Side Attack: Off-Path Website Fingerprinting via Cross-Circuit Interference in Tor

Mohamed Farrag Gad
mohamed.farrag.gad@hof-university.de
Hof University of Applied Sciences
Hof, Germany

Sebastian Pahl
spahl-research@online.de
Independent Researcher
Hof, Germany

Stefan Gast
stefan.gast@tugraz.at
Graz University of Technology
Graz, Austria

Daniel Gruss
daniel.gruss@tugraz.at
Graz University of Technology
Graz, Austria

Florian Adamsky
florian.adamsky@hof-university.de
Hof University of Applied Sciences
Hof, Germany

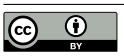
Abstract

Tor is a privacy-enhancing technology that enables anonymous communication by leveraging onion routing. A client chooses a path of three Tor relays, known as a *circuit*, and encrypts the traffic for each relay in reverse order, sending it to the first relay. Each relay knows only its predecessor and successor, unaware of the entire circuit, thus enabling anonymous communication. Tor builds the circuit telescopically, i. e., the first relay establishes a TCP connection to the second, and the second establishes a TCP connection to the third. However, Tor allows cross-circuit interference (CCI), meaning that if two circuits share two consecutive relays, data transmitted over the same TCP connection interferes with each other.

In this paper, we demonstrate that CCI can be exploited as a side channel to conduct an off-path (non-MitM) website fingerprinting attack. An attacker builds a two-hop circuit between a shared link and measures the Round Trip Time (RTT), while the victim browses on the same link, visiting different websites. That creates a bottleneck, which reveals the website the victim was accessing. We conducted an open-world experiment and show that an attacker can distinguish between monitored and unmonitored websites, achieving an F_1-score of 94.2 % in a single-class setup and 63.2 % in a multi-class setup, where we differentiate between all websites and unmonitored ones. Additionally, we conducted a closed-world experiment with different exit relays on the Tor network. We demonstrate that an attacker can achieve an F_1-score of up to 66.7 % when distinguishing the top 95 websites using only 150 traces per website. We also show that the security settings in the Tor browser can be an effective countermeasure when set to *safest*, reducing the F_1-score to 1.2 %, but making web surfing unusable.

CCS Concepts

• **Security and privacy → Network security**.



This work is licensed under a Creative Commons Attribution 4.0 International License.
WPES '26, The Hague, Netherlands
© 2026 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-3026-9/2026/11
https://doi.org/10.1145/3847192.3847375

ACM Reference Format:
Mohamed Farrag Gad, Sebastian Pahl, Stefan Gast, Daniel Gruss, and Florian Adamsky. 2026. Side Attack: Off-Path Website Fingerprinting via Cross-Circuit Interference in Tor. In *25th Workshop on Privacy in the Electronic Society (WPES '26), November 15–19, 2026, The Hague, Netherlands.* ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/3847192.3847375

1 Introduction

Onion routing is a technique for anonymizing networks that encrypts network traffic multiple times and sends it through a network via intermediate relays. Tor [13] is the most prominent anonymization network that leverages onion routing with approximately 8 000 000 daily active clients [34]. Hereby, a client chooses a path of three Tor relays (known as a *circuit*), e. g., $A \leftrightarrows B \leftrightarrows C$, and encrypts a message including routing information for each relay in reverse order, i. e., C, B, and then A. This mechanism ensures that each relay knows only its predecessor and successor, without awareness of the entire circuit.

Tor builds circuits telescopically, meaning A builds a TCP connection to B and then B builds a TCP connection to C. If another circuit builds a path, e. g., $D \leftrightarrows B \leftrightarrows C$, the two circuits share a link from e. g., $B \leftrightarrows C$. In that case, the data goes over the same TCP connection as before. This phenomenon is called cross-circuit interference (CCI) [6, 40], and means that Tor multiplexes several circuits, which correspond to separate communication channels, over a single Transport Layer Security (TLS)-secured TCP connection. Thereby, these multiple circuits can influence each other. Alsabah and Goldberg [6] found out that CCI is one of Tor's main performance problems and still a problem today [40]. There are many proposals [5, 7, 18, 46] from the research community to solve this issue, and some of them have already been integrated into Tor, e. g., Kernel Informed Socket Transport (KIST) [28, 31].

One threat against Tor are website fingerprinting (WF) attacks, in which an attacker observes encrypted traffic [25] and infers which websites were accessed, thereby breaking Tor's anonymity. Previous research [22, 41, 48, 49, 56] put the attacker in a Machine-in-the-Middle (MitM) position, capturing encrypted Tor traffic together with meta data, e. g., direction, number, and timing of transmitted packets. Subsequently, the attacker trains a machine learning (ML) classifier with this data and uses the resulting model to identify the website a victim has accessed.

Our paper presents a novel WF attack, in which the attacker is off-path, i. e., not in an MitM position. Our attacker exploits the CCI in Tor by building a two-hop circuit over a link that the victim uses. The attacker sets up a Round Trip Time (RTT) echo server somewhere on the Internet and sends small packets through the shared link from Tor and measures the RTT fluctuation. If an attacker has previously used the RTT fluctuation and trained an ML classifier, the attacker can predict the website that the victim has accessed. We evaluate this attack in different scenarios on the live Tor network. To avoid measuring other Tor users, we discussed our measurements with the Tor Research Safety Board (TRSB) [52] and implemented multiple precautions.

We evaluate a realistic open-world scenario and show that an adversary can distinguish between monitored and unmonitored websites (single class) with an F_1-score of 94.2 % and 63.2 % in a multi-class setup in which we differentiate between all websites and unmonitored. Furthermore, we evaluate a closed-world scenario with this WF attack on three different exit nodes A, B and C with different RTT baselines and achieved an F_1-score of 42.4 % (75 websites), 66.7 % (95 websites), and 14.6 % (90 websites), respectively. Thus showing that our attack depends on the RTT baseline of the exit nodes.

Contributions. In summary, our work makes the following main contributions:

(1) We present a novel off-path side-channel attack on Tor, where an adversary infers visited websites by measuring RTT fluctuations on a shared link.
(2) We conducted an open-world experiment and achieved an F_1-score of 94.2 % (single-class) and 63.2 % (multi-class), and discuss the scalability of the attack across the Tor network, highlighting both its potential impact.
(3) We demonstrate the attack on the live Tor network using a closed-world scenario, achieving an F_1-score up to 66.7 % on the top 95 websites with only 150 traces per site.
(4) We show that setting the Tor Browser's security levels to *safest* effectively mitigates our attack (F_1-score down to 1.2 %) but makes surfing unusable, while showing that *safer* offers little to no protection.

2 Background

This section gives an overview of Tor and CCI.

2.1 Tor

Figure 1 uses the example of Alice to show how to use the Tor client Onion Proxy (OP) to send a message M anonymously to a website. Alice must first download a list of all Onion Router (OR) descriptors from the Directory Authority (DA). This includes each OR's IP address, port number, public key and bandwidth capability. In the second step, Alice selects n ORs based on Tor's path selection algorithm, where n is equal to 3. The algorithm selects ORs with higher bandwidth capabilities more frequently. The path that results from the selected OR is called a *circuit*. Alice first establishes a link to the first OR by sending an `EXTEND` cell, then over the same link, a second `EXTEND` cell to the second OR. This continues in a telescope-like manner until the end. If a link between ORs is already established, no new link is created. Alice creates several circuits

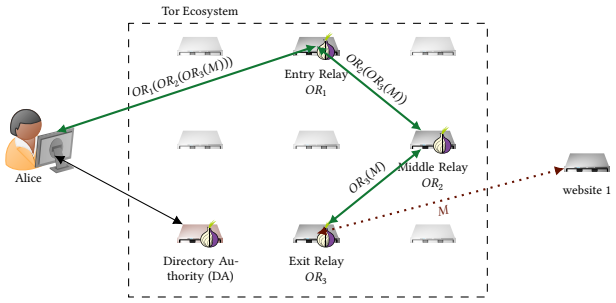


Figure 1: Example of how Tor works. Alice's Tor client first contacts the Directory Authority (DA) to get a list of Onion Routers (ORs). Then it encrypts her message M three times in reverse order and sends it to the first Tor relay, also referred to as the Tor entry relay. The straight green line contains traffic that Tor encrypts, and the dotted red line is traffic that might or might not be encrypted.

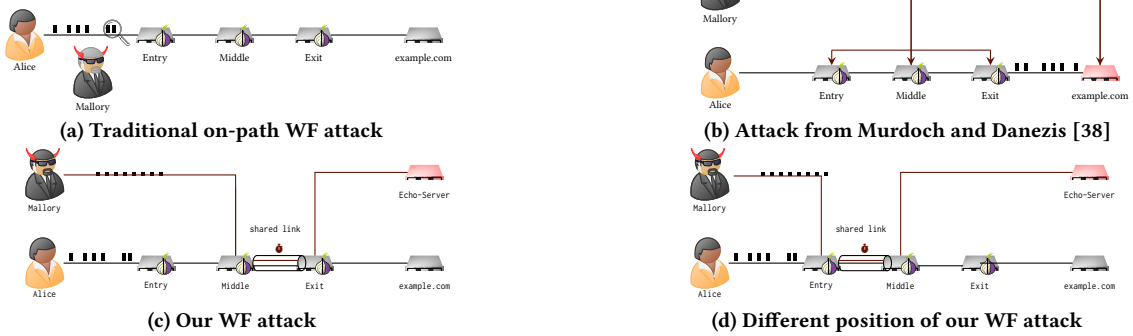
in advance to keep the performance up consistently. The ORs use TLS to communicate with each other. Simultaneously, reliability and in-order delivery are ensured. In the last step, the message is encrypted in reverse order: Alice first encrypts the message M for onion router OR_3 and then for onion router OR_2. This step continues until onion router OR_1 is reached, resulting in an onion-like encrypted message. Finally, this encrypted message can then be sent over to onion router OR_1 where the first layer is decrypted and then onion router OR_1 forwards the encrypted message to onion router OR_2. This process continues until the message M reaches *website 1*. The response from *website 1* is sent back to Alice in the same way.

2.2 Cross-Circuit Interference (CCI)

CCI is a specialization of the head-of-line blocking (HolB) problem, which arises when there is a single data queue and multiple receiver queues. For example, when Alice creates a circuit from $A \leftrightarrows B \leftrightarrows C$ and Bob, on the other hand, creates a circuit from $D \leftrightarrows B \leftrightarrows C$, Alice and Bob share the link from $B \leftrightarrows C$. This is called a *shared link*, which is where CCI can happen. Tor's cryptographic protocols require reliable and in-order data delivery. Otherwise, the cells cannot be decrypted correctly. For this reason, Tor multiplexes multiple circuits, in our example, $B \leftrightarrows C$, over a single TCP-based TLS connection. When a packet from Bob is lost, the packets from Alice also have to wait because they both share one congestion window. How long Alice has to wait depends on the state of the TCP congestion control, i. e., when the packet loss happens at a slow start or in the congestion avoidance phase.

3 Threat Model

In our threat model, an attacker has the same access to the Tor network as normal users; importantly, the attacker is *not* on the network path and can only observe RTT fluctuations over a link, e. g., between a middle and exit relay. Our threat model is depicted in Figure 2c. Users sent traffic through the network as they usually do. By deploying multiple probes, an attacker could observe traffic over monitored links (connections between middle and exit relays) and



(a) Traditional on-path WF attack

(b) Attack from Murdoch and Danezis [38]

(c) Our WF attack

(d) Different position of our WF attack

Figure 2: In a traditional WF attack (a), Mallory is in the position of Alice during the training phase and trains the ML model. During the prediction phase, Mallory needs to be on-path, analyze traffic, and predict which website Alice has visited. In the attack by Murdoch and Danezis [38] (b), the attacker controls a Tor relay and a website (corrupted server) and sends specific patterns over that server, which are detectable by the RTT measurements. In our WF attack (c), Mallory is off-path for both phases and builds a two-hop circuit with a specific middle and exit relay. During the training phase, Mallory collects RTT traces between two relays while performing website accesses from Alice's position. Mallory uses these traces to train an ML model. To predict a website Alice visits, Mallory collects new RTT traces from Alice's traffic and uses the ML model to predict the websites Alice visits. In (d), our website fingerprinting attack measures the RTT fluctuations between guard and middle relay instead of middle to exit relay.

identify that traffic through a a WF attack with a certain probability. With our attack, an adversary can not directly identify a Tor user. However, by knowing the starting time, duration, exit relay IP address, and probable destination of the traffic, an attacker can use such information in other attacks to shrink the anonymity set.

Additionally, by monitoring guard and middle links, an attacker could also trace the path of a user through the network and use the guard IP address as an additional attribute. For example, an administrator of a local network might log network connections to the Internet. With these logs, they can see that a user was connected and sent some data to the Tor network at a particular time. If we have a data source with the above attributes, and combine it with these logs, we could deanonymize a user later on.

4 RTT-based Side-Channel Attack

We assume an off-path attacker (non-MitM) performing RTT measurements between a middle and an exit relay to conduct a website fingerprinting attack. When a victim uses these relays and surfs to a website, the attacker can use the specific RTT fluctuations to predict the website the victim was trying to reach.

4.1 Attack Approach

The attacker chooses a middle and exit relay in the Tor network to observe and builds a two-hop circuit to the chosen relays. By sending a network packet that includes a timestamp to an echo server outside the Tor network, the attacker can measure the RTT between itself and the echo server, including the chosen relays. When the attacker does this repeatedly, they can measure RTT fluctuations between these relays. Suppose now that a victim uses Tor, and the Tor path selection algorithm selects the same relays that the attacker monitors. That is possible, since Tor allows CCI (see Section 2.2), i. e., multiple circuits over a single TCP connection. Since it is a single TCP connection, the victim's packets over this link affect the attacker's repeating RTT measurements. When the attacker has trained an ML model on different websites and their unique RTT fluctuations, they can run a WF attack on that link.

RTT Measurement. We measure the RTT by sending a TCP packet every 10 ms (100 Hz) from the RTT client with the TCP option `TCP_NODELAY` over a two-hop circuit to the RTT server. The two-hop circuit reduces latency by removing one additional relay. The content of this packet is a Unix Time Stamp denoted as T_S encoded as a 64-bit integer. We use the TCP option `TCP_NODELAY` to turn off Nagle's algorithm [39], which collects small packets and sends them at once, potentially introducing additional latency. This TCP option does not affect the middle and exit relay. When the RTT server receives the packet, it just echoes it back. When the client receives the packet, the attacker uses the current timestamp, denoted as T_C, and computes the RTT by subtracting T_S, i. e., $RTT = T_C - T_S$.

ML Training and Prediction Phase. During the training phase, the attacker also acts as the victim and browses different websites, measuring the RTT. Every website creates unique RTT fluctuations (see Figure 3), which the attacker uses to train an ML model. We use a convolutional neural network (CNN) classifier for the training and prediction. Since the RTT measurements in the traces do not arrive at regular intervals, we resample them by doubling the sample rate to achieve uniform spacing. With that, we do not lose any information, because the actual sample rate is also slightly lower. The traces are first truncated to a fixed length of traces to ensure uniform input dimensions.

Afterward, we convert each trace into a Short-Time Fourier Transform (STFT) spectrogram using a Hann window of length equal to the sample rate. The window is moved over the data by 10 data points. Each spectrogram is given to the CNN. The CNN

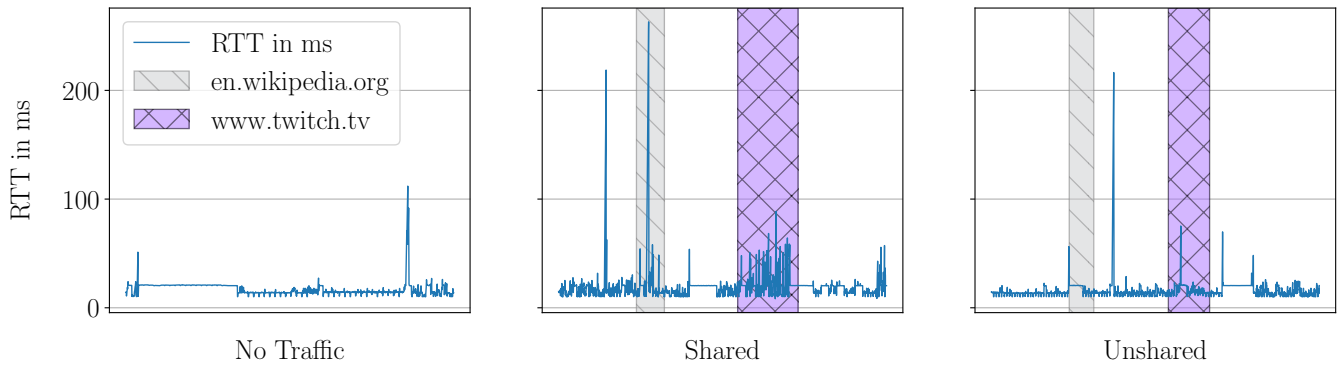


Figure 3: Sample RTT measurements for two websites under different scenarios. *Shared* means the middle and exit relays are the same for the attacker and the victim. *Unshared* means both have the same middle relay but a different exit relay.

architecture consists of three convolutional blocks with 32, 64, and 128 filters, respectively, each followed by a max pooling layer and a dropout layer for regularization. The feature maps are then flattened and passed through two dense layers with 1024 and 512 neurons, followed by a final softmax classification layer. Training uses categorical cross-entropy loss with the Adam optimizer and early stopping with 300-epoch patience to prevent overfitting. We used an 80/20 data split, i. e., we train with 80 % of the data and test and predict with the remaining 20 %. For more details about the architecture of the ML model, we refer interested readers to Table 6 in the Appendix.

4.2 Distinctive RTT Fluctuation Pattern

Every website has a unique RTT fluctuation, which we use as a fingerprint to identify it. In Figure 3, we show the RTT measurements for three scenarios: no traffic, shared, and unshared. If both circuits share the middle and exit relays, we denote that as *shared*; if they do not share the same exit relay, only the middle relay, we denote that as *unshared*. The shared and unshared scenarios involve two different website visits: *en.wikipedia.org* and *www.twitch.tv*.

The first diagram in Figure 3 shows the results without any traffic. The RTT fluctuations without any traffic are constant except for some outliers. In the shared scenario, the grey shaded area (striped) shows a visit to *en.wikipedia.org*, and the lila shaded area (crossed) shows a visit to *www.twitch.tv*. Both websites have unique *RTT* fluctuations. Even in the unshared scenario, both websites remain distinguishable, but the fluctuations are less pronounced.

5 Website Fingerprinting

This section describes our experimental setup and evaluates our attack.

5.1 Experimental Setup

Initial measurements with Shadow [29] show that simulated traffic over a relay can influence the RTT measurements. We wanted to continue with Shadow, but since Shadow has multiple problems, we did our measurements on the Tor network. We refer interested readers to Section 6 in the Appendix for an detailed explanation. However, we reached out to the TRSB [52] before and asked for guidance on how to do measurements on the real network without interfering with other users. We described this process in our ethics section in Section 9.4 in more detail.

For our experimental setup, we rented a virtual server in Germany with 2 Gbit/s bandwidth capabilities and used that as our middle relay. That step is not necessary for the attack, but it gives us more control to avoid measuring RTT measurements from other Tor users by accident. For that, we used two safety mechanisms: First, we set `MaxAdvertisedBandwidth` to `38400`, the lowest value possible. That makes it quite unlikely that the Tor's path selection algorithm chooses our relay. That configuration has no influence on our measurements. Second, we set up a firewall that blocks all traffic except from DAs servers, the prepared victim, and the attacker. Traffic from DAs' servers was necessary; otherwise, the DAs would kick out our server from the consensus, and we could not continue with our measurements. The TRSB [52] suggested the last step. The firewall does not render the first step obsolete, because our relay is still part of the consensus, and other Tor users could still select it for a circuit; however, they then fail to build a circuit. By setting the advertised bandwidth to the lowest value, we reduce the likelihood that our relay gets chosen. The entire configuration of our setup is presented in the Appendix, listed in Listing 1.

Figure 4 depicts an overview of our experimental setup for the training phase. The attacker controls the RTT client, RTT server, both Tor clients, and the browser setup. The browser setup uses the Tor Browser Selenium project [3]. The RTT measurement and the browser setup are located on the same server. Both setups do not influence each other, because we are not bottleneck on the network path, due to a 2 Gbit/s connection. The attack by Gast et al. [17] uses the bottleneck at the last mile and states that intermediate connections have no influence on the RTT measurements. All Tor relays under our control use version 0.4.8.14 (attacker) and 0.4.8.16 (browser).

Selection of Exit Relays. We selected three exit relays with different RTT baseline values for our measurements to test different scenarios. Throughout the paper, we refer to the *RTT baseline* as the average RTT observed when the victim generates no traffic. To select the exit relays, we measured the RTT for every exit relay

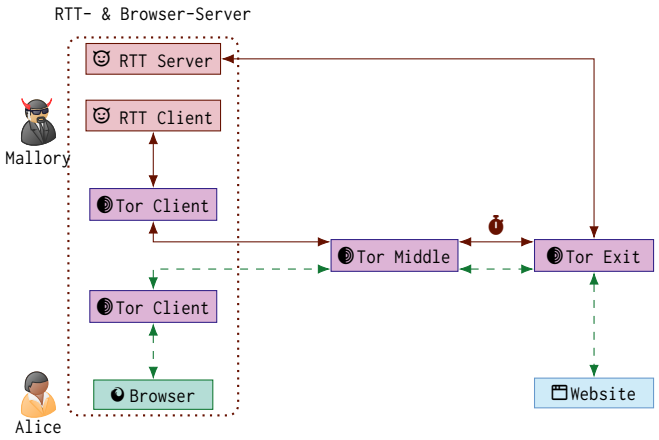


Figure 4: The experimental setup for our side-channel attack. The dashed green lines depict the victim's communication. The straight red lines depict the adversary's communication. The adversary is in control of the RTT client and server and measures the RTT between the middle and exit relay.

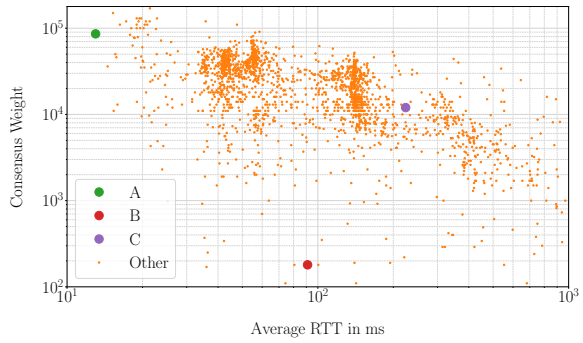


Figure 5: This shows the average RTT in ms against the weight in the consensus measure. The RTT is measured from our middle relay. Our chosen exit relays are marked.

(around 2200) for 30 s. The relays with the lowest variation and RTT on average also have the highest consensus weight.

We selected three exit relays, which we refer to as exit relays *A*, *B*, and *C*. To establish their baseline, we continuously measure the RTT between our middle relay and an exit relay over a 5-minute interval. At the same time, the victim remains idle, and then we compute the arithmetic mean of all collected RTT traces. We selected the exit relays in this way so that we have an exit relay with a short, a medium, and a long RTT baseline. The attacker and the victim, and the chosen exit relays were all in different Autonomous Systems (ASs). The details for our chosen exit relays are listed in Table 1.

Tor Client Configuration and Website Sampling. We use the Tor client configuration in Listing 2 for our fingerprinting attacks, both for the RTT measurements (attacker) and the browsing setup (victim). To automate the Tor Browser, we used the Tor Browser Selenium project [3], which automates the Tor Browser itself. Every browser loads websites differently [36], so using Playwright, which

Table 1: Details of the chosen exit relays in our measurements with their RTT characteristics. All numerical values are in milliseconds.

Exit Relay	Country	Avg.	Var.	Std. Err.	Min.	Max.
A	Germany	13	75.21	0.052	3	259
B	Germany	91	316.50	0.234	53	260
C	Iceland	224	974.76	0.183	136	333

automates Chromium, gives us different results. For all our measurements, we used our own middle relay. We visited the 150 most visited websites from [53]. We iterated through the 150 most-visited websites sequentially, rather than testing each website multiple times. That process took around 158 h (≈ 6.5 d) for one measurement.

Overall, we iterated the list 150 times, i. e., for every website, we collected 150 traces. We wait for 30 s before continuing with the next website. Since we are going through the list one after the other, we took traces at different times, avoiding measuring special conditions (e. g., congestion) multiple times. We save a screenshot for each visit to ensure the website has downloaded completely. After that, we removed websites that resulted in a 404 error, access denied, a CAPTCHA that the user has to solve, or a empty website. Also, we removed identical websites, e. g., amazon.com, amazon.de, and, amazon.in and only kept one of these in our measurements. At the end, we used 150 traces for 95 websites. Table 7 in the Appendix shows all the tested websites.

To avoid additional stress on the Tor network, the victim does not use a guard relay, i. e., it uses a two-hop circuit. However, in Section 5.3, we evaluated a three-hop circuit to make sure that this also works. Additionally, we turn off circuit-level [54] and connection-level padding [55] for the attacker. Circuit-level padding [54] is only active for onion service circuits in the setup phase. Connection-level padding [55] is only active when the connection is idle and only applies to the connection from the client to the guard relay, not between relays. Both do not have an impact on our measurements but they might have an impact in the future, therefore, we disable them as a precaution. For all experiments, we use only one circuit for the attacker and the victim, so traffic splitting (e. g., Conflux [4, 12, 21]) does not interfere with our measurements, as discussed in Section 9.3.

5.2 Closed-world Scenario with Different Exit Relays

The closed-world scenario defines that the victim can only visit a predefined number of websites. The attacker tries to predict which website the victim has accessed. We utilized three distinct exit relays, labeled *A*, *B*, and *C* (see Table 1 for their characteristics). We conducted these measurements to evaluate the effectiveness of the attack in relation to the chosen exit relay. The *A*, *B*, and *C*, demonstrated minimal variation in RTT, ensuring a consistent comparison across the different exit relays. We conducted all closed-world measurements with the *standard* security level. Because each exit relay blocks a distinct set of websites, the number of websites

Table 2: Prediction Results for the Exit Relays A, B, and C.

Exit Relay	# of Websites	TP	FP	Acc. [%]	F_1 [%]
A	75	932	1264	42.44	42.44
B	95	1891	942	66.75	66.75
C	90	392	2290	14.62	14.62

used during training differs for each relay. For a detailed view of the websites associated with each relay, see Table 7.

To examine the results in detail, see Table 2 and Figure 6 for a tabular and graphical comparison, respectively. Exit relay B demonstrated the highest performance with an accuracy and F_1-score of 66.75 %, correctly identifying 1891 true positives while maintaining relatively low false positives (942) across 95 monitored websites. In contrast, exit relay A achieved moderate performance with 42.44 % accuracy when monitoring 75 websites, producing 932 true positives but suffering from a higher false positive rate (1264). Exit relay C exhibited notably poor performance, despite monitoring 90 websites, achieving only 14.62 % accuracy with merely 392 true positives, which were dominated by 2290 false positives. Our evaluation of our WF attack across three different Tor exit relays reveals variability in attack effectiveness.

One question that arises is why our attack's effectiveness is so different. Every exit relay has a different RTT baseline, which influences the website visits. If the RTT baseline decreases, websites load faster; if it increases, websites load slower. Due to this loading behavior, the resulting RTT fluctuations are shrunk or stretched in time. Therefore, an ML model trained on a low RTT baseline cannot correctly predict websites from a high RTT baseline, or vice versa. That means, the ML model is sensitive to the website loading time. To correct that, we would need to train the model on many different RTT baselines and other network conditions. However, an attacker can easily rent servers close to the relay to achieve a low RTT.

5.3 Influence of Real Scenario (Three-Hop)

In the previous measurements, the victim has used a two-hop circuit instead of a three-hop circuit. Reason behind that was to avoid additional stress on the Tor network and to keep the experimental setup as simple as possible. However, Tor builds circuit with three ORs by default. For this reason, we wanted to repeat our attack while a victim uses a three-hop circuit to visit a website.

We used *B* relay as the exit for this experiment and used a high-consensus guard relay located in Germany. For each website we collected 60 traces. Surprisingly, the three-hop (guard → middle → exit) F_1 score was very close to the two-hop (middle → exit) F_1 score where we used 150 traces. Consequently, we hypothesize that the three-hop circuit can achieve a higher score than a two-hop circuit using the same number of traces. Refer to Figure 10, which illustrates the relationship between F_1 and the number of traces used. Figure 10 shows that as the number of traces increases during classification, the F_1 score also improves. In the three-hop experiment with relay B as the exit relay, the results showed 729 true positives (TP), 401 false positives (FP), with an accuracy of 64.41 % and F_1 score of 64.41 %.

5.4 Influence of the Security Levels of the Tor Browser

In this section, we analyze which influence the different security level of the Tor browser has on the RTT side-channel attack.

5.4.1 Security Levels in the Tor Browser. The Tor browser offers three security levels with progressively stricter privacy protections, namely *Standard*, *Safer*, and *Safest*. A screenshot of that setting is depicted in Figure 7. The security level can only be set globally, not per website. However, there is a possibility to define custom settings which overrides the other security levels. Each security level represents a trade-off between functionality and privacy protection, with the Standard level providing maximum compatibility and the Safest level offering enhanced security at the cost of reduced website functionality. According to the Tor Browser manual [1], these levels can be distinguished as follows:

Standard is the default setting in which the Tor Browser is delivered. In this setting, all website and Tor Browser features are enabled. It offers the most functional browsing experience.

Safer turns off dangerous website features, which could reduce its functionality. For example, JavaScript is disabled on all websites that do not provide HTTPS. Additionally, some fonts and math symbols are blocked, and HTML5 media elements all become click-to-play, i. e., they are only loaded once clicked.

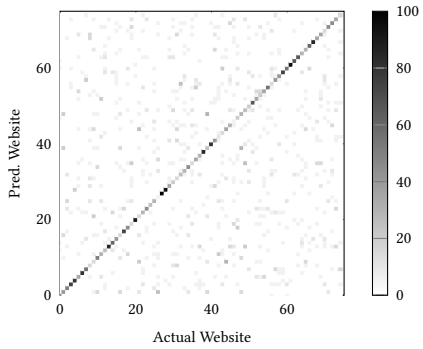
Safest only allows static websites without dynamic content. That means JavaScript is turned off for all websites by default. Additionally, more fonts, icons, math symbols, and images are blocked. The HTML5 media elements remain in the click-to-play setting.

5.4.2 Experimental Setup. For all our measurements in this section, we used our own middle relay, as described in Section 5.1, and as an exit relay, we used relay *B*. We visited the 150 most visited websites in [53] with the three different security levels, namely *standard*, *safer*, and *safest*. As before, we iterated through the 150 most visited websites sequentially instead of testing one website multiple times.

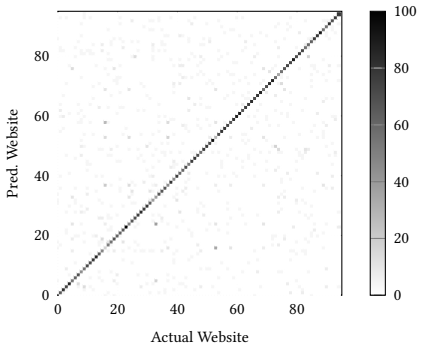
6 Problems with Shadow Measurements

Our first idea was to run Shadow simulations. We conducted our first experiments in Shadow; our RTT measurements show apparent differences, which indicate that our attack can be effective. The problem is that if we download files of 1 MiB to 2 MiB to simulate downloading a website. We observe significant spikes in the RTT measurements in Shadow, with a relative difference of around 200 ms compared to the ground floor. In comparison, if we download the Linux kernel over a Tor circuit in the real network and measure the RTT, the difference is only around 20 ms to 30 ms to the ground floor. Compare Figure 8 with Figure 3. The *Unshared (Middle)*-scenario in Figure 8 can be compared to *No Traffic*-scenario in Figure 3, because we used an exit relay that has a high consensus weight. With that, we should see constant RTT fluctuations, but we do not. Thus, experiments in Shadow for our particular attack do not reflect the real network.

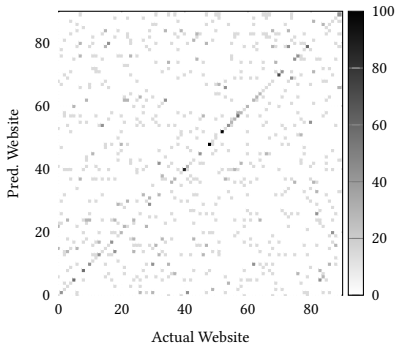
Another reason is that we cannot reach the Internet and do website fingerprint measurements in Shadow. However, Jansen et al. [32] published *Retracer* in which the authors showed that website



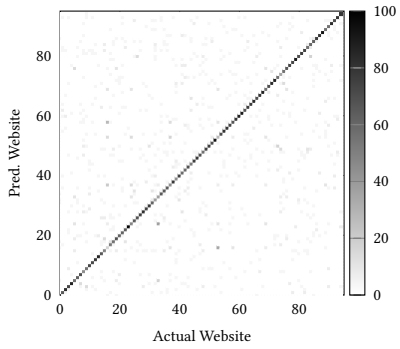
(a) Number of predictions for 75 websites with the exit relay A. The overall F_1 score is 42.44 %.



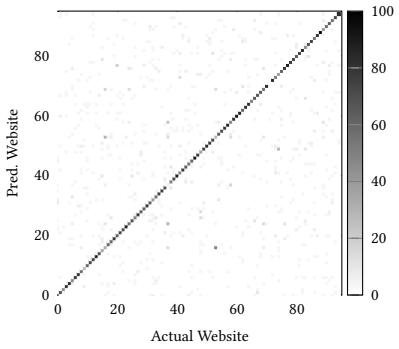
(b) Number of predictions for 95 websites with the exit relay B. The overall F_1 score is 66.75 %.



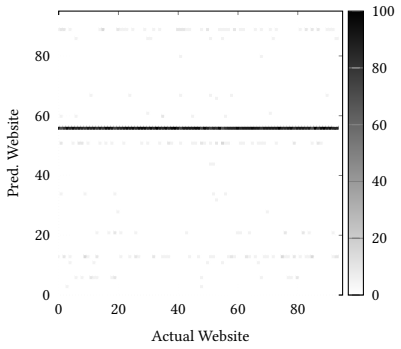
(c) Number of predictions for 90 websites with the exit relay C. The overall F_1 score is 14.62 %.



(d) *Standard* security level with an overall F_1 score of 66.75 %.



(e) *Safer* security level with an overall F_1 score of 67.09 %.



(f) *Safest* security level with an overall F_1-score of 1.2 %.

Figure 6: Subfigure *(a)–(c)* shows the number of predictions for each website measured from our middle relay to exit relay A, B, and C. Subfigure *(d)–(f)* shows the number of predictions for 95 websites with different security levels measured from our middle relay to exit relay B. For all subfigures, the actual website is shown on the x axis, the website predicted by the model is shown on the y axis.

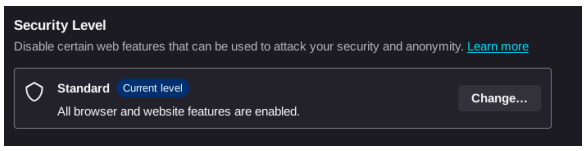


Figure 7: Screenshot of the security level in the Tor Browser. Users can change it in the settings menu under ☰ › Settings › Privacy & Security.

fingerprinting attacks can also be conducted in Shadow. However, the authors did not publish the source code on purpose, since an attacker could use Retracer to train a model to conduct real-world attacks. Retracer focuses on a traditional website fingerprinting attacker (see Figure 2a), while we focus on an off-path attacker (see Figure 2c). Shadow does correctly model the packet and cell level, which is necessary for traditional WF attacks, however it seems not to model the side channel correctly. Later, we can use these insights and incorporate them into Shadow.

7 Feasibility of Two-Hop RTT Measurements

We also investigated measuring RTT fluctuations from an alternative position: from the guard to the middle relay (fig. 2d). This scenario is inherently unlikely because it requires the middle relay to possess an *exit* flag to reach the attacker's echo server, contradicting standard path selection algorithms which typically assign exit flags only to exit relays.

To quantify this limitation, we used SALSA [40] to generate 10 million circuits. We analyzed two specific configurations for a two-hop attack where the attacker is unconstrained by path selection:

- Forward measurement: The attacker measures from guard → middle. This requires the middle relay to hold both *Guard* and *Exit* flags.
- Reverse measurement: The attacker measures from guard ← middle. This requires the guard relay to hold both *Guard* and *Exit* flags.

Our analysis reveals that only approximately 7.4 % of the generated circuits satisfy the requirements for the forward scenario, and similarly, only 7.4 % satisfy the reverse scenario. Even when

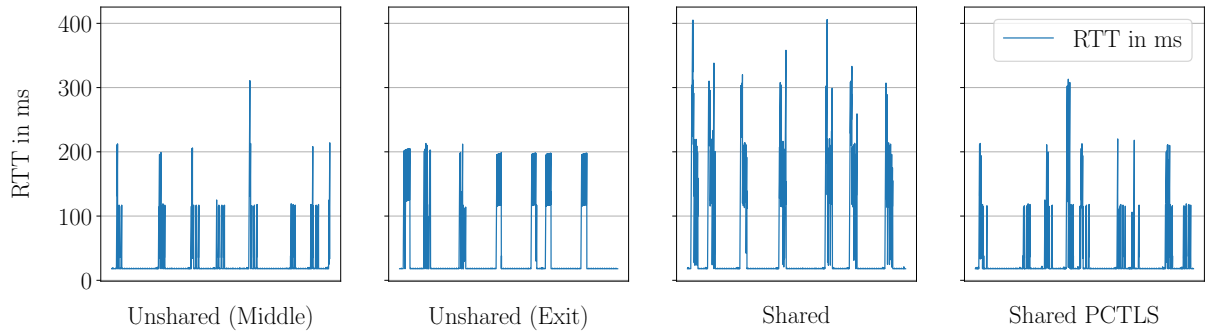


Figure 8: Shows the result of multiple Shadow simulations within four scenarios. We measure the RTT, while sending artificial traffic between 1 MiB and 2 MiB with a spacing of 1 s to 10 s. Each scenario has a duration of 1 min. *Unshared (Middle)* means the middle relay is the same for the attacker and the victim. *Unshared (Exit)* means the exit relay is the same for the attacker and the victim. *Shared* means the middle and exit relays are the same for the attacker and the victim. *Shared PCTLS* means the middle and exit relays are the same for the attacker and the victim; however, all relays use Per-Circuit TCP-over-TLS [40].

combining these cases and accounting for overlap, the attacker can only perform guard-to-middle measurements in 13.7 % of all circuits. Consequently, the distribution of relay flags and path selection policies significantly restricts the viability of this attack vector. Note that while we focused on two-hop circuits, an attacker could theoretically utilize three-hop circuits, though this remains outside the scope of our current investigation.

8 Influence of Noise to the Prediction

We investigate how stable the WF attack is against background traffic, i. e., noise. Therefore, we repeated our measurements while running background traffic. To generate background traffic, we utilize `TGen` [19], which is configured by `tornettools` [51] to generate realistic client traffic in Shadow [29] simulations. We configure one `TGen` client for our experiments. We consider the following noise configurations:

Shared is a configuration in which `TGen` runs on the same circuit that the victim uses, i. e., (middle ⇆ exit).

Unshared is a configuration in which `TGen` shares only the middle relay with the victim but has a different exit relay. We only tested the middle relay; however, in our experiments with Shadow (see Figure 8), the patterns were the same for the exit relay, indicating that it would not make a significant difference in the prediction.

For each configuration, we collected 75 traces. Using the exit relay B model trained using the standard security level, we evaluated its performance on both the shared and unshared traces. As anticipated, the unshared scenario achieved the highest score with an F_1-score of 41.69 %, whereas the shared scenario performed worse with an F_1-score of 10.72 %. In the shared case, the noise generated by `TGen` influenced the RTT, making it difficult for the model to recognize the underlying pattern. Conversely, in the unshared case, we observed no noticeable RTT distortion caused by the noise between victim visits. The detailed results are shown in Table 3.

8.0.1 Results. Table 4 displays the results for each measurement using each of the security levels and Figures 6d to 6f visualize the

Table 3: Performance of the exit relay B model under different noise configurations.

Configuration	TP	FP	Acc. [%]	F_1 [%]
Shared	145	1207	10.72	10.72
Unshared	587	821	41.69	41.69

Table 4: Prediction Results measured from our middle relay to the exit relay *B* depending on different Security Levels.

Security Level	TP	FP	Acc. [%]	F_1 [%]
Standard	1891	942	66.75	66.75
Safer	1906	935	67.09	67.09
Safest	33	2789	1.17	1.17

data for the security levels' *standard*, *safer*, and *safest*, respectively. On exit relay B, the *safest* security level neutralized the attack, resulting in a very low classification F_1 score of 1.2 %. The model cannot distinguish between any classes; therefore, it assigns the same class deterministically to every website. Because every class is equally represented in the data, the 1.2 % is very close to random guessing. This artifact can occur during the training of an ML classifier. This reduction in effectiveness can be attributed to the security level's default JavaScript disabling and strict feature restrictions, as documented in the Tor Browser manual [1]. Examination of screenshots revealed widespread website functionality degradation, with trace visualizations showing nearly identical *RTT* patterns across all traces; essentially flat baselines with minimal variation. Switching to the *safest* security level is a countermeasure against our attack.

The *safer* security setting actually provided the highest attack performance, slightly surpassing the *standard* setting. However, the difference is minimal, indicating that switching from *standard* to *safer* has no impact on the attack's effectiveness. The taken screenshots indicate that all websites loaded normally in the *safer*

Table 5: Results for the open-world experiments for exit relay *A* & *B*.

Exit Relay	Monitored	Unmonitored	TP	FP	FN	TN	Pre. [%]	Rec. [%]	Acc. [%]	F_1 [%]
A	75	1878	1356	274	115	102	83.30	89.87	86.46	79
B	95	1896	906	58	65	321	93.98	93.31	93.64	90.9

setting. These findings highlight the interplay between the Tor browser security configurations and the website fingerprinting attack efficacy.

8.1 Results for the Open-World Scenario

What we have conducted until now is considered a *closed-world* setting, where classification is limited to specific websites. However, this scenario has been criticized as unrealistic [33, 43], because no attacker can create an ML model with every website. Since then, *open-world* scenario is seen as a more realistic attacker scenario. According to Panchenko et al. [41] there are two different fluctuations, namely *single-class* and *multi-class*.

8.1.1 Single-Class Scenario. Single-class differentiates between two sets of websites, namely monitored and unmonitored websites. That scenario reflects a censor that does not care about specific websites, but rather that someone has accessed a website from a blocked list. We followed the methodology from Sirinam et al. [49] and collected a dataset of unmonitored websites by visiting each website once. We gathered these websites from the same source used for our monitored data [53], which provided up to 1000 websites, and supplemented this with an additional 2000 websites from [2], which lists the top 2000 visited websites in the US. After removing duplicates and our original 150 monitored websites, we were left with 2218 unmonitored websites; 1878 of them had one successful visit from exit relay A, and 1896 had one successful visit from exit relay B.

Note that this experiment was conducted solely at the standard security level of the Tor browser. We performed binary classification, labeling monitored traces as 1 and unmonitored traces as 0. We trained the open-world model the same way as we trained the closed-world model; by converting each trace into a spectrogram and feeding them into a CNN. This experiment was conducted on exit relay A and B. As noted in Section 5.2, the number of classes (i. e., monitored websites) differs in the open-world scenario because each exit relay blocks a distinct set of sites; consequently, the training classes vary per relay. A detailed overview of the websites associated with each relay is provided in Table 7. Table 5 shows our results.

Table 5 shows that we achieve an accuracy of 93.64 % and an F_1-score of 90.9 % for exit relay B. These values suggest the attack can accurately distinguish between monitored and non-monitored websites with minimal false positives. Exit relay A showed lower but still high accuracy of 79 % and F_1-score of 86.46 %, though with more false positives (274), leading to reduced precision of 83.30 %. The variation in performance between relays likely reflects differences in network conditions or relay characteristics. However, both relays maintained recall rates above 89 %, demonstrating that the attack consistently identifies monitored websites.

8.1.2 Multi-Class Scenario. Multi-class is similar to the closed-world scenario, except that we have an additional class called *unmonitored*. In this scenario, the attacker not only wants to know if the victim has accessed a monitored or unmonitored website, but also wants to know which monitored website was accessed. We achieved an F_1-score of 63.2 % for exit relay B, which includes 2218 unmonitored websites. For the exit relay A, we achieved an F_1-score of 38.1 % with 1896 unmonitored websites.

9 Discussion

This section discusses the limitations and the ethical principles of our study.

9.1 Scalability of our Attack

Let us assume an attacker wants to observe the maximum number of circuits possible with the least amount of resources. An attacker needs to balance the time it takes to observe the largest number of victims, the required bandwidth on the side of the attacker, as well as the consumed bandwidth in the Tor network. We simulated this scenario to argue what an attacker could observe.

We simulated 1.5 M circuits with SALSA [40] to get the top N relays in the middle and exit position. To compute the probability of how many circuits the attacker can observe while picking the top N relays, we take the cumulative probability of the top N in both positions (P_m, P_e) and multiply them $P_m \times P_e$.

To get how much bandwidth the attack consumes in Tor network and how much bandwidth the attacker requires, we assume 100 samples per second (denoted as S), 20 B [44] for the IP header in version 4 (I), 20 B [10] for the TCP header (T), and a Tor cell size of 514 B (C). Each RTT probe-packet requires a 64-bit timestamp, that means, it has a size of $E = I + T + 8\,\text{B} = 48\,\text{B}$, outside of Tor, and as probe-cell a size of $O = I + T + C = 554\,\text{B}$. The attacker sends and receives one probe-cell and sends and receives one probe-packet, which means the attacker requires a bandwidth of $A = (O + E) \times S \times 2 = 120.4\,\text{kB/s}$ per circuit. With a two-hop circuit in Tor, the middle relay sends and receives one probe-cell from the attacker and the same on the backward path from the exit relay. The exit relay sends and receives one probe-cell as well as one probe-packet. Therefore, the attack consumes $R = (O+O+O+E)\times S\times 2 = 342.0\,\text{kB/s}$ per circuit in Tor. We multiply A and R by the number of circuits we want to monitor.

To get the time duration how long it takes to observe 50 % of users at least once if every user builds a new circuit every 10 min, we use SALSA [40]. We already know the top N relays in both positions from the previous simulation. In a loop, we create 10 000 circuits, each representing a user, and we count each unique user where the two circuit positions are from the top N. We stop the loop if over 50 % of users had a circuit in the top N. By doing that for different top N values, from 10 to 1000, we get for each, the number of iterations until we reach 50 %. Afterwards, we can multiply the number of iterations by 10 min to get an estimate of the time duration. Our results of this simulation can be seen in Figure 9.

Figure 9 shows how many circuits an attacker could observe if the attacker picks the top N relays in the middle and exit position and monitors all possible combinations. Additionally, the figure

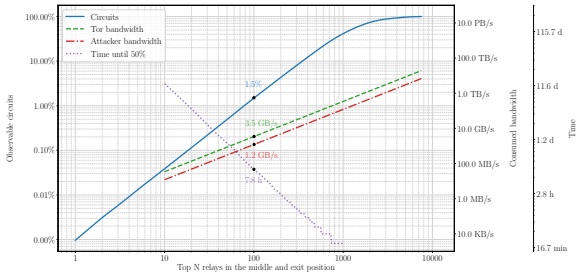


Figure 9: How many circuits an attacker could observe, the impact on the bandwidth, and the time duration. The x-axis refers to the top N most likely relays in the middle and exit position that an attacker could pick. All lines share the same x-axis but use a different y-axis. The blue line, which is the observable circuits in percent, uses the y-axis on the left side. The green and red lines, Tor and attacker bandwidth, use the first y-axis to the right. The lila line, time duration until 50 %, uses the second y-axis to the right.

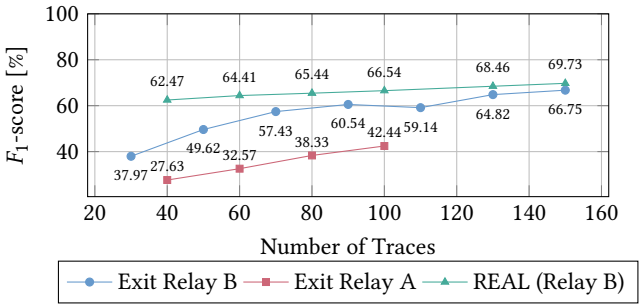


Figure 10: F_1-score depended on the number of traces for exit relays *A* and *B* and for the REAL experiment on exit relay *B*. The results indicate that increasing the number of traces improves the classification performance.

shows the impact on the bandwidth of Tor and the attacker, as well as the time duration it takes to observe 50 % of all users at least once. The top N relays are the most likely in that position. For example, suppose the attacker picks the top 100. In that case, the attacker can monitor 1.5 % of all circuits, which requires around 1.2 GB/s of bandwidth for the attacker, consumes around 3.5 GB/s in the Tor network and takes around 7.8 h to observe 50 % of all users at least once.

However, there are a couple of problems with this approach. Picking the most likely relays implies that the monitored links are more likely to be shared. A shared link is not a problem as long as there are no overlapping requests. The probability of overlapping requests needs to be low, and we need to determine this probability empirically. That depends on how many active shared links exist for the average duration of a website visit. Other researchers [30, 40] shows that around 100 k circuits per second or 1.5 M circuits within 10 min are active. Additionally, picking the busiest relays can produce noise in the RTT measurements that could impact the classification results. RTT probes might influence each other; the attacker might need multiple servers to monitor all links.

9.2 Countermeasure

There have been plenty of works [5, 7, 18, 46] that solve the CCI problem by creating a new connection for every circuit. Therefore, two circuits cannot share a link or interfere with each other. The same work would also countermeasure our side-channel attack because the attacker and the victim are on different connections. We tested that with two different exit relays; see *Unshared* in Figure 3, but not with two connections to the same exit relay, which can not be easily done on public networks. The problem is that by creating a new connection for every circuit, socket exhaustion attacks [18] are possible.

Our experiments in Section 5.4.1 show that it is also possible to countermeasure the attack by setting the appropriate security level. According to our measurements, the security level *safer* does not protect against this attack, only the level *safest*. The problem with this security level is that it only allows static websites since it disables JavaScript. According to [26], out of millions of websites, 98.9 % of these use JavaScript as the client-side programming language. That means that with this setting enabled, most of the websites are no longer usable.

9.3 Limitations

9.3.1 Measurement Position. We also measured RTT fluctuations from a different position i. e., from guard to middle relay (Figure 2d). However, this scenario is unlikely because it requires the middle relay to hold an exit flag to reach the echo server, which contradicts standard path selection algorithms. We refer interested readers to Section 7 in the Appendix, where we verified that with a simulator.

9.3.2 Influence of the Number of Traces. We hypothesize that the F_1-score of our classifier is influenced by the number of traces utilized during the classification phase (Section 4.1). To validate this hypothesis, we experimented by comparing the F_1 scores obtained for Relays A and B with varying numbers of traces. The results, shown in Figure 10, reveal a clear positive correlation between the number of traces and the achieved F_1 score. Specifically, as the number of traces used in the classification phase increases, the F_1 score consistently improves. This trend suggests that our attack can benefit significantly from a larger number of traces, resulting in higher classification accuracy.

9.3.3 Influence of Traffic Splitting. Tor has implemented Conflux [4, 12, 21], a traffic-splitting technique to countermeasure website fingerprinting attacks. Conflux builds two circuits to a single exit relay and switches between them when the RTT gets smaller for one circuit. This improves performance and can hinder traffic analysis for guard and middle relays. In our experiments, we only build one circuit, therefore it had no influence. However, in real-world scenarios, Conflux could make our attack more difficult. Our hypothesis is, that if an attacker trains an ML with more traces then we did, including half traces, that our attack can work with Conflux as well. We leave it for future work to investigate the impact of traffic splitting.

9.4 Ethical Principles

Before conducting our experiments, we reached out to the TRSB [52] for guidance on how to carry out our measurements on the live Tor network, ensuring that we do our measurements safely and avoid accidentally measuring traffic from other Tor users. We explained our idea, our experimental approach, and the precautions we plan to implement. We received highly valuable feedback from the TRSB [52], which we implemented. To protect the privacy of other Tor users, we took multiple precautions. First, we set up our own Tor middle relay to have finer control over which users access our relay. That step was not necessary for the attack, but only to protect other users. Second, we created a firewall rule that only allowed traffic from DAs relays (IPv4 and IPv6) to stay in the consensus and from our prepared victim, and blocked all other traffic. We also manually set the consensus weight of our OR to the minimum value (38 400 B/s) to prevent Tor users from using our OR for a circuit. That setting does not influence the performance and our RTT measurements. Due to the firewall rule, our relay is still in the consensus.

10 Related Work

In this section, we provide related works about website fingerprinting attacks and Tor traffic analysis.

10.1 Website Fingerprinting Attacks

Previous works have demonstrated website fingerprinting attacks in different scenarios. Some of them exploited side channels in the CPU microarchitecture [23, 45, 57], while many others [8, 9, 22, 24, 25, 42, 47] rely on network traffic analysis. These attacks typically assume an MitM, capable of passively observing certain features of the traffic between the website and the victim, such as the sizes [9, 25], the directions and order [22, 42] of packets, or the shape of the traffic, i. e., the traffic volume over time [47, 50]. The structure of traditional WF attacks can be seen in Figure 2a. Therefore, WF attacks usually target privacy-enhancing network tunnels, as otherwise an MitM attacker can easily infer the website visited from the unencrypted IP addresses.

Traditional fingerprinting methods are challenging to apply to Tor because they rely on fixed-size cells and multiplex multiple streams over a single connection. However, Shi and Matsuura [48] demonstrated a website-fingerprinting attack using a malicious entry relay. Their method segments traffic into intervals and constructs fingerprint vectors from observed packet sequences. They use a similarity metric to match these vectors with pre-established fingerprints of target webpages. Additionally, a few off-path website fingerprinting attacks have been demonstrated. Instead of requiring an an MitM attacker, the off-path attacker indirectly observes traffic from a benign connection by measuring response latencies to probe packets. Gong et al. [20] demonstrated a website fingerprinting attack based on ICMP Echo requests. Their attack does not apply to Tor because it does not support the transmission of ICMP packets.

The authors in [15, 17] demonstrated video and website fingerprinting attacks that also work off-path based on TCP's ACK mechanism influenced by a network path bottleneck, which is usually at the last mile to the user. By slowly loading an image or other data over a TCP connection, ACKs are delayed by other traffic at the bottleneck, especially during higher volumes. Effectively, they measure the RTT, and the variation of this value is the side channel. They can not directly identify users, but monitoring specific websites is possible [20]. In a follow-up [16], they demonstrated a countermeasure based on delayed TCP ACKs.

10.2 Traffic Analysis in Tor

Panchenko et al. [42] performed local website fingerprinting, meaning they measured the volume, timing, and direction between the Tor client and the entry relay. After they collect the data, they use a Support Vector Machine (SVM) for training and prediction. Their threat model is an adversary between the Tor client and the entry relay, e. g., a local administrator or an Internet Service Provider (ISP). Murdoch and Danezis [38] measures the congestion of a Tor relay via RTT measurements and builds a circuit through the target relay and back to their relay. Their attack setup is depicted in Figure 2b They control not only the relay but also a corrupted server, which introduces congestion, which they want to detect via their RTT measurements. If a user connects to their corrupted server, they can detect the pattern on every relay they measure. Their goal is to track a user's path through the Tor network.

The attack from Evans et al. [14] is an improvement of Murdoch and Danezis [38]. They inject JavaScript into the response at an exit relay under their control to determine whether a victim's circuit is congested. They do that by sending an HTTP request every second. Then they target specific relays to identify which are in the victim's circuit. By building a longer circuit that loops back on itself, they reduce the bandwidth required by an attacker. The attack from Mani et al. [35] replicates Murdoch and Danezis [38] and confirms its viability by rapidly downloading large files over a circuit.

The attack from Hopper et al. [27] tries to find the location of the user's client via RTT measurements. Mittal et al. [37] utilizes throughput as a side channel to determine if two connections originate from the same client. They measure the throughput of the relays every 100 ms. Cangialosi et al. [11] introduce a technique to measure the RTT between pairs of relays without modifying the Tor client. At first, they construct a four-hop circuit with the ORs $W \leftrightarrows X \leftrightarrows Y \leftrightarrows Z$. The ORs W and Z are located on the measurement server; and; $X \leftrightarrows Y$ is the relay pair under measurement. They measure the latency of the whole circuit, then the latency between the two-hop circuits ($X \leftrightarrows W$), and finally, the latency between the two-hop circuits ($Y \leftrightarrows Z$). Tor does not allow one hop-circuit; otherwise, the setup would be simpler. Each measurement is performed with an end-to-end echo client and server.

None of these papers use CCI as a side channel to conduct website fingerprinting attacks by measuring RTT.

11 Conclusion and Future Work

In this work, we presented a novel WF attack that exploits the CCI, allowing an attacker to conduct off-path attacks, i. e., no MitM position is needed. An attacker establishes a two-hop circuit on a link between two Tor relays, measures RTT fluctuations, and then predicts the website the victim has accessed. We present our experimental results on the real network, demonstrating that these attacks are practical. Our attack achieves an F_1-score of 94.2 % for open-world in a single-class setting, 63.2 % for open-world in a

multi-class setting, and up to 66.7 % in distinguishing the top 95 websites for closed-world.

Future work should investigate the influence of traffic-splitting techniques on the CCI side-channel attacks. We did not optimize the classifier and assume that classification results can be improved by using the Dynamic Time Warping (DTW) algorithm, which finds similarities even when two temporal sequences vary in speed. In addition, the RTT measurements could be improved by synchronizing with KIST to further reduce noise.

Acknowledgements

This research is supported in part by the European Union under grant number ROF-SG20-3066-3-2-2, the European Research Council (ERC project FSSec 101076409), and the Austrian Science Fund (FWF SFB project SPyCoDe 10.55776/F85). Any opinions, findings, conclusions, or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of the funding parties. The authors used Grammarly and Claude to revise text and to correct grammatical errors.

References

[1] URL: https://tb-manual.torproject.org/security-settings/ (visited on 08/12/2025).

[2] URL: https://sitestatsdb.com/top-2000-website (visited on 08/12/2025).

[3] Gunes Acar, Marc Juarez, and individual contributors. *tor-browser-selenium - Tor Browser automation with Selenium*. 2023. URL: https://github.com/webfp/tor-browser-selenium (visited on 04/14/2026).

[4] Mashael AlSabah, Kevin Bauer, Tariq Elahi, and Ian Goldberg. "The Path Less Travelled: Overcoming Tor's Bottlenecks with Traffic Splitting". In: *PETS*. 2013. DOI: 10.1007/978-3-642-39077-7_8.

[5] Mashael AlSabah and Ian Goldberg. "PCTCP: Per-Circuit TCP-over-IPsec Transport for Anonymous Communication Overlay Networks". In: *ACM CCS*. 2013. DOI: 10.1145/2508859.2516715.

[6] Mashael Alsabah and Ian Goldberg. "Performance and Security Improvements for Tor: A Survey". In: *ACM Computing Surveys* 49.2 (Sept. 21, 2016), pp. 1–36. ISSN: 03600300. DOI: 10.1145/2946802.

[7] Lamiaa Basyoni, Aiman Erbad, Mashael Alsabah, Noora Fetais, Amr Mohamed, and Mohsen Guizani. "QuicTor: Enhancing Tor for Real-Time Communication Using QUIC Transport Protocol". In: *IEEE Access* 9 (2021). DOI: 10.1109/ACCESS.2021.3059672.

[8] Sanjit Bhat, David Lu, Albert Kwon, and Srinivas Devadas. "Var-CNN: A Data-Efficient Website Fingerprinting Attack Based on Deep Learning". In: *PETS*. 2019. DOI: 10.2478/popets-2019-0070.

[9] George Dean Bissias, Marc Liberatore, David Jensen, and Brian Neil Levine. "Privacy Vulnerabilities in Encrypted HTTP Streams". In: *PETS*. 2006. DOI: 10.1007/11767831_1.

[10] David Borman. *TCP Options and Maximum Segment Size (MSS)*. Tech. rep. 6691. July 2012. 9 pp. DOI: 10.17487/RFC6691. URL: https://www.rfc-editor.org/info/rfc6691.

[11] Frank Cangialosi, Dave Levin, and Neil Spring. "Ting: Measuring and Exploiting Latencies Between All Tor Nodes". In: *ACM IMC*. Oct. 28, 2015. DOI: 10.1145/2815675.2815701.

[12] Wladimir De La Cadena, Asya Mitseva, Jens Hiller, Jan Pennekamp, Sebastian Reuter, Julian Filter, Thomas Engel, Klaus Wehrle, and Andriy Panchenko. "TrafficSliver: Fighting Website Fingerprinting Attacks with Traffic Splitting". In: *ACM CCS*. 2020. DOI: 10.1145/3372297.3423351.

[13] Roger Dingledine, Nick Mathewson, and Paul Syverson. *Tor: The Second-Generation Onion Router*. Tech. rep. 2004. URL: https://svn-archive.torproject.org/svn/projects/design-paper/tor-design.pdf (visited on 02/28/2025).

[14] Nathan S. Evans, Roger Dingledine, and Christian Grothoff. "A Practical Congestion Attack on Tor using Long Paths". In: *USENIX Security Symposium*. USA, Aug. 10, 2009.

[15] Stefan Gast, Roland Czerny, Jonas Juffinger, Fabian Rauscher, Simone Franza, and Daniel Gruss. "SnailLoad: Exploiting Remote Network Latency Measurements without JavaScript". In: *USENIX Security Symposium*. 2024. ISBN: 978-1-939133-44-1.

[16] Stefan Gast, Simone Franza, Martin Heckel, Jonas Juffinger, Daniel Gruss, and Johanna Ullrich. "Client-Side Mitigation of Remote Latency Side-Channel Attacks". In: *DIMVA*. 2026.

[17] Stefan Gast, Nora Puntigam, Simone Franza, Sudheendra Raghav Neela, Daniel Gruss, and Johanna Ullrich. "Zero-Click SnailLoad: From Minimal to No User Interaction". In: *ESORICS*. 2025.

[18] John Geddes, Rob Jansen, and Nicholas Hopper. "IMUX: Managing Tor Connections from Two to Infinity, and Beyond". In: *WPES*. Nov. 3, 2014. DOI: 10.1145/2665943.2665948.

[19] *GitHub: TGen*. July 22, 2022. URL: https://github.com/shadow/tgen (visited on 09/13/2022).

[20] Xun Gong, Nikita Borisov, Negar Kiyavash, and Nabil Schear. "Website Detection Using Remote Traffic Analysis". In: *PETS*. 2012. DOI: 10.1007/978-3-642-31680-7_4.

[21] David Goulet and Mike Perry. *329-traffic-splitting - Tor design proposals*. Nov. 25, 2020. URL: https://spec.torproject.org/proposals/329-traffic-splitting.html (visited on 02/28/2025).

[22] Jamie Hayes and George Danezis. "k-fingerprinting: A Robust Scalable Website Fingerprinting Technique". In: *USENIX Security Symposium*. 2016. DOI: 10.5555/3241094.3241186.

[23] Martin Heckel, Florian Adamsky, Jonas Juffinger, Fabian Rauscher, and Daniel Gruss. "Verifying DRAM Addressing in Software". In: *ESORICS*. 2025.

[24] Dominik Herrmann, Rolf Wendolsky, and Hannes Federrath. "Website Fingerprinting: Attacking Popular Privacy Enhancing Technologies with the Multinomial Naïve-Bayes Classifier". In: *CCSW*. 2009. DOI: 10.1145/1655008.1655013.

[25] Andrew Hintz. "Fingerprinting Websites Using Traffic Analysis". In: *PETS*. 2003. DOI: 10.1007/3-540-36467-6_13.

[26] *Historical yearly trends in the usage statistics of client-side programming languages for websites*. W3Techs. URL: https://w3techs.com/technologies/history_overview/client_side_language/all/y (visited on 08/25/2025).

[27] Nicholas Hopper, Eugene Y. Vasserman, and Eric Chan-TIN. "How Much Anonymity does Network Latency Leak?" In: vol. 13. 2. Mar. 5, 2010. DOI: 10.1145/1698750.1698753.

[28] Rob Jansen, John Geddes, Chris Wacek, Micah Sherr, and Paul Syverson. "Never Been KIST: Tor's Congestion Management Blossoms with Kernel-Informed Socket Transport". In: *USENIX Security Symposium*. 2014. ISBN: 978-1-931971-15-7.

[29] Rob Jansen and Nicholas Hopper. "Shadow: Running Tor in a Box for Accurate and Efficient Experimentation:" in: *NDSS*. Feb. 2012. DOI: 10.21236/ADA559181.

[30] Rob Jansen, Justin Tracey, and Ian Goldberg. "Once is Never Enough: Foundations for Sound Statistical Inference in Tor Network Experimentation". In: *USENIX Security Symposium*. 2021. ISBN: 978-1-939133-24-3.

[31] Rob Jansen, Matthew Traudt, John Geddes, Chris Wacek, Micah Sherr, and Paul Syverson. "KIST: Kernel-Informed Socket Transport for Tor". In: *ACM Transactions on Privacy and Security (TOPS)* 22.1 (2018), pp. 1–37. DOI: 10.1145/3278121.

[32] Rob Jansen, Ryan Wails, and Aaron Johnson. "Repositioning Real-World Website Fingerprinting on Tor". In: *WPES*. 2024. DOI: 10.1145/3689943.3695047.

[33] Marc Juarez, Sadia Afroz, Gunes Acar, Claudia Diaz, and Rachel Greenstadt. "A Critical Evaluation of Website Fingerprinting Attacks". In: *ACM CCS*. Nov. 3, 2014. DOI: 10.1145/2660267.2660368.

[34] Akshaya Mani, T. Wilson-Brown, Rob Jansen, Aaron Johnson, and Micah Sherr. "Understanding Tor Usage with Privacy-Preserving Measurement". In: *ACM IMC*. 2018. DOI: 10.1145/3278532.3278549.

[35] G Mani, P V V S Srinivas, Ganga Rama Koteswara Rao, Chitturi Prasad, Donepudi Priyanka, and Naresh Cherukuri. "Traffic Analysis of High Throughput Traffic on Tor". In: *I-SMAC*. Nov. 2021. DOI: 10.1109/I-SMAC52330.2021.9640934.

[36] Robin Marx. *How Browsers REALLY Load Web Pages*. Feb. 1, 2025. URL: https://fosdem.org/2025/schedule/event/fosdem-2025-4852-how-browsers-really-load-web-pages/ (visited on 02/24/2025).

[37] Prateek Mittal, Ahmed Khurshid, Joshua Juen, Matthew Caesar, and Nikita Borisov. "Stealthy traffic analysis of low-latency anonymous communication using throughput fingerprinting". In: *ACM CCS*. Oct. 17, 2011. DOI: 10.1145/2046707.2046732.

[38] Steven James Murdoch and George Danezis. "Low-cost traffic analysis of Tor". In: *IEEE S&P*. 2005. DOI: 10.1109/SP.2005.12.

[39] John Nagle. *Congestion Control in IP/TCP Internetworks*. RFC 896. RFC Editor, Jan. 1984.

[40] Sebastian Pahl, Florian Adamsky, Daniel Kaiser, and Thomas Engel. "Examining the Hydra: Simultaneously Shared Links in Tor and the Effects on Its Performance". In: *PETS*. 2023. DOI: 10.56553/popets-2023-0081.

[41] Andriy Panchenko, Fabian Lanze, Jan Pennekamp, Thomas Engel, Andreas Zinnen, Martin Henze, and Klaus Wehrle. "Website Fingerprinting at Internet Scale". In: *NDSS*. 2016. DOI: 10.14722/ndss.2016.23477.

[42] Andriy Panchenko, Lukas Niessen, Andreas Zinnen, and Thomas Engel. "Website Fingerprinting in Onion Routing Based Anonymization Networks". In: *WPES*. 2011. DOI: 10.1145/2046556.2046570.

[43] Mike Perry. *A Critique of Website Traffic Fingerprinting Attacks | Tor Project*. Nov. 7, 2013. URL: https://blog.torproject.org/critique-website-traffic-fingerprinting-attacks/ (visited on 08/26/2025).

[44] J. Postel. *Internet Protocol*. Tech. rep. 791. Sept. 1981. 51 pp. DOI: 10.17487/RFC0791. URL: https://www.rfc-editor.org/info/rfc791.

[45] Fabian Rauscher, Andreas Kogler, Jonas Juffinger, and Daniel Gruss. "IdleLeak: Exploiting Idle State Side Effects for Information Leakage". In: *NDSS*. 2024. DOI: 10.14722/ndss.2024.24078.

[46] Joel Reardon and Ian Goldberg. "Improving Tor using a TCP-over-DTLS tunnel". In: Aug. 10, 2009. ISBN: 978-1-71380-400-0.

[47] Vera Rimmer, Davy Preuveneers, Marc Juarez, Tom Van Goethem, and Wouter Joosen. "Automated Website Fingerprinting through Deep Learning". In: *NDSS*. 2018. DOI: 10.14722/ndss.2018.23105.

[48] Yi Shi and Kanta Matsuura. "Fingerprinting Attack on the Tor Anonymity System". In: *ICICS*. 2009. DOI: 10.1007/978-3-642-11145-7_33.

[49] P. Sirinam, M. Imani, M. Juarez, and M. Wright. "Deep Fingerprinting: Undermining Website Fingerprinting Defenses with Deep Learning". In: (Oct. 2018), pp. 1928–1943.

[50] Payap Sirinam, Mohsen Imani, Marc Juarez, and Matthew Wright. "Deep Fingerprinting: Undermining Website Fingerprinting Defenses with Deep Learning". In: *ACM CCS*. 2018. DOI: 10.1145/3243734.3243768.

[51] Steven Engler. *tornettools/README.md*. 2025. URL: https://github.com/shadow/tornettools/blob/main/README.md (visited on 02/27/2025).

[52] The Tor Project. *Tor Research Safety Board*. 2026. URL: https://research.torproject.org/safetyboard/ (visited on 04/14/2026).

[53] *Top Websites Ranking in the World, February 2025*. Feb. 12, 2025. URL: https://ahrefs.com/websites (visited on 02/12/2025).

[54] *Tor Specification: Circuit-level padding*. URL: https://spec.torproject.org/padding-spec/circuit-level-padding.html (visited on 02/28/2025).

[55] *Tor Specification: Connection-level padding*. URL: https://spec.torproject.org/padding-spec/connection-level-padding.html (visited on 02/28/2025).

[56] Tao Wang, Xiang Cai, Rishab Nithyanand, Rob Johnson, and Ian Goldberg. "Effective Attacks and Provable Defenses for Website Fingerprinting". In: *USENIX Security Symposium*. 2014. ISBN: 978-1-931971-15-7.

[57] Ruiyi Zhang, Taehyun Kim, Daniel Weber, and Michael Schwarz. "(M)WAIT for It: Bridging the Gap between Microarchitectural and Architectural Side Channels". In: *USENIX Security Symposium*. 2023. ISBN: 978-1-939133-37-3.

A Tor Configuration

In this section is a list of our Tor clients and relay configuration. The configuration for our middle relay is in Listing 1. The configuration for the attacker is in Listing 2 and for the victim it is Listing 3.

Listing 1: Full configuration of our middle relay.

```
Nickname        [...]
ORPort          9000
ExitRelay       0
SocksPort       0
Log notice      syslog
DataDirectory   /var/lib/tor
User            tor
ControlPort     127.0.0.1:9051
MaxAdvertisedBandwidth 38400
```

Listing 2: Full configuration of our Tor client for the RTT measurements attacker).

```
ORPort 0
ExtORPort 0
DirPort 0
ClientOnly 1
ExitRelay 0
SocksPort 127.0.0.1:19050
ControlPort 127.0.0.1:19051

ConnectionPadding 0
CircuitPadding 0
NewCircuitPeriod 0
UseEntryGuard 0
```

Listing 3: Full configuration of our Tor client that visits websites victim).

```
ORPort 0
ExtORPort 0
DirPort 0
ClientOnly 1
ExitRelay 0
SocksPort 127.0.0.1:19050
ControlPort 127.0.0.1:19051

NewCircuitPeriod 0
```

B CNN architecture

Table 6 shows the architecture of our CNN.

Layer	Filter/ Units	Kernel/ Pool Size	Strides	Activation
Conv2D	32	5, 5	2, 2	relu
Conv2D	32	5, 5	1, 1	relu
Conv2D	32	5, 5	1, 1	relu
MaxPolling2D		2, 2	2, 2	
Dropout				
Conv2D	64	3, 3	1, 1	relu
Conv2D	64	3, 3	1, 1	relu
Conv2D	64	3, 3	1, 1	relu
MaxPolling2D		2, 2	2, 2	
Dropout				
Conv2D	128	3, 3	1, 1	relu
Conv2D	128	3, 3	1, 1	relu
Conv2D	128	3, 3	1, 1	relu
MaxPolling2D		2, 2	2, 2	
Dropout				
Flatten				
Dropout				
Dense	1024			relu
Dropout				
Dense	512			relu
Dropout				
Dense	50			softmax

Table 6

C Websites for Fingerprinting

Table 7 lists 150 websites that we use for our experiments and indicates whether they can be loaded via exit relay A, B, or C.

Table 7: Overview of websites during Tor Browser experiments. A check mark indicates successful load through the respective exit relay.

Website	A	B	C	Website	A	B	C	Website	A	B	C
1mg.com				gsmarena.com		✓	✓	reddit.com	✓	✓	✓
365scores.com	✓	✓	✓	healthline.com	✓	✓	✓	roblox.com	✓	✓	✓
accuweather.com				hindustantimes.com	✓	✓	✓	rottentomatoes.com			
adobe.com				homedepot.com				sahibinden.com			
aiscore.com	✓	✓	✓	hotstar.com				samsung.com			
aliexpress.com				hurriyet.com.tr		✓		sattamatkadpboss.co	✓	✓	✓
allegro.pl				ikea.com				savefrom.net	✓	✓	✓
allrecipes.com				ilovepdf.com	✓	✓	✓	scribd.com	✓	✓	✓
alodokter.com				imdb.com	✓	✓	✓	shopee.co.id	✓	✓	✓
amazon.co.jp				indeed.com				shutterstock.com		✓	✓
amazon.co.uk				indianexpress.com	✓	✓	✓	sikayetvar.com			
amazon.com		✓	✓	indiatimes.com	✓	✓	✓	skysports.com	✓	✓	✓
amazon.de				infobae.com	✓	✓	✓	smallpdf.com	✓	✓	✓
amazon.fr				instagram.com	✓	✓	✓	speedtest.net		✓	✓
amazon.in				interia.pl	✓	✓	✓	spotify.com		✓	✓
amazon.it				investing.com	✓	✓	✓	ssstik.io	✓	✓	✓
apple.com	✓	✓	✓	iplt20.com	✓	✓	✓	steampowered.com	✓	✓	
bbc.co.uk	✓	✓	✓	justdial.com				t.me		✓	
bbc.com	✓	✓	✓	kompas.com				telegram.org		✓	
booking.com				kumparan.com				terra.com.br			
britannica.com	✓	✓	✓	linkedin.com	✓	✓	✓	texastribune.org	✓	✓	✓
cambridge.org	✓	✓	✓	marca.com		✓	✓	theguardian.com	✓	✓	✓
canva.com		✓	✓	mayoclinic.org				thehindu.com	✓	✓	✓
casibom-tr-adres.com				medicalnewstoday.com	✓	✓	✓	tiktok.com	✓	✓	✓
chatgpt.com	✓	✓	✓	medlineplus.gov	✓	✓	✓	tokopedia.com			
clevelandclinic.org				mercadolivre.com.ar				tradingview.com		✓	✓
cnn.com				mercadolivre.com.br				trendyol.com			
cnnbrasil.com.br	✓	✓	✓	mercadolivre.com.mx				tripadvisor.com		✓	✓
conectate.com.do	✓	✓	✓	merriam-webster.com	✓	✓	✓	twitter.com	✓	✓	✓
cookpad.com	✓	✓	✓	microsoft.com	✓	✓	✓	uefa.com	✓	✓	✓
crazygames.com	✓	✓	✓	milliyet.com.tr		✓		uol.com.br	✓	✓	✓
cricbuzz.com	✓	✓	✓	mlb.com				vk.com			
croxyproxy.com	✓	✓	✓	moneycontrol.com				walmart.com		✓	✓
deepl.com	✓	✓	✓	msdmanuals.com			✓	weather.com	✓	✓	✓
detik.com				msn.com	✓	✓	✓	webmd.com	✓	✓	✓
discord.com	✓	✓		mykhel.com				whatsapp.com	✓	✓	✓
dpboss.boston	✓	✓	✓	namu.wiki				wikipedia.org	✓	✓	✓
ebay.com				nba.com		✓	✓	wiktionary.org	✓	✓	✓
epfindia.gov.in		✓		ndtv.com	✓	✓	✓	wise.com			
espn.com	✓	✓	✓	netflix.com	✓	✓	✓	x.com	✓	✓	✓
espncricinfo.com			✓	nhs.uk	✓	✓	✓	yahoo.co.jp			
etsy.com		✓	✓	nih.gov		✓	✓	yahoo.com		✓	✓
facebook.com	✓	✓	✓	nike.com				yandex.com.tr	✓	✓	✓
fandom.com	✓	✓	✓	nintendo.com	✓	✓	✓	yandex.ru			
flipkart.com	✓	✓	✓	nytimes.com	✓	✓	✓	yelp.com			
freepik.com				openai.com				youtube.com	✓	✓	✓
github.com	✓	✓	✓	pinterest.com							
globo.com				poki.com	✓	✓	✓				
goal.com	✓	✓	✓	primevideo.com	✓	✓	✓				
google.com		✓	✓	quillbot.com	✓	✓	✓				
gov.br				quora.com							
gov.uk	✓	✓	✓	rakuten.co.jp	✓	✓	✓				