# SNOWFLAKE AUDIT
# -
# SNAPSHOT

CAROLIN ZÖBELEIN

*This paper is dedicated to all the brave snowflakes who die every year during winter.*

ABSTRACT. STATUS: DRAFT

## CONTENTS



## PREAMBLE

The following document is for discussion purposses only. It gives no warranty for completeness and correctness.

## 1. INTRODUCTION

*Snowflake* is a pluggable transport, which uses WebRTC to proxy traffic through emporary proxies. It aims to work kind of like flash proxy [1] [2]. This document is a short snapshot audit of the current state of the existing snowflake code [3] on `https://gitweb.torproject.org/pluggable-transports/snowflake.git/`.

## 2. BASIC PACKAGE STRUCTURE

At first, let's have a first look at the basic package structure of Snowflake [3].

---

*Date*: Last change: December 22, 2018, Status: Draft.

The author believes in the importance of the independence of research and is funded by the public community. If you also believe in this values, you can find ways for supporting the author's work here: `https://research.carolin-zoebelein.de/crowdfunding.html`.

2.1. **General package information.** *Snowflake* is a *Go*-based Pluggable Transport using WebRTC, inspired by Flashproxy.

2.2. **Package tree.** Given are the following main parts of the package:

- **appengine (d):** Runs an Google App Engine and reflects domain-fronted requests from a client to the Snowflake broker.
- **broker (d):** Handles the rendezvous by matching Snowflake clients with proxies. It passes the client's WebRTC session descriptions. So the clients and proxies can establish a peer connection.
- **client (d):** Tor client component of Snowflake.
- **proxy (d):** Browser proxy component of Snowflake.
- **proxy-go (d):** Standalone version of the Snowflake proxy.
- **server (d):** Server transport plugin for Snowflake. The client connects to the proxy using WebRTC and the proxy connects to the server using WebSocket.
- **server-webrtc (d):** WebRTC server plugin which uses an HTTP server that simulates the interaction that a client would have with the broker, for direct testing.
- **CONTRIBUTING.md (f), LICENSE (f), README.md (f):** Standard package files.

d= directory, f=file.

2.3. **Interaction structure.** In figure 1 you can see the interaction structure between the different parts of Snowflake.

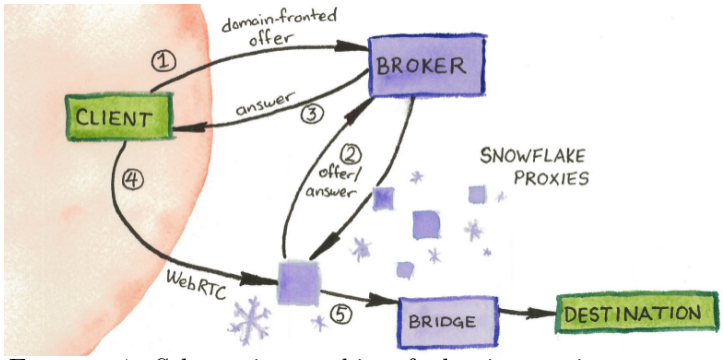


FIGURE 1. Schematic graphic of the interaction structure of Snowflake. Taken from [4].

1a. The client makes a request to the broker.
1b. The appengine reflects the domain-fronted request from the client to the broker.
2a. The broker matches the client with proxies.
2b. The broker passes the client's WebRTC session descriptions.
3. The broker gives an answer about the matching proxy to the client.
4. The client uses WebRTC for the connection to the given proxy.
5. Finally, the client connects to a bridge over WebRTC.

## 3. Client

In the following sections we will do an raw audit of the Snowflake client.

## 4. Conclusion

## References

[1] Serene: *[tor-dev] Introducing Snowflake (webrtc pt)*. https://lists.torproject.org/pipermail/tor-dev/2016-January/010310.html. Version: 2016
[2] *Snowflake*. https://trac.torproject.org/projects/tor/wiki/doc/Snowflake. Version: 2018
[3] *Snowflake*. https://gitweb.torproject.org/pluggable-transports/snowflake.git/. Version: 2018/12/17
[4] *cypherpunks*. https://trac.torproject.org/projects/tor/attachment/wiki/doc/Snowflake/snowflake-schematic.png. Version: 2018

## License



Carolin Zöbelein, Independent mathematical scientist, Josephsplatz 8, 90403 Nürnberg, Germany, https://research.carolin-zoebelein.de

*E-mail address*: contact@carolin-zoebelein.de, PGP: D4A7 35E8 D47F 801F 2CF6 2BA7 927A FD3C DE47 E13B