Scary: A really scary Pluggable Transport

... or the magic of obscuration for Tor.*

Carolin Zöbelein†
Independent mathematical scientist
Josephsplatz 8, 90403 Nürnberg, Germany

ABSTRACT

STATUS: Draft

KEYWORDS

Tor, Bridge, Scary, Obscuration, Censorship, Circumvention, Pluggable Transport

PREAMBLE

This paper was written in the context of a job application as Pluggable Transport Software Developer for Anti-Censorship Team of The Tor Project[1].

1 INTRODUCTION

In August 2018, The Intercept published a story about plans of Google for launching a censored version of its search engine in China, which will blacklist websites and search terms about human rights, democracy, religion, and peaceful protest [6]. This project, with the code-name *Dragonfly*, started in spring prior year, is the newest step in the ongoing work of creating a censored environment of information in China.

If we look back, the story of censorship in China started in 1998. The Communist Party feared that the China Democracy Party would create a powerful new network. The China Democracy Party was immediately banned, members arrested and imprisonment [2]. Finally, this resulted in the beginning of the *Great Firewall (GFW)* project, a combination of legislative actions and technologies enforced by the People's Republic of China to regulate the Internet domestically. It blocks access to selected websites, internet tools, mobile apps and slows down cross-border internet traffic.

Since the GFW blocks destinations and inspects the data being transmitted, ways for censorship circumvention need proxy nodes and encrypted data traffic. Typically, this is done these days by the help of foreign proxy servers, regional website mirrors, Tor, virtual private networks (VPNs) and secure shell (SSH).

Over the years, more and more of this circumvention tools have been blocked due to deep packet inspections and the detailed analysis of its content. So now, many VPNs are no longer useable to circumvent the Great Firewall of China and also the access to the Tor anonymity network [14], with its public list of relays, is no longer possible.

To solve the problem of relay blocking, Tor introduced so-called *bridges* [16] which are non-public relays, to help censored users reach the Tor network. Because of the ability of dynamically blocking bridges by looking for their TLS fingerprint [18] [1], packet fragmentation and Tor obfsproxy in combination with private bridges, were added [18].

Finally, this lead us to *Pluggable Transports (PT)* [17], which help to bypass censorship attempts against Tor. PTs transform the Tor traffic between client and bridges, in such a way that it looks like innocent traffic instead of the actual Tor traffic. In this paper, we will talk about this PTs, their general construction constraints and an introducing of an sketch of a new PT called *Scary*.

1.1 Outline

TODO

1.2 Notation

TODO

2 TLS FINGERPRINTING

If we look at the previous story of censorship and blocking, this leads us to modern cryptographic protocols to provide communications security in computer networks. *Transport Layer Security (TLS)* [9] is the one we have to look at here in the context of Tor traffic fingerprinting and its examination.

2.1 A brief history of TLS

In 1999, the history of Transport Layer Security started when TLS v1.0 was introduced in RFC 2246 [7] as an upgrade of SSL v3.0. Seven years later, in 2006, TLS v1.1 was definied in RFC 4346 [8] as an update of TLS v1.0 with significant changes, like e.g. protection against cipher-block chaining (CBC) attacks. Two years later, in 2008 TLS 1.2 was published in RFC 5246 [9], which its major differences, like e.g. replacing of MD5-SHA-1 with SHA-256 in the pseudorandom function (PRF) and in the finished message hash. Finally, just in August 2018, the newest version TLS 1.3 was definied in RFC 8446 [10], with changes like e.g. separating key agreement and authentication algorithms from the cipher suites, removing support for MD5 and SHA-224 hash functions and much more.

In this papper, we will always assume that we are talking about TLS 1.2, since it was the state-of-the-art, when the PT *Scary*, which will be described in the following, was developed, in 2015.

---

*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

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

[1]The Tor Project, Inc., is a 501(c)(3) nonprofit organization advancing human rights and freedoms by creating and deploying free and open source anonymity and privacy technologies. [15]

October 9, 2018.

2.2 The TLS Handshake Protocol

The TLS Handshake Protocol ([9] Sections 7.3 & 7.4), which operates on the top of the TLS record layer, is the first interesting part for us. The TLS client and server agree on a protocol version, select cryptographic algorithms, optionally authenticate each other, and generate shared secrets.

We will look at the hello messages, which are used to exchange security enhancement capabilities between client and server, at first.

2.2.1 TLS Client Hello. During the beginning of the connection between a client and a server, the client sends the *ClientHello* as its first message, at all. It consists of a random structure with gmt_unix_time, the current time and date in standard UNIX 32-bit format, and random_bytes, 28 bytes which are generated by a secure random number gnerator. In addition it includes a variable-length session identifier, so a SessionId becomes valid when the handshake negotiation is completed.

Furthermore, the *ClientHello* consists of the cipher suite list, which contains the combinations of cryptographic algorithms supported by the client in order of the client's preference. The cipher suites define key exchange algorithms, encryption algorithms, a MAC algorithm, and a PRF. The server will choose a cipher suite uint8 CipherSuite from this list.

The *ClientHello* also includes a list of supported compression algorithms, in order according to the client's preferences and optional extensions.

In listing 1 you can see a summary of the full *ClientHello* structure.

Listing 1: *ClientHello* structure [9].

```
struct {
    ProtocolVersion client_version;
    Random random;
    SessionID session_id;
    CipherSuite cipher_suites<2..2^16-2>;
    CompressionMethod compression_methods<1..2^8-1>;
    select (extensions_present) {
        case false:
            struct {};
        case true:
            Extension extensions<0..2^16-1>;
    };
} ClientHello;
```

After sending this message, the Client waits for a *ServerHello* response.

2.2.2 TLS Server Hello. The *ServerHello* is send by the server as response to the *ClientHello* message from the client, when it found an acceptable set of algorithms.

It is very similar to the *ClientHello*, consists of a server_version, the suggested one by the client, a random structure, independently generated by the server from the ClientHello.random, the session_id, the single cipher suite selected by the server from the list in ClientHello.cipher_suite, the single compression algorithm selected by the server from the list in ClientHello.compression_methods and optional extensions.

In listing 2 you can see a summary of the full *ServerHello* structure.

Listing 2: *ServerHello* structure [9].

```
struct {
    ProtocolVersion server_version;
    Random random;
    SessionID session_id;
    CipherSuite cipher_suite;
    CompressionMethod compression_method;
    select (extensions_present) {
        case false:
            struct {};
        case true:
            Extension extensions<0..2^16-1>;
    };
} ServerHello;
```

2.3 TODO

TODO

3 TOR'S PLUGGABLE TRANSPORTS

To bypass censorship attempts against Tor, *Pluggable Transports (PT)* were developed [17], to fake Tor traffic in such a way that it looks like ordinary traffic. We want to give a short summary [12] of Tor's most important pluggable transports [13].

3.1 Ordinary Tor traffic

In the section above, we already looked at the ordinary TLS Handshake protocol and its *ClientHello* and *ServerHello* message. The interesting parts from this, for us, are the cipher suite list, server name and TLS extensions, which are send by the client and responded by the server.

Tor used a distinctive cipher suite list, until China started to block Tor [18] [11], in 2011. In answer to this, Tor changed its cipher suite list to be the same as Firefox's, but Tor had still a different TLS extension like Firefox because of e.g. other Elliptic curves sets [12]. Finally, the server responds with its choosen parameters and a random server name, too.

3.2 The obfs family

The first pluggable transports originated from the obfs, "look-like-nothing", family and consists of *obfs2* [4], *obfs3* [5] and *obfs4* [3].

4 CONCLUSIONS

A APPENDIX

A.1 Definitions

- **Virtual private network (VPN).** TODO
- **Secure shell (SSH).** TODO
- **Bridges.** TODO
- **Pluggable Transport (PT).** TODO

REFERENCES

[1] Roya Ensafi, Philipp Winter, Abdullah Mueen, and Jedidiah R. Crandall. 2015. Analyzing the Great Firewall of China Over Space and Time. *PoPETs* 2015 (2015), 61–76.

[2] https://en.wikipedia.org/wiki/Great_Firewall. 2018. Great Firewall. (2018).

[3] https://gitweb.torproject.org/pluggable-transports/obfs4.git/tree/doc/obfs4-spec.txt. 2018. Obfs4 specification. (2018).

[4] https://gitweb.torproject.org/pluggable-transports/obfsproxy.git/tree/doc/obfs2/obfs2-protocol-spec.txt. 2015. Obfs2 specification. (2015).

[5] https://gitweb.torproject.org/pluggable-transports/obfsproxy.git/tree/doc/obfs3/obfs3-protocol-spec.txt. 2015. Obfs3 specification. (2015).

[6] https://theintercept.com/2018/08/01/google-china-search-engine-censorship/. 2018. Google plans to launch censored search engine in China, Leaked documents reveal. (2018).
[7] https://tools.ietf.org/html/rfc2246. 1999. The TLS Protocol, Version 1.0. (1999).
[8] https://tools.ietf.org/html/rfc4346. 2006. The Transport Layer Security (TLS) Protocol, Version 1.1. (2006).
[9] https://tools.ietf.org/html/rfc5246. 2008. The Transport Layer Security (TLS) Protocol, Version 1.2. (2008).
[10] https://tools.ietf.org/html/rfc8446. 2018. The Transport Layer Security (TLS) Protocol, Version 1.3. (2018).
[11] https://trac.torproject.org/projects/tor/ticket/4744. 2011. Ticket #4744: GFW probes based on Tor's SSL cipher list. (2011).
[12] https://trac.torproject.org/projects/tor/wiki/doc/AChildsGardenOfPluggableTransports. 2016. A Childs Garden Of Pluggable Transports. (2016).
[13] https://trac.torproject.org/projects/tor/wiki/doc/PluggableTransports/list. 2017. List of Pluggable Transports. (2017).
[14] https://www.torproject.org/. 2018. The Tor Project. (2018).
[15] https://www.torproject.org/about/jobs-developer-anti-censorship.html.en. 2018. Internet Freedom Nonprofit Seeks Software Developer for Anti-Censorship Team. (2018).
[16] https://www.torproject.org/docs/bridges.html.en. 2018. Tor: Bridges. (2018).
[17] https://www.torproject.org/docs/pluggable-transports.html.en. 2018. Tor: Pluggable Transports. (2018).
[18] Philipp Winter and Stefan Lindskog. 2012. How the Great Firewall of China is blocking Tor. In *Proceedings of the USENIX Workshop on Free and Open Communications on the Internet (FOCI 2012)*.

LICENSE