF because of the traffic generated by the temporary web page.

In this research, we examine the Tor browser version and user-configurable security level. Future research can apply similar approaches to analyze other browser settings or the combination of multiple browser settings. Also, more sophisticated learning algorithms and feature engineering can be conducted to improve BSF accuracy.

doi:10.6342/NTU202202954

1.1 Contributions

The main contributions of this research are:

- **BSF attack** We verify that the Tor browser settings influence the network traffic and propose an NTA-based browser setting fingerprinting attack called BSF.
- **An in-depth root cause analysis** We propose a systematic root cause analysis through changelog inspection and web request and response analysis. We discover the safest level of security configuration blocks the JavaScript and font requests, resulting in notable traffic feature differences. The analysis regarding browser versions suggests that the number of bytes sent and the number of POST requests has minor difference among browser versions.
- **Open-source dataset and code** Previously published datasets only comprise traffic from a single Tor browser setting. We publish our network traffic datasets (containing multiple Tor browser settings), root cause analysis dataset, traffic feature list, and code on GitHub[2] for reproducibility and future studies.

This thesis is an extended version of my Master's thesis at Waseda University in 2021 and the poster [13] published at the Web Conference 2022. Compared to Chang et al. [13], this thesis includes additional analysis and details that were omitted in the poster paper due to the page limit. The differences are highlighted below.

- **Background and Related work** The introduction of network traffic analysis, WF, WF assumption, traffic features, and WF defenses are added.

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

doi:10.6342/NTU202202954

- **Methodology** The environment setup, website list, crawler implementation details, and data cleaning process are provided in the thesis.
- **Evaluation** The closed-world setting classification results are added in figure **??**.
- **Discussion** We discuss the problems we found during our traffic dataset collection and the possible defenses against BSF.
- **Root Cause Analysis** Aside from providing the details of existing work, we conduct root cause analysis to investigate the impact of different browser settings on network traffic patterns. Section 3.1 and 3.6 and 5.4 present the experiment details and implications.

Furthermore, as I am in a dual degree program hosted by both Waseda University and National Taiwan University (NTU), this NTU thesis is a follow-up work of my master thesis at Waseda University in 2021. In the Waseda thesis, the investigation is limited to Tor browser versions 7, 8, and 9, while this research includes the examination of the latest TBB version (i.e., version 10, as of July 2022), security setting examination, and root cause analysis. The followings are the new content:

- **Methodology** The RCA website lists and security setting dataset are added.
- **Evaluation** The impact of user-configurable security level on network traffic is provided in the NTU thesis.
- **Analysis** We conduct the root cause analysis and discuss our findings on how the Tor browser setting affects the network requests and responses.
- **Discussion** The data collection pitfall we discover when collecting the version RCA dataset is provided in the NTU thesis.

doi:10.6342/NTU202202954

Chapter 2 Background and Related Work

Our work aims to identify a user's Tor browser setting using network traffic analysis in the Tor network and develop browser-setting classifiers leveraging website fingerprinting techniques. Therefore, this section introduces the Tor network and Tor browser, related research in network traffic analysis, and website fingerprinting.

2.1 Tor

Tor [18] is a low-latency anonymity network consisting of approximately 6,000 relays around the world [7]. Tor introduces several mechanisms, inclusive of traffic relay, data segmentation, and layers of packet encryption to prevent its users from Internet censorship. When the users surf the Internet through the Tor network, the packets will be forwarded to the locally installed Tor proxy. The Tor proxy selects three relays and builds a three-hop circuit for its users. The Tor proxy will forward the packets to the three relays sequentially and eventually reach the target server. Thus, the adversaries cannot identify the website users visit simply by inspecting the IP packet header. Besides, the packet is encrypted and segmented to a fixed size before sending the packet to the Tor circuit. The

doi:10.6342/NTU202202954

Tor proxy constructs the shared keys with the three Tor relays in the circuit respectively, encrypts the packet with the three shared keys according to the relays' order, and splits the network packet into 512-byte packets known as Tor cells. When the Tor relay receives the Tor cell, the relay decrypts the packet with the shared key and forwards the packet to the next relay. Eventually, the packet will be fully decrypted at the last Tor relay and will be sent to the target server. The data segmentation and layers of encryption guarantee data confidentiality during data transmission. To visit the websites through the Tor network, the Tor official provides an application known as the Tor Browser Bundle (TBB). A TBB comprises a Firefox browser with security enhancement add-ons (e.g., HTTPS Everywhere and NoScript) and a Tor proxy. Besides, the Tor browser supports three levels of security configuration that limit the browser's features to a certain level. Table 2.1 lists the security levels and corresponding changes. The security level is set to standard level by default, and all the browser features are supported. The safer level disables JavaScript on non-HTTPS websites, and performance optimization features include JIT compiler and web assembly. The safest level disables webGL, frame, object API, and JavaScript on all websites.

Table 2.1: 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, OpenType SVG, 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

doi:10.6342/NTU202202954

2.2 Network Traffic Analysis (NTA)

Network Traffic Analysis (NTA) is a side-channel attack that analyzes the network packets to extract sensitive information, inclusive of users' devices [41], applications [20], and users' behaviors [19]. As the packet data is usually encrypted during the transmission, recent traffic analysis focuses on investigating the packet metadata to obtain sensitive information or leveraging machine learning algorithms to extract complex features to increase classification accuracy [22, 39]. For instance, related research includes user's operating system identification through the TCP window size and TTL values in the TCP packet header [8], IoT identification through the packet transmission time statistics [22], and mobile app fingerprinting through performing semi-supervised learning on the network traffic patterns [21]. However, NTA is more challenging in the Tor network because of the privacy enhancements that conceal traffic features.

2.3 NTA Targeting Tor

Current NTA research in Tor mostly focuses on the WF [24] and the HSF [26, 33]. In both attacks, local and passive adversaries predict the websites or hidden services users visit through classifiers. Each web page and hidden service yield unique network traffic patterns, so the adversaries can fingerprint each web page and service based on the network traffic features. The HSF specifically aims at fingerprinting the Tor hidden service, while the WF aims at fingerprinting normal web pages. Few research extends the NTA in Tor to other perspectives such as behavior fingerprinting [16]. To the best of our knowledge, we are the first to examine the NTA browser setting fingerprinting in the Tor network.

doi:10.6342/NTU202202954

Aside from those difficulties in the WF, browser setting fingerprinting requires more effort and automation because a new traffic dataset is required for each examined browser setting. Specifically, to examine the impact of four tor browser versions and three security level configurations on network traffic, we collect four version traffic datasets and three security setting traffic datasets, and this requires lots of automation and data collection effort. Besides, verifying the browser settings' impact on the network traffic is not trivial because there is no standardized approach. We propose new approaches to investigate the browser settings' impact on network traffic with the cross-setting examination, browser setting fingerprinting through NTA, and the root cause analysis. For the browser setting fingerprinting through NTA, since there is no prior study conducting the experiments, we widely survey traffic features used in WF research, conclude those features, and apply them to browser setting fingerprinting. The root cause analysis also requires knowledge of the browser configuration, we study the browser configurations and add-ons to automatically export the browser network panel data.

2.3.1 Website Fingerprinting (WF)

In WF, adversaries train a WF classifier with machine learning algorithms and labeled training datasets. The labeled training dataset consists of the traffic features and their corresponding websites. Subsequently, the adversaries collect network packets between the targeted tor clients and the tor guard node, extract the traffic features, and perform classification to identify the web pages the Tor clients visit. Past research utilize k-nearest-neighbor [42], support-vector-machine [34], and random forest [24] to train WF classifier. Recent research uses state-of-the-art models including CNN [40], DNN [38], and ResNet [9]. However, due to the large number of web pages, the success of WF par-

doi:10.6342/NTU202202954

tially depends on the adversaries' knowledge of the website list, if the website list in the training dataset is considerably different from the websites user visited, the WF classifier will yield low accuracy.

2.3.2 Threat Model and Assumptions

We follow the threat model and assumptions commonly adopted by previous WF work. We consider a local passive adversary (e.g., a nation-state censor) that locates between the user (more precisely the client running the user's browser) and the guard nodes of the Tor network. That is, the adversary is able to eavesdrop on all traffic the user sends to and receives from the Tor network.Figure 2.1 illustrates the WF workflow. The adversary eavesdrops on the network traffic between the user and the Tor guard node, extracts the traffic features, and identifies the website user visit. The adversary has polynomial-bounded computing resources and cannot break cryptography in a reasonable time. We do not consider a global adversary that can correlate traffic from multiple vantage points because Tor cannot defend against such global adversaries by design.

Depending on the adversary's knowledge of the website list, prior research [35, 37] evaluates the WF classifier under two assumptions. Chang et al. [13] explained the two assumptions in their research: "the closed-world assumption assumes the adversary knows the exact set of websites the users may visit. In this case, the adversary 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 adversary knows only a subset of websites the users may visit. Thus, some websites visited by the users are excluded from the training dataset." The websites that do not appear in the attacker's website list will be

doi:10.6342/NTU202202954

classified as unmonitored websites, and only the websites that appear in the adversary's website list should be identified. In this research, we also evaluate our proposed BSF attack under these two assumptions.

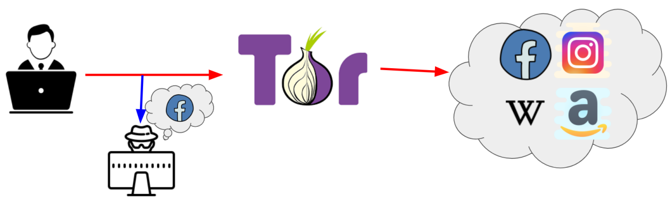


Figure 2.1: Website fingerprinting workflow.

2.3.3 Traffic Features

Prior research explored various traffic features that are distinctive among different websites. Li et al. [30] conduct a thorough evaluation of the traffic features effectiveness on WF with information theory, and Hayes and Daneniz [24] use the random forest to extract the top 20 important traffic features for WF. Common traffic features [30] can be categorized into three categories: packet flow statistics (e.g., number of packets), time-based packet features (e.g., packet flow duration), and traffic flow features.

2.3.4 Defense

To mitigate the WF and HSF, various defense approaches have been proposed to obfuscate the website's traffic features, including padding dummy packets [23, 28], traffic morphing [45], and delaying packets [10, 11]. However, these approaches come with significant bandwidth and latency overheads. As mentioned by the Tor Project (the official team maintaining Tor and TBB), Tor has not deployed the WF defense because WF and HSF attacks remain unrealistic due to several assumptions and limitations [32]. Furthermore, these defenses usually come with bandwidth overhead, latency, or implementation

doi:10.6342/NTU202202954

difficulties.

doi:10.6342/NTU202202954

doi:10.6342/NTU202202954

Chapter 3 Methodology

This section provides the details of the collected website lists, the crawler implementation, the data collection process, traffic feature extraction, and model training.

3.1 Website list

In this work, we would like to investigate the impact of Tor browser settings on network traffic features and, ultimately, the accuracy of the WF. To achieve this, we first compile two website lists, one for WF evaluation under the closed-world assumption and the other under the open-world assumption, as our crawling target.

We select the top 250 websites as the closed-world list and 7,000 websites as the open-world list from the Tranco website list [36]. The numbers were chosen to ensure our dataset sizes are comparable to those used in prior work after data cleansing. We surveyed prior work [12, 24, 34] and found that many of them collected the top 100 websites as the closed-world list to examine the closed-world assumption and 2,000 [30] to 9,000 [42] websites as the open-world list to examine the open-world assumption.

Additionally, following the previous approach [43], we remove Chinese websites and localized sites from the closed-world list. Chinese websites are likely to be blocked when

doi:10.6342/NTU202202954

visited with Tor [25], and localized sites tend to share similar traffic features. For instance, google.com, google.co.uk, and google.co.jp all appear in the top 200 Tranco website list, and the WF needs not to distinguish these sites, so we only keep one localized site if they appear multiple times in the website list. After manual removal, We have 196 websites left on the closed-world list. For the open-world website list, prior research does not perform website cleaning because the purpose of the open-world website list is to simulate real-world scenarios and the websites in the list are identified as unmonitored websites. No matter if the website visit succeeds or not, the WF should classify these websites as unmonitored sites. Thus, we collect 7,000 websites as our open-world website list but do not perform website cleaning.

Furthermore, to answer RQ3, we collect root cause analysis (RCA) website lists and analyze the requests and responses of these websites. Each RCA website list includes two sets of websites: The high accuracy website set contains websites whose traffic patterns are most distinguishable when visited with different browser settings, and the low accuracy website set is those whose traffic patterns are least distinguishable when visited with different browser settings.

- The version RCA website lists consist of the high accuracy website list and low accuracy website list. We construct the website list that consists of websites whose version classification accuracy is equal to 100% as the high accuracy website list and those less than 85% as the low accuracy website list. Because enableAuto-ExportToFile configuration is not introduced until Tor browser version 9 and the auto-har-export add-on cannot be installed in Tor browser version 8 and 7. We manually collect the dataset and only conduct small-scale examinations. The high accuracy website list contains 16 websites, and the low accuracy website contains

doi:10.6342/NTU202202954

12 websites.

- For the setting RCA website list, we collect those websites with security setting classification accuracy higher than 85% accuracy as high accuracy website list and those lower than 66% accuracy as low accuracy website list. Thus, the setting RCA website list consists of 49 websites in the high accuracy website list and 14 websites in the low accuracy website list.

Table 3.1 shows all the website lists used in this study.

Table 3.1: Website list

Website List	# of sites	Purpose
Closed-world list	196	Examine closed-world assumption
Open-world list	7,000	Examine open-world assumption
Version RCA	16	Websites of of high version classification accuracy
	12	Websites of low version classification accuracy
Setting RCA	49	Websites of of high security setting classification accuracy
	14	Websites of low security setting classification accuracy

3.2 Crawler Implementation

The crawler is implemented with an open-source library, tbselenium [46]. We follow prior work's crawling approach [43]. The crawler visits the 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 fixed Tor circuit, we rebuild the Tor circuit after visiting every website once in a round and restart the browser after every visit. We set up multiple crawlers, each with an examined browser setting, and crawl the website list in parallel. To isolate each crawling process, the crawler is executed in the docker container. Table 3.2 lists the corresponding Tor proxy along with Firefox version of the examined TBB version.

doi:10.6342/NTU202202954

Table 3.2: TBB and Firefox version mapping

TBB version	Tor proxy version	Firefox version
7.5.2	0.3.2	52.7.2ESR
8.0.6	0.3.5.7	60.5.1ESR
9.0.2	0.4.1.6	68.3.0ESR
10.5.2	0.4.5.9	78.12.0ESR

3.3 Data Collection

Prior work [42, 43] did not collect traffic datasets using multiple browser settings, so we collect our own datasets to compare traffic features across browser settings. To collect the datasets, we implement a crawler to visit each website and collect the network traffic with tcpdump. After collecting the multiple traffic instances for each website, we extract the packet header, discard the encrypted payload, and perform feature engineering to extract informative features such as the number of packets, incoming and outgoing packet ratios, inter-packet time interval as the final instance in our datasets, each instance is labeled based on the website it belongs to and the browser setting used to collect it. The datasets are collected under the closed-world and open-world assumptions, and for each assumption, we collect the version dataset and security setting dataset. The version dataset contains four TBB versions, and the security setting dataset includes three security levels. As we only examine one variable at a time, we only consider the standard security level in the version dataset and TBB version 10 in the security setting dataset. Thus, we end up having 14 datasets in total. Table 3.3 summarizes our datasets.[1]

Environment setup The data collection is conducted with AWS EC2 service. We created virtual machines on AWS EC2 with the same VM configuration (location: us-east-1b, t3.medium). Within each virtual machine, we isolate each crawling process with

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

doi:10.6342/NTU202202954

docker container and have multiple containers executed on each virtual machine. To collect the version dataset, the container installs the examined TBB version and runs the crawler. To collect the security setting dataset, the container installs TBB version 10, configures the Tor browser to the examined security level, and starts the crawling process. For each web page visit, the page load timeout is set to the default value of 300 seconds. The crawler waits 10 seconds after the website is fully loaded[2] before termination. The crawler collects the version and security setting datasets during October and November for two months in 2021.

Closed-world assumption The set of websites the user visits is known to the attackers. The crawler collects the traffic instances from the closed-world list 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.

Data Cleaning Due to the instability of Tor circuit, website visits occasionally fail due to the website's blocking policy or page load timeout. Previous research [43] removes the traffic instance of a website if its number of packets is fewer than 20% of the median size of that site. Therefore, we also consider traffic instances that contain fewer packets than 20% of the median size of that site as failure visits and remove those instances.

Open-world assumption The set of websites the user visits is partially known or unknown to the attackers. We consider the worst case and assume the website set is 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-

[2]As documented in selenium [3], the web page is considered fully loaded if it completes loading the DOM structure and sub-resources, including CSS, JavaScript, and images, before the page load timeout.

doi:10.6342/NTU202202954

world dataset consists of traffic instances from the open-world list. For each website, we collect one traffic instance with each examined browser setting. Because websites in the open-world website list are only visited once, we can hardly perform the statistic-based data cleaning approach as in the closed-world dataset. One possible way to perform the data cleaning is to compare the screenshot or the response HTML data to check if it is a successful visit or a failed visit.

Table 3.3: 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

3.4 Feature Selection

We conclude the informative traffic features including cell features, time features, and flow features in the past studies [24, 30] and add the wavelet transform features as our new features. We apply the wavelet transform on the time features as our new feature because the wavelet transform is used to extract the frequency information, and the frequency information is widely used as training features in image and signal processing. We perceive packet arrival time as signals and apply the wavelet transform to extract frequency information. While these traffic features are evaluated to be informative in WF but not in BSF, we evaluate the setting classification accuracy with these traffic features. The traffic features can be categorized into three categories. Table 3.4 summarizes the 117 features used in our study. The full feature list is provided on Github [3]

[3]https://github.com/csienslab/torbrowser-nta/blob/master/code/CellFeature.py and https://github.com/csienslab/torbrowser-nta/blob/master/code/TimeFeature.py

doi:10.6342/NTU202202954

Cell Features capture the packet statistics, including the total number of packets, incoming and outgoing packets ratio, and packet ordering. Cell features leak information about how the browser requests resources and how the server responds.

Time Features [37] represent the timing information of the traffic flow, inclusive of packet timestamp, the interval time between packets, and the wavelet transform of interval time.

Flow Features focus on the traffic flow statistics and traffic flow duration. A traffic flow is a group of consecutive outgoing or incoming packets.

Table 3.4: Feature Set Summary

Category	**Feature Type**	**Description**	**No.**
Cell feature	Packet Count	Number and ratio of incoming and outgoing packets	9
	Packet Order [30]	For each packet, we record total number of packets sent or received before it	4
	Packet Concentrate [24]	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
	Packet Burst [42]	Consecutive outgoing / incoming packets	6

3.5 Model Training

The most effective WF classifiers use deep learning techniques for feature extraction and model training [9, 38]. However, manually crafted traffic features are still critical for feature analysis and root cause analysis. Thus, we use random forest instead of state-of-

doi:10.6342/NTU202202954

the-art classification algorithms for classification, and we describe the training and testing data under the two assumptions. Under the closed-world assumption, we split the closed-world dataset into training and testing datasets. Under the open-world assumption, we train the classifiers using the closed-world dataset and validate their accuracy using the open-world dataset.

3.6 Root Cause Analysis (RCA) dataset

So far, we have collected data at the network layer for examining our BSF attack. Now, to better analyze the root cause of the success/failure of the BSF attack, we also collect application-layer information into the RCA datasets. Specifically, the RCA datasets consist of the requests and responses of the websites in version and setting RCA website list, allowing us to identify the content type (e.g., JavaScript, HTML, image) by simply checking the HTTP headers. We collect this information by exporting the logs from the browser's network panel.

Setting RCA dataset Two modifications are made in the crawler to collect the setting RCA dataset. We install the har-export-trigger add-on [17] and set the `enableAutoExportToFile` browser configuration to true to automatically export the HTTP requests and response information listed in the browser network panel to the HTTP Archive (HAR) format after each web page load. The crawler visits the website in the setting RCA website lists using the identical approach introduced in Section 3.2. We automatically collect the requests and responses of the websites for a week and end up with around 70 requests and responses instances for each website.

Version RCA dataset Because some examined TBB version does not support the

doi:10.6342/NTU202202954

har-export-trigger add-on and the auto-export configuration, we manually visit each website and export the HTTP requests and responses shown in the browser network panel into HAR-formated files. The version RCA website list only contains a small set of websites, and there are only approximately ten request and response instances for each website in the version RCA dataset.

doi:10.6342/NTU202202954

doi:10.6342/NTU202202954

Chapter 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).

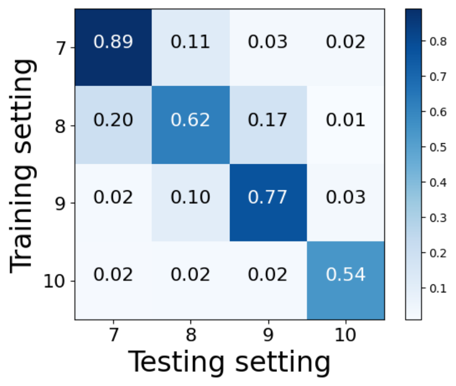
4.1 Cross-setting Examination (RQ1)

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 and closed-world dataset. We train a WF classifier with one browser setting dataset and test its accuracy on another dataset. Because the cross-setting examination aims to investigate the decrease in classification accuracy when training and testing traffic instances are collected with different browser settings, the closed-world dataset is enough to perform the examination, and there is no need to evaluate under the open-world assumption.

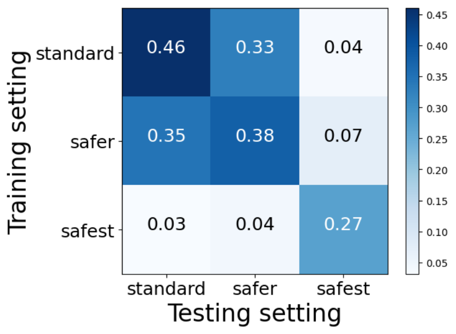
Figure 4.1 summarizes the results. The WF attack accuracy drops significantly when the training and testing datasets are collected with inconsistent settings. The browser version drops from 40% to 87%, and the security setting drops from 3% to 42%. The results confirm that browser settings influence traffic features. Moreover, we discover that web-

doi:10.6342/NTU202202954

sites' traffic features barely change between the standard and safer security settings. In Figure 4.1b, even though the training and testing datasets are collected with inconsistent security settings, the WF models achieve above 33% accuracy in the standard and safer settings. We conduct a more detailed analysis and discuss the reasons in Section 5.4.



(a) TBB version



(b) Security setting

Figure 4.1: cross-setting examination
The training setting indicates the WF classifier is trained with the dataset collected with the specific browser setting, and the testing setting implies the browser setting used to collect the testing dataset. Each cell shows the experiment is conducted with the specific training setting and testing setting, and the number is the testing accuracy. The testing accuracy is the rate that the WF classifier successfully identifies the website given the traffic dataset collected with the testing browser setting.

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 data and train the classifiers using the training data. The version classifier achieves 89% accuracy on the testing data, and the security setting classifier achieves 77% accuracy on the testing data. The accuracy rate is sufficiently high in practice because a user will change

doi:10.6342/NTU202202954

browser settings infrequently, and the attackers can assume that consecutive n traffic instances have identical browser settings. The attackers 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 provided in Equation 4.1.

- n: The number of predictions
- P: Probability of getting correct result after n predictions
- p: the probability of getting an incorrect result

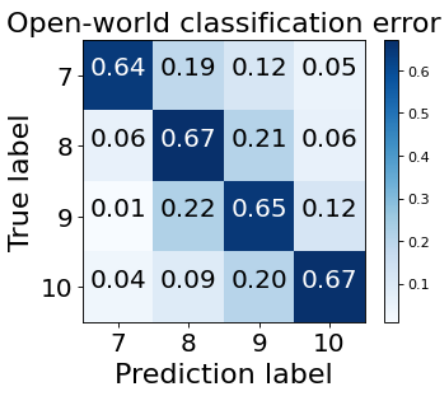
$$P = 1 - \sum_{i=0}^{i=\frac{n}{2}} C_i^n * (1-p)^i * p^{n-i} \tag{4.1}$$

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 99% accuracy in seven predictions.

Open-world Result We train the version classifier and security setting classifier with the closed-world dataset and test their accuracy with the open-world dataset. The version classifier reaches 65% accuracy. The attacker can repeatedly apply the version classifier and reach 99% accuracy after 59 predictions. However, the accuracy for the security setting classifier only reaches 60%, which means the security setting classifier cannot reach 99% accuracy within 100 predictions. Note that this work aims to evaluate the feasibility of browser setting fingerprinting through NTA and explain the observations. We do

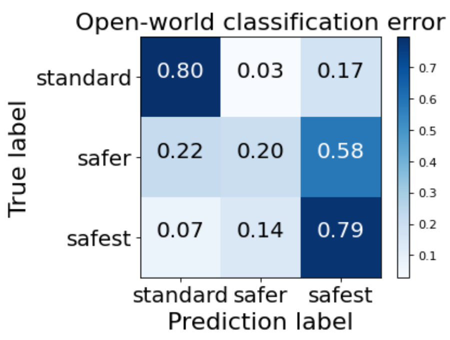
doi:10.6342/NTU202202954

not invest in feature engineering or ML model optimization and only perform classification with random forest. We expect the accuracy to be improved further via sophisticated ML models and extra data-processing effort. Moreover, through the confusion matrix in figure 4.2, we find that prediction errors in the safer level account for most of the classification errors. Though the current security setting classifier cannot accurately distinguish the three examined 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 disabling of the JIT compiler and will be discussed in Section 5.3.



True label \ Prediction label	7	8	9	10
7	0.64	0.19	0.12	0.05
8	0.06	0.67	0.21	0.06
9	0.01	0.22	0.65	0.12
10	0.04	0.09	0.20	0.67

(a) version classification



True label \ Prediction label	standard	safer	safest
standard	0.80	0.03	0.17
safer	0.22	0.20	0.58
safest	0.07	0.14	0.79

(b) setting classification

Figure 4.2: Open-world setting classification results

doi:10.6342/NTU202202954

Chapter 5 Analysis (RQ3)

To understand why Tor browser settings affect traffic features, we conduct comprehensive and in-depth investigations through feature analysis, changelog inspection, and RCA dataset analysis. This section discusses our findings and answers RQ3.

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 the scikit-learn library [2]. The feature importance is calculated by normalizing each feature's Gini impurity. A feature's Gini impurity is a measurement of how effectively the feature split the dataset to form the decision tree. More precisely, Gini impurity indicates the likelihood of data being misclassified after splitting the data based on the examined features. The lower the Gini impurity of a feature, the less likely the new data will be misclassified after splitting the data according to the feature value. Equation 5.1 shows the Gini impurity formula. Figure 5.1 lists the most informative features whose normalized Gini impurity is higher than 0.01.

- L: Browser setting categories.

doi:10.6342/NTU202202954

- $P(i)$: The probability of selecting the instance of browser setting i after the data splitting.

- $p(i)$: The probability of selecting the instance of browser setting i before the data splitting.

$$Gini = \sum_{i=0}^{i=L} P(i)^2 - \sum_{i=0}^{i=L} p(i)^2 \tag{5.1}$$

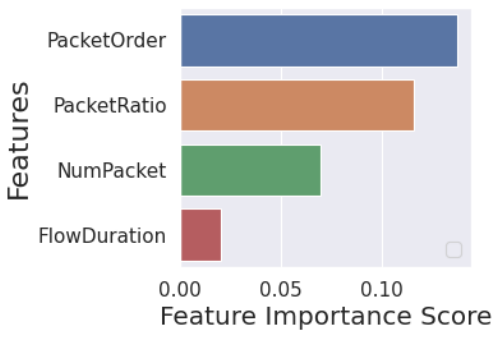
The results indicate that traffic features concerning the alteration in packet orders or quantity traffic features have apparent differences among browser settings, while the time features (e.g, total time, packet timestamp, or packet interval time) and flow features (e.g, number of flows, flow duration, packet bursts) have comparatively less influence in the browser setting identification. One possible explanation is that the packet arrival time and flow duration are primarily dependent on Tor's network condition.

For the security setting classification, traffic features tend to be less informative. The most informative feature, the number of packets (`NumPacket`), accounts for 0.03 of feature importance. We suppose that the total time (`TotalTime`) feature is relatively more informative in the security setting classification because the JIT compiler is disabled in safer and safest security levels, which increases page load time.

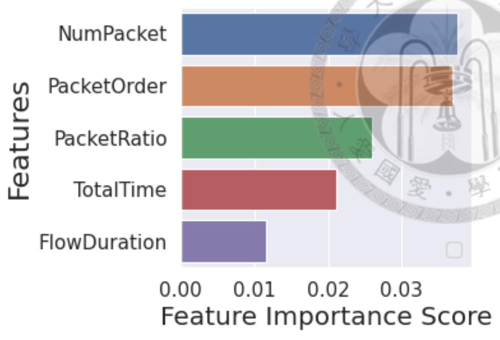
5.2 Changelog Inspection

To uncover the root cause, we manually scrutinize the changelog of Tor proxy and Firefox and identify updates that may affect network traffic. The changes can mainly be categorized into two types, security policy updates and browser optimization. Table 5.1

doi:10.6342/NTU202202954



(a) version classifier



(b) Security setting classifier

Figure 5.1: Classifier's feature importance

shows the summary of changelog inspection.

Table 5.1: 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 adopted TLS1.3 by default since version 61 and has introduced several tracking protection configurations. Changes in the content-blocking policy may 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) in

doi:10.6342/NTU202202954

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

Since the pref.js file records real-time browser setting changes [1], we record the pref.js differences using the diff command while changing the security levels and study the Firefox browser source code to compare the browser setting changes among the three security levels. Table 2.1 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 under the safer security level. We suppose this change attributes to fewer traffic data and inaccuracy in the safer security level.

5.4 RCA Dataset Analysis

We investigate the requests and responses in the RCA dataset and discuss our observations in this section. We evaluate the websites' network traffic when different browser settings are used to visit the websites. We investigate the request content type and number of bytes sent and received when visiting the websites with different browser settings.

doi:10.6342/NTU202202954

5.4.1 Content-type Analysis

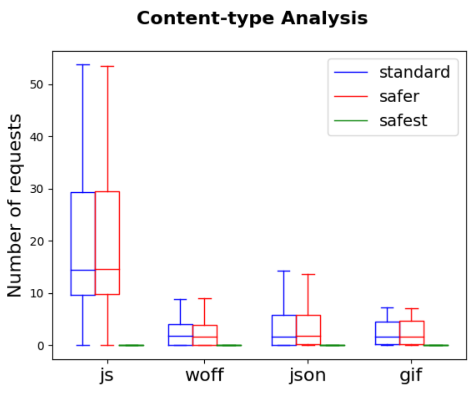
We categorize the requests based on their request content type and count the average number of each content type when the crawler visits the website with different browser security settings and browser versions. The results are shown in Figure 5.2 and Figure 5.3. Figure 5.2 shows the content type analysis result of security settings, and Figure 5.3 shows the result of the browser version. There are various request content types in the analysis, and we only visualize four content types: javascript request, WOFF font request, JSON request, and gif image requests, because these four content types yield the most significant difference among browser settings and because of the space limit.

The traffic features of the safest security setting are considerably distinguishable The safest security setting blocks all JavaScript, ASPX, and font-related requests. Thus, the number of requests decreases significantly, making the safest security setting easily identified. Figure 5.2 shows the websites' request content type statistics under standard, safer, and safest security settings. The standard and safer levels have a similar distribution in terms of the four examined content types, while the safest level has virtually no JavaScript, font, JSON, and gif image requests for all the websites. The result is consistent with the security setting inspection in Section 5.3.

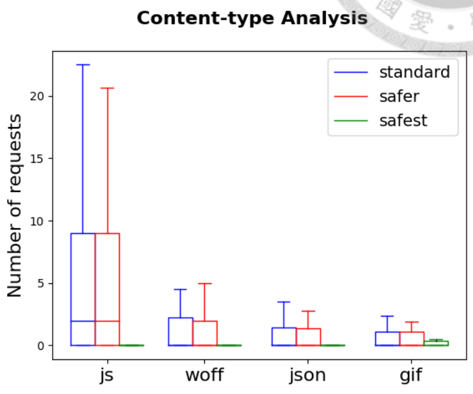
Standard and safer security setting does not have notable differences in terms of the request content-type In figure 5.2, we observe that the numbers of each request content-type only have subtle differences or even no difference between the safer and standard security settings, and this is identical to the changelog inspection in Section 5.2. The result indicates that the browser setting classifiers may not identify the standard and safer security settings due to little difference in the network traffic and that the WF classifiers

doi:10.6342/NTU202202954

that are trained with the standard security setting may also work on the network traffic collected with the safer security setting.

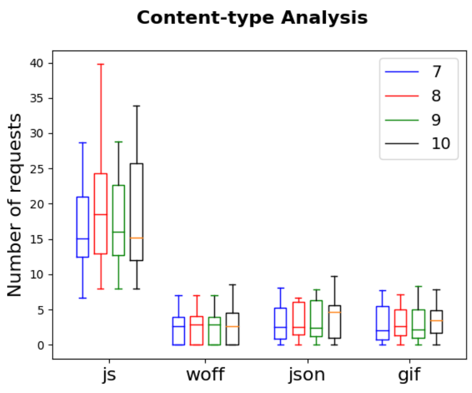


(a) High accuracy website list

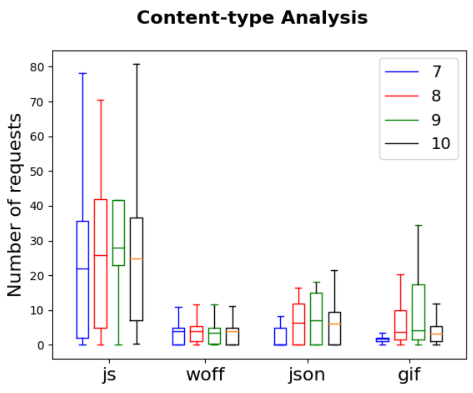


(b) Low accuracy website list

Figure 5.2: Boxplot of websites' request content type under different browser security settings



(a) High accuracy website list



(b) Low accuracy website list

Figure 5.3: The boxplot of websites' request content type under different browser versions

Request content-type does not exist observable difference among browser versions The average numbers of examined request content types only exist slight differences when visiting the websites with different tor browser versions. The result indicates that there is not a single examined factor that can successfully distinguish the TBB version and that combining multiple factors may be an approach to yield notable traffic feature differences among the TBB version.

doi:10.6342/NTU202202954

5.4.2 Request and Response Byte Analysis

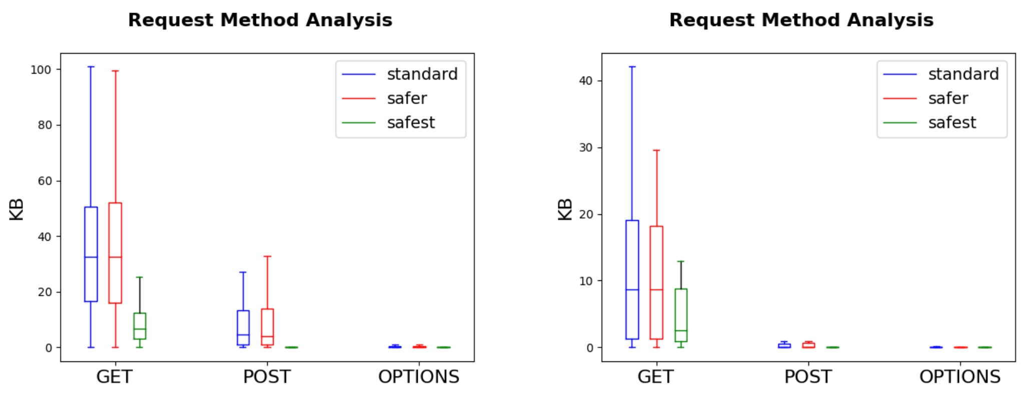
We categorize the requests and responses based on the request method(i.e. GET, POST, PUTS, OPTIONS requests) and investigate the request bytes and response bytes of each request method when visiting the RCA website list under different browser settings. For the security setting analysis, we visit the setting RCA website list with different browser security settings. For the browser version analysis, we visit the version RCA website list with different browser versions. Figure 5.4 and figure 5.5 show the request bytes and response bytes of the security setting analysis, and figure 5.6 and figure 5.7 show the request bytes and response bytes of the browser version analysis. We discover four request methods: GET, POST, OPTIONS, and PUTS methods in the investigation. The PUTS method requests only appear on fewer than five websites, so we do not visualize the result as the average request bytes and response bytes for the PUTS method are close to zero. In the four figures(i.e. figure 5.4, 5.5, 5.6, and 5.7), the x-axis represents the request method, and the y axis represents the kilobytes of the request bytes or response bytes.

The safer security setting has little impact on network traffic We study the number of requests and responses made for each website from the setting RCA dataset, and we observe that there is only a minor difference in the network requests and responses when the websites are visited with standard and safer security settings. This result accords with the cross-setting examination in Section 4.1. Therefore, more sophisticated learning algorithms are required to distinguish traffic generated by standard and safer security settings.

Low accuracy websites tend to have fewer requests and responses in security settings analysis but have more requests and responses in browser versions analysis In

doi:10.6342/NTU202202954

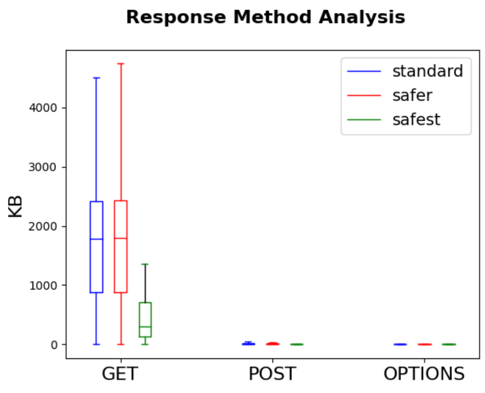
the security setting analysis, we discover that the low accuracy websites tend to have fewer request bytes and response bytes than the high accuracy websites. The reason is that higher levels of security settings block browser and website features that are potentially vulnerable. Websites generating large amounts of traffic data will perform differently when visited with different security settings, and websites with simple features may behave identical when visited with different security settings. However, in the browser version analysis, we observe opposite results, the low accuracy websites tend to have more request bytes and response bytes than high accuracy websites. The reason behind the browser version analysis observation is unknown and may require more investigation. This indicates that in BSF, websites generating more traffic data when visited tend to be more indistinguishable in the security setting identification but will be more distinguishable in the browser version identification and that BSF may not successfully identify both the browser version and security setting if all the visited websites have similar features.

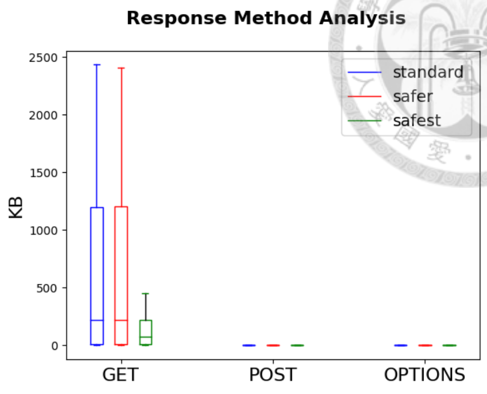


(a) High accuracy website list (b) Low accuracy website list

Figure 5.4: Boxplot of average websites' request bytes under different browser security settings

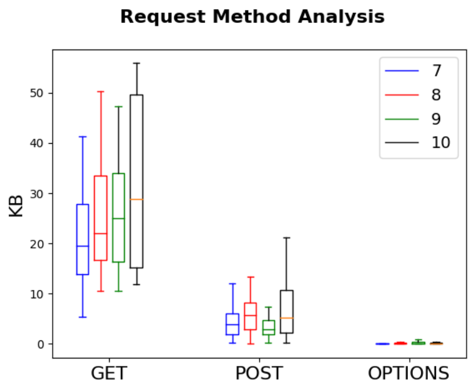
doi:10.6342/NTU202202954

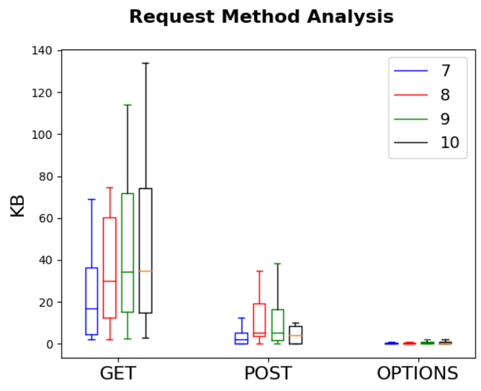




(a) High accuracy website list (b) Low accuracy website list

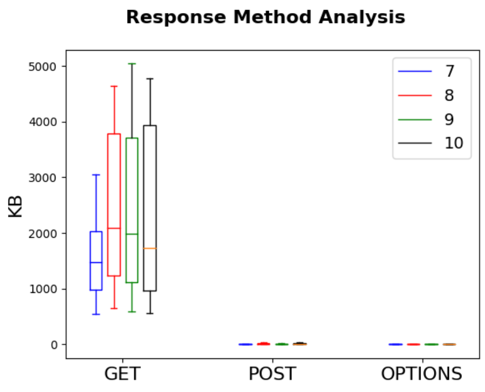
Figure 5.5: Boxplot of average websites' response bytes under different browser security settings

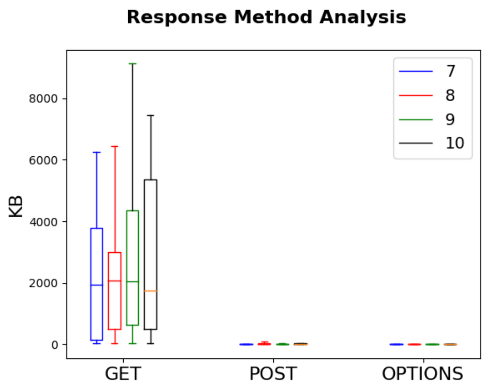




(a) High accuracy website list (b) Low accuracy website list

Figure 5.6: Boxplot of average websites' request bytes under different browser versions





(a) High accuracy website list (b) Low accuracy website list

Figure 5.7: Boxplot of average websites' response bytes under different browser versions

doi:10.6342/NTU202202954

doi:10.6342/NTU202202954

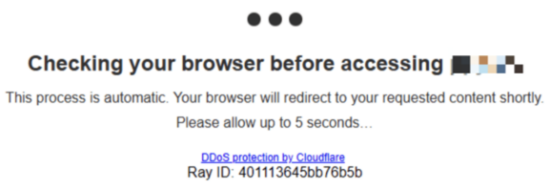
Chapter 6 Discussion

In this section, we discuss the pitfalls lots of research may encounter when collecting traffic data. Also, we discuss the countermeasures against BSF, the limitations, and the possible future directions of this research.

6.1 Data Collection Pitfalls in the Tor Network

Most WF research implements crawlers to visit websites and collect network traffic automatically. However, while manually collecting the request and response datasets for the version root cause analysis, we find that several websites' behaviors may influence the network traffic and that these behaviors have not been considered previously. For instance, Spotify may block requests from certain IP addresses, and several websites occasionally redirect traffic to Cloudflare for DDoS protection before handling those requests. Figure 6.1 shows a Cloudflare DDoS protection example. These behaviors are not considered during the automatic data collection process and may affect the traffic features along with WF attack effectiveness. Based on this observation, we recommend that, for future NTA research, the traffic collection process should not only collect the network packets but also verify the requests and responses to ensure the website is successfully visited.

doi:10.6342/NTU202202954


Checking your browser before accessing [redacted]

This process is automatic. Your browser will redirect to your requested content shortly.

Please allow up to 5 seconds...

DDoS protection by Cloudflare

Ray ID: 401113645bb76b5b

Figure 6.1: Example of Cloudflare DDoS protection.

6.2 Two-phase Classifier

Most WF research assumes the attackers collect the traffic datasets with browser settings identical to the users. However, former research [27] and our work prove that browser settings have remarkable impacts on WF accuracy. Future research should take the browser settings into consideration, and more work should be conducted to evaluate the impact of browser settings on the WF. A possible scheme is the two-phase classifier, the two-phase classifier consists of a browser setting classifier and the WF classifiers. The browser setting classifier is similar to the version and the security setting classifier in our second research question but with a wider range of browser setting fingerprinting. For each browser setting combination, we collect the traffic dataset with each browser setting combination and train a WF classifier with the dataset. Thus, the two-phase classifier performs the browser setting classification in the first stage and selects the WF classifiers based on the identified browser settings to perform the WF. A possible future direction is to evaluate the effectiveness of the two-phase classifier and to compare the former WF classifiers with the two-phase classifier under the condition that users' browser settings are unknown. However, this approach suffers from the limitation of browser setting fingerprinting: the exponential growth of traffic datasets, and the discussion is provided in Section 6.4.

doi:10.6342/NTU202202954

6.3 Potential Countermeasures

The BSF can increase the WF and the HSF accuracy. Besides, the BSF can threaten users' anonymity if a custom setting is used. Existing WF defenses can be applied to reduce the browser setting fingerprinting accuracy. However, these defenses cannot be easily deployed since they have to be implemented in the Tor browsers or relays. Furthermore, the bandwidth overhead and latency can degrade the browser performance. More research should be conducted to evaluate the effectiveness of the WF defenses against the BSF. Based on our observations and analysis, we discuss several alternatives. From the web developers' perspective, adopting common cloud protection can conceal the website's traffic pattern. For instance, ResearchGate adopts Cloudflare under-attack-mode protection [15]. This protection inserts an interstitial page that shows a five seconds challenge to block potential DDoS attacks. The similar traffic generated by the interstitial page can obfuscate the website's traffic.

6.4 Limitations

Because the evaluation and the data collection of BSF research are based on the WF research, this research shares a similar limitation as criticized in WF [27, 34]: the synthetic dataset. Besides, the proposed approach for browser-setting analysis only allows evaluating a small number of browser settings at a time.

Single browser setting examination The proposed scheme can only examine a small number of browser settings at a time because a traffic dataset is required for a new combination of browser settings. The numbers of browser setting combinations grow exponen-

doi:10.6342/NTU202202954

tially when multiple browser settings are considered, and large numbers of setting combinations require an enormous traffic dataset. The current scheme will face difficulties in traffic data collection and analyzing large datasets. However, in the real world, users usually change multiple browser settings at a time, and the websites' traffic patterns will be influenced by multiple browser settings.

Synthetic data Like most WF research, the traffic dataset is collected automatically using crawlers to evaluate the BSF accuracy instead of using real user traffic data. Since the users' browsing habits and the crawler's automatic process are unalike, the collected traffic dataset is synthetic and cannot represent real-world traffic. For instance, users tend to open multiple browser tabs when surfing the Internet, and the traffic will be mixed with lots of noises generated by other browser tabs. Also, most WF research uses Alexa or Tranco website lists as their closed-world dataset for evaluation, but the website lists may not represent the real ranking in the Tor network as users more often visit censored websites that may not appear in the popular website list. Last, the traffic collection is mainly limited to the landing web page of each website, but the users may visit the internal web page in the domain instead of the landing web page. Several works discuss and propose approaches to solving these limitations [14, 34, 44]. However, these studies conclude that these difficulties limit the WF threat in the real world. These difficulties also limit the BSF's practicability in the real world.

6.5 Future Work

We propose a systematic approach to examine the browser version and security setting's impact on the network traffic. Future research can apply further analysis based on

doi:10.6342/NTU202202954

our approach. The followings are the possible future directions:

- **Browser setting evaluation** One future direction is to apply a similar approach to evaluate other browser settings, such as the language settings and popular add-ons.
- **Deep learning models** Applying state-of-the-art deep learning models to the BSF may yield better classification results and even generate complex traffic features that can distinguish the standard and safer security levels.
- **Real-world impact** Prior WF and HSF research assume the training and testing traffic dataset is collected with identical browser settings. We identify that the browser settings influence the network traffic and discuss the feasibility of the two-phase classifier. Follow-up work can conduct more extensive studies on the effectiveness of the two-phase classifier.
- **Defenses** We discuss the possible defenses against BSF, including applying WF defenses or DDoS protection. More experiments can be conducted to verify the effectiveness of these defenses.

doi:10.6342/NTU202202954

doi:10.6342/NTU202202954

Chapter 7 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 fingerprinting is feasible when the attackers can continuously eavesdrop on traffic. The existing browser setting fingerprinting approach known as the browser fingerprinting [31] requires the users to visit attacker-controlled URLs. Our approach allows the attackers to identify the browser 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 fingerprinting, an interesting future direction is to examine other browser settings or even the combination of multiple browser settings. Current approach examines one setting at a time, and it will be more practical to consider the effect of multiple browser settings on network traffic instead of a single setting. Another direction is to improve the browser 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 classification accuracy. Because we prioritize feature analysis over the accuracy, we do not utilize any deep learning model for classification.

doi:10.6342/NTU202202954

doi:10.6342/NTU202202954

References

[1] Configuration editor for firefox. https://support.mozilla.org/en-US/kb/about-config-editor-firefox.

[2] scikit-learn. https://scikit-learn.org/stable/.

[3] selenium page load documentation. https://www.selenium.dev/documentation/en/webdriver/page_loading_strategy/.

[4] Attacks on tor. https://github.com/Attacks-on-Tor/Attacks-on-Tor, February 2020.

[5] How to shut firefox up about updates. https://support.mozilla.org/en-US/questions/1289766, June 2020.

[6] Tor metrics. https://metrics.torproject.org/userstats-relay-country.html, June 2020.

[7] Tor mtrics. https://metrics.torproject.org/networksize.html, June 2022.

[8] A. Aksoy and M. H. Gunes. Operating system classification performance of tcp/ip protocol headers. In 2016 IEEE 41st Conference on Local Computer Networks Workshops (LCN Workshops), pages 112–120, 2016.

doi:10.6342/NTU202202954

[9] S. Bhat, D. Lu, A. Kwon, and S. Devadas. Var-cnn: A data-efficient website fingerprinting attack based on deep learning. Proceedings on Privacy Enhancing Technologies, 2019:292–310, 10 2019.

[10] X. Cai, R. Nithyanand, and R. Johnson. Cs-buflo: A congestion sensitive website fingerprinting defense. In Proceedings of the 13th Workshop on Privacy in the Electronic Society, WPES '14, page 121–130, New York, NY, USA, 2014. Association for Computing Machinery.

[11] X. Cai, R. Nithyanand, T. Wang, R. Johnson, and I. Goldberg. A systematic approach to developing and evaluating website fingerprinting defenses. In Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, CCS '14, page 227–238, New York, NY, USA, 2014. Association for Computing Machinery.

[12] X. Cai, X. C. Zhang, B. Joshi, and R. Johnson. Touching from a distance: Website fingerprinting attacks and defenses. In Proceedings of the 2012 ACM Conference on Computer and Communications Security, CCS '12, page 605–616, New York, NY, USA, 2012. Association for Computing Machinery.

[13] C.-M. Chang, H.-C. Hsiao, T. Lynar, and T. Mori. Know your victim: Tor browser setting identification via network traffic analysis. In Companion Proceedings of the Web Conference 2022, WWW '22, page 201 – 204, New York, NY, USA, 2022. Association for Computing Machinery.

[14] G. Cherubin, R. Jansen, and C. Troncoso. Online website fingerprinting: Evaluating website fingerprinting attacks on tor in the real world.

[15] Cloudflare. Understanding Cloudflare Under Attack mode.

doi:10.6342/NTU202202954

https://support.cloudflare.com/hc/en-us/articles/
200170076-Understanding-Cloudflare-Under-Attack-mode-advanced-DDOS-protection-
July 2022.

[16] A. Cuzzocrea, F. Martinelli, F. Mercaldo, and G. Vercelli. Tor traffic analysis and detection via machine learning techniques. In 2017 IEEE International Conference on Big Data (Big Data), pages 4474–4480, 2017.

[17] F. DevTools. har-export-trigger. https://github.com/firefox-devtools/har-export-trigger, May 2022.

[18] R. Dingledine, N. Mathewson, and P. F. Syverson. Tor: The second-generation onion router. In USENIX Security Symposium, 2004.

[19] G. Draper-Gil, A. H. Lashkari, M. S. I. Mamun, and A. A. Ghorbani. Characterization of encrypted and vpn traffic using time-related. In Proceedings of the 2nd international conference on information systems security and privacy (ICISSP), pages 407–414, 2016.

[20] T. Ede, R. Bortolameotti, A. Continella, J. Ren, D. Dubois, M. Lindorfer, D. Choffnes, M. Steen, and A. Peter. Flowprint: Semi-supervised mobile-app fingerprinting on encrypted network traffic. 01 2020.

[21] T. Ede, R. Bortolameotti, A. Continella, J. Ren, D. Dubois, M. Lindorfer, D. Choffnes, M. Steen, and A. Peter. Flowprint: Semi-supervised mobile-app fingerprinting on encrypted network traffic. 01 2020.

[22] N. M. R. S. T. Engel. IoT Device Fingerprinting: Machine Learning based Encrypted Traffic Analysis. In IEEE Wireless Communications and Networking Conference, 2019.

doi:10.6342/NTU202202954

[23] J. Gong and T. Wang. Zero-Delay Lightweight Defenses against Website Fingerprinting. USENIX Association, USA, 2020.

[24] J. Hayes and G. Danezis. k-fingerprinting: A robust scalable website fingerprinting technique. In 25th USENIX Security Symposium (USENIX Security 16), pages 1187–1203, Austin, TX, Aug. 2016. USENIX Association.

[25] N. P. Hoang, A. A. Niaki, J. Dalek, J. Knockel, P. Lin, B. Marczak, M. Crete-Nishihata, P. Gill, and M. Polychronakis. How great is the great firewall? measuring china's DNS censorship. In 30th USENIX Security Symposium (USENIX Security 21), pages 3381–3398. USENIX Association, Aug. 2021.

[26] R. Jansen, M. Juarez, R. Galvez, T. Elahi, and C. Diaz. Inside job: Applying traffic analysis to measure tor from within. 01 2018.

[27] M. Juarez, S. Afroz, G. Acar, C. Diaz, and R. Greenstadt. A critical evaluation of website fingerprinting attacks. In Proceedings of the ACM SIGSAC Conference on Computer and Communications Security, 2014.

[28] M. Juarez, M. Imani, M. Perry, C. Diaz, and M. Wright. Wtf-pad: Toward an efficient website fingerprinting defense for tor. 12 2015.

[29] A. Kwon, M. AlSabah, D. Lazar, M. Dacier, and S. Devadas. Circuit fingerprinting attacks: Passive deanonymization of tor hidden services. In Proceedings of the 24th USENIX Conference on Security Symposium, SEC'15, page 287–302, USA, 2015. USENIX Association.

[30] S. Li, H. Guo, and N. Hopper. Measuring information leakage in website fingerprinting attacks and defenses. In Proceedings of the 2018 ACM SIGSAC Conference on

doi:10.6342/NTU202202954

Computer and Communications Security, CCS '18, page 1977 – 1992, New York, NY, USA, 2018. Association for Computing Machinery.

[31] D. G. Michael Schwarz, Florian Lackner. JavaScript Template Attacks: Automatically Inferring Host Information for Targeted Exploits. In Proceedings of Network and Distributed System Security Symposium, 2019.

[32] mikeperry. A critique of website traffic fingerprinting attacks. https://blog.torproject.org/critique-website-traffic-fingerprinting-attacks/, November 2013.

[33] R. Overdorf, M. Juarez, G. Acar, R. Greenstadt, and C. Diaz. How unique is your .onion? an analysis of the fingerprintability of tor onion services. 08 2017.

[34] A. Panchenko, F. Lanze, A. Zinnen, M. Henze, J. Pennekamp, K. Wehrle, and T. Engel. Website fingerprinting at internet scale. 02 2016.

[35] A. Panchenko, L. Niessen, A. Zinnen, and T. Engel. Website fingerprinting in onion routing based anonymization networks. In Proceedings of the 10th Annual ACM Workshop on Privacy in the Electronic Society, WPES '11, page 103 – 114, New York, NY, USA, 2011. Association for Computing Machinery.

[36] V. L. Pochat, T. V. Goethem, S. Tajalizadehkhoob, M. Korczynski, and W. Joosen. Tranco: A research-oriented top sites ranking hardened against manipulation. In Proceedings 2019 Network and Distributed System Security Symposium. Internet Society, 2019.

[37] M. S. Rahman, P. Sirinam, N. Mathews, K. Gangadhara, and M. Wright. Tik-tok: The utility of packet timing in website fingerprinting attacks. Proceedings on Privacy Enhancing Technologies, 2020:5–24, 07 2020.

doi:10.6342/NTU202202954

[38] V. Rimmer, D. Preuveneers, M. Juarez, T. Van Goethem, and W. Joosen. Automated website fingerprinting through deep learning. 02 2018.

[39] S. D. Siby, M. Juárez, C. Díaz, N. Vallina-Rodriguez, and C. Troncoso. Encrypted dns -> privacy? a traffic analysis perspective. ArXiv, abs/1906.09682, 2020.

[40] P. Sirinam, M. Imani, M. Juarez, and M. Wright. Deep fingerprinting: Undermining website fingerprinting defenses with deep learning. In Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, CCS '18, page 1928–1943, New York, NY, USA, 2018. Association for Computing Machinery.

[41] A. Sivanathan, D. Sherratt, H. H. Gharakheili, A. Radford, C. Wijenayake, A. Vishwanath, and V. Sivaraman. Characterizing and classifying iot traffic in smart cities and campuses. In 2017 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), pages 559–564, 2017.

[42] T. Wang, X. Cai, R. Nithyanand, R. Johnson, and I. Goldberg. Effective attacks and provable defenses for website fingerprinting. In Proceedings of the 23rd USENIX Conference on Security Symposium, SEC'14, page 143–157, USA, 2014. USENIX Association.

[43] T. Wang and I. Goldberg. Improved website fingerprinting on tor. In Proceedings of the 12th ACM Workshop on Workshop on Privacy in the Electronic Society, WPES '13, page 201–212, New York, NY, USA, 2013. Association for Computing Machinery.

[44] T. Wang and I. Goldberg. On realistically attacking tor with website fingerprinting. Proceedings on Privacy Enhancing Technologies, 2016(4):21–36, 2016.

doi:10.6342/NTU202202954

[45] T. Wang and I. Goldberg. Walkie-talkie: An efficient defense against passive website fingerprinting attacks. In Proceedings of the 26th USENIX Conference on Security Symposium, SEC'17, page 1375–1390, USA, 2017. USENIX Association.

[46] webfp. tor-browser-selenium. https://github.com/webfp/tor-browser-selenium, June 2020.

doi:10.6342/NTU202202954