Evaluating Snowflake as an Indistinguishable Censorship Circumvention Tool

Kyle MacMillan
Princeton University

Jordan Holland
Princeton University

Prateek Mittal
Princeton University

Abstract

Tor is the most well-known tool for circumventing censorship. Unfortunately, Tor traffic has been shown to be detectable using deep-packet inspection. WebRTC is a popular web framework that enables browser-to-browser connections. Snowflake is a novel pluggable transport that leverages WebRTC to connect Tor clients to the Tor network. In theory, Snowflake was created to be indistinguishable from other WebRTC services.

In this paper, we evaluate the indistinguishability of Snowflake. We collect over 6,500 DTLS handshakes from Snowflake, Facebook Messenger, Google Hangouts, and Discord WebRTC connections and show that Snowflake is identifiable among these applications with 100% accuracy. We show that several features, including the extensions offered and the number of packets in the handshake, distinguish Snowflake among other WebRTC-based services. Finally, we suggest recommendations for improving identification resistance in Snowflake.

1 Introduction

Authoritarian governments continue to employ a myriad of technical mechanisms to detect and suppress internet activity [1]. Censors can trivially deny access to specific websites by blocking IP addresses or impeding DNS resolution. These techniques are only successful when the censor can accurately detect the activity it wishes to suppress. Pluggable transports transform Tor traffic into seemingly benign traffic to disguise user activity [2]. We look to evaluate a new pluggable transport, Snowflake.

Snowflake overview. Snowflake is composed of three core components: (1) the client, a user in a censored region, (2) the Snowflake broker, a server that connects clients to Snowflake proxies, and (3) a Snowflake proxy, a volunteer with an uncensored internet connection [3]. After the broker has paired the client with an available proxy, the client and proxy establish a WebRTC connection. The client can then connect to a Tor relay through the Snowflake proxy.

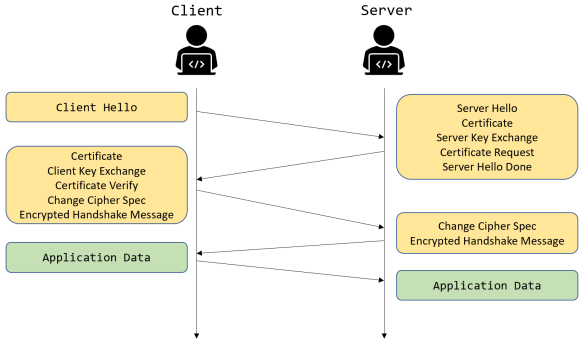

Figure 1: Messages exchanged during DTLS handshake

WebRTC connection. WebRTC is a web framework that supports peer-to-peer communication between browsers. The WebRTC handshake utilizes either Data Transport Layer Security (DTLS) or Stream Control Transmission Protocol (SCTP). Every application examined in this work employs DTLS. As illustrated in Figure 1,

Snowflake detection. Snowflake's success relies on the ubiquity and indistinguishability of WebRTC [4]. If few applications use WebRTC, blocking all instances of WebRTC is a reasonable approach to blocking Snowflake. However, many web applications, such as Facebook Messenger and Google Hangouts, use WebRTC to facilitate browser to browser connections. As more services adopt WebRTC, the collateral consequences of a blanket WebRTC ban would outweigh the benefits of blocking Snowflake. In this case, one way the censor can identify Snowflake is by its WebRTC handshake [5].

2 Evaluating Snowflake's Indistinguishability

Threat model. Any surveillance state can easily observe its internet access points, with some bandwidth and computational limitations. We consider an adversary with access to a client's WebRTC packets, including headers, protocols, and payloads. We assume that Snowflake is resistant to detection by IP address. Given a large volume of temporary proxies

and the use of techniques such as domain fronting, a censor's ability to detect Snowflake connections via IP addresses is limited [6].

Data collection. We collect data by capturing isolated DTLS handshakes. Table 1 summarizes the handshakes collected.

	Snowflake	Facebook	Google	Discord
Firefox	991	796	1000	992
Chrome	0	784	995	997
Total	991	1580	1995	1989

Table 1: Number of handshakes collected for each services on a given browser.

Average Packets per Handshake. Immediately observable is the difference between the average number of packets sent per handshake among the services, as shown in Table 2. The Snowflake handshake tends to require several retransmissions, resulting in a much longer handshake than other services, where retransmissions are observed sparingly.

Snowflake	Facebook	Google Hangouts	Discord
13.42	4.4	4.5	5.6

Table 2: Average number of handshakes collected for each service on a given browser.

3 Classifying Handshakes

Classification Methods Table 3 summarizes the 20 features extracted. We use one-hot-encoding to transform non-numeric data into binary features. We choose a random forest classifier because it allows us to examine which features drive model performance. We use 5-fold cross validation for all evaluation metrics. We evaluate our classifier using accuracy and micro-weighted F1 scores.

Feature	Client Hello	Server Hello
Length	✓	✓
Message Sequence	✓	✓
Fragment Offset	✓	✓
DTLS Version	✓	✓
SID Length	✓	✓
Cookie Length	✓	
Cipher Suite Length	✓	
Cipher Suites	✓	
Extension Length	✓	✓
Extension	✓	✓
Cipher Chosen		✓

Table 3: Features extracted from WebRTC handshakes.

Classification evaluation. One-hot-encoding the features in Table 3 produces 61 total features. We train a classifier using scikit learn's Random Forest Classifier module [7]. The model has an average accuracy of 100% across all classes. The classifier has a micro-weighted F1 score of 1.0. Given these results, we search for *identifiers*: features whose values are unique to each class.

Analyzing feature importance. We leverage the model's feature importances to search for Snowflake identifiers. Table 4 shows features unique to Snowflake. *supported_groups* and *renegotiation_info* are extensions offered in the Server Hello. *Server Message Sequence: "1"* indicates that the Snowflake DTLS protocol includes optional *Client Hello* and *Hello Verify Request* packets that the other services omit.

	Application			
Feature	**SF**	**FB**	**G**	**D**
Server Message Sequence: 1	100	0	0	0
renegotiation_info	0	100	100	100
supported_groups	100	0	0	0

Table 4: Percentage of handshakes that contained a given feature for Snowflake (SF), Facebook Messenger (FB), Google Hangouts (G), and Discord (D).

4 Recommendations

Based on our results, it is necessary to modify the Snowflake WebRTC implementation to resist detection by content. The following modifications are short-term fixes that can improve Snowflake's indistingiushability:

- Do not send the optional *Client Hello* and *Hello Verify Request* from the DTLS handshake
- Offer 'renegotiation_info' as an extension in the server hello
- Do not offer 'supported_groups' as an extension in the server hello

However, modifying Snowflake's WebRTC approach to mimic popular services may be futile [8]. As a long-term solution, we suggest that Snowflake use an existing WebRTC-based service's implementation [9].

References

[1] Joseph Lorenzo Hall, Michael D. Aaron, Stan Adams, Amelia Andersdotter, Ben Jones, and Nick Feamster. A Survey of Worldwide Censorship Techniques. Internet-Draft draft-irtf-pearg-censorship-03, Internet Engineering Task Force, May 2020. Work in Progress.

[2] Paul Syverson, Roger Dingledine, and Nick Mathewson. Tor: The second generation onion router. In *Usenix Security*, pages 303–320, 2004.

[3] David Fifield. *Threat modeling and circumvention of Internet censorship*. PhD thesis, UC Berkeley, 2017.

[4] David Fifield and Mia Gil Epner. Fingerprintability of WebRTC. *arXiv preprint arXiv:1605.08805*, 2016.

[5] Liang Wang, Kevin P Dyer, Aditya Akella, Thomas Ristenpart, and Thomas Shrimpton. Seeing through network-protocol obfuscation. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pages 57–69, 2015.

[6] David Fifield, Chang Lan, Rod Hynes, Percy Wegmann, and Vern Paxson. Blocking-resistant communication through domain fronting. *Proceedings on Privacy Enhancing Technologies*, 2015(2):46–64, 2015.

[7] Diogo Barradas, Nuno Santos, and Luís Rodrigues. Effective detection of multimedia protocol tunneling using machine learning. In *27th USENIX Security Symposium (USENIX Security 18)*, pages 169–185, 2018.

[8] Amir Houmansadr, Chad Brubaker, and Vitaly Shmatikov. The parrot is dead: Observing unobservable network communications. In *2013 IEEE Symposium on Security and Privacy*, pages 65–79. IEEE, 2013.

[9] Amir Houmansadr, Thomas J Riedl, Nikita Borisov, and Andrew C Singer. I want my voice to be heard: IP over voice-over-IP for unobservable censorship circumvention. In *NDSS*, 2013.