Pluggable Transports Roadmap

Steven J. Murdoch George Kadianakis

2012-03-17

Abstract

Of the currently available pluggable transports, obfs2 is a good default due to its efficiency and resistance to blocking. StegoTorus offers alternative pluggable transports which are significantly harder to block, but come with a much higher overhead. Suggestions for future development include: efficiency improvements, splitting and joining Tor cells to disguise Tor's distinctive packet-size probability distribution, efficient ways to hide ciphertext in compressed data, hardening obfs2 against a passive adversary, and designing a HTTP transport capable of traversing a proxy server.

1 Introduction

This paper discusses the currently available pluggable transports for Tor, and their various advantages and disadvantages. It concludes with suggestions for further development.

1.1 Dust

Dust [3] defines a packet format in which there are no identifiable patterns in the payload. Therefore, it is hoped, Deep Packet Inspection (DPI) equipment should be unable to reliably block Dust packets, or identify Dust hosts. In addition to the packet format, Dust defines a public-key key-exchange protocol which is resistant to a passive or active adversary, but does not provide perfect-forward-secrecy. Part of the key exchange is performed offline, and the remainder is online.

1.1.1 Strengths

Dust assumes a reasonably strong threat model, in that the adversary may be active. Dust also goes to some lengths to ensure that there are no static fingerprints in the payload packets, and permits padding to be added so as to disguise packet length. Dust is versatile, and can be carried both by TCP and UDP.

1.1.2 Weaknesses

Currently Dust does not provide reliable in-order delivery, so cannot be directly used as a Tor pluggable transport. Deployment is also made more difficult by having key-exchange partially offline. Packets which are entirely random do stand out, but it is not clear that DPI equipment can perform such entropy tests.

1.2 obfs2

obfs2 [1] is currently the only protocol supported by the obsfproxy framework. In a similar way to Dust, it aims to have no static protocol fingerprint, but it is simpler. obfs2 depends on a reliable in-order transport (typically TCP) and offers a reliable in-order transport to layers above (i.e. Tor).

1.2.1 Strengths

obfs2 is relatively simple and has no static protocol signature to block. It is already implemented as a pluggable transport and has been tested in the field. obfs2 has a low overhead (just a key-exchange) and uses only symmetric cryptography so is fast.

1.2.2 Weaknesses

obfs2 is not resistant to either an active or passive adversary, as someone who has recorded the initial key exchange can recover the session key and decode the traffic. obfs2 also makes no attempt to hide packet lengths. Like Dust, obfs2's high-entropy output format does make it possible to distinguish from most other protocols.

1.3 StegoTorus: Packet Chopping

StegoTorus defines a series of transports, all based on the packet chopper, but the packet chopper can be used by itself. Like Dust and obfs2, packets are indistinguishable from random. By using public key cryptography, StegoTorus is resistant to a passive adversary, but the lack of key-exchange authentication means that it is vulnerable to a man-in-the-middle attack. The key exchange also does not provide perfect-forward-secrecy.

The chopper provides a reliable in-order transport, and so is suitable to be used as a Tor pluggable transport. The chopper depends on an underlying reliable transport, but it does not have to be in-order. The packet format permits padding to be added.

1.3.1 Strengths

The packet chopper has been implemented and tested as a pluggable transport, and is more resilient than obfs2 to passive attacks. The packet chopper is also more versatile in that it does not need an in-order transport.

1.3.2 Weaknesses

The packet chopper is more complex than obfs2 and has more overhead. Like obfs2 and Dust, the high entropy packets could conceivably be detected.

1.4 StegoTorus: Embed Module

The embed module is built on the packet chopper. It takes a network trace and substitutes the output of the chopper into the payload of the trace. Since traces are to be collected from encrypted communication, an adversary should not be able to tell that the encrypted payload has been replaced with another encrypted payload.

1.4.1 Strengths

Unlike the previous transports, the embed module does not have an unusually high entropy of payload. Provided the packet traces are selected appropriately it should be very hard to distinguish this transport's use.

1.4.2 Weaknesses

Before this transport can be used, a sufficiently large number packet traces have to be distributed to clients that they cannot all simply be blocked. If the operator of the DPI system has a better understanding of the protocol being impersonated than the designer of the embedder, it might be possible to develop blocking mechanisms. The embed module can only impersonate encrypted protocols, and has a moderately high overhead due to padding.

1.5 StegoTorus: HTTP Module

An alternative to the embed module is HTTP. This also requires network traces, but is no longer restricted to only encrypted protocols. Instead, Javascript, PDF and SWF files are used for server to client communication (client to server communication uses the cookie header).

1.5.1 Strengths

Here, even blocking all encrypted protocols will not be sufficient to block the HTTP transport. As such, it is one of the most blocking-resistant protocols developed.

1.5.2 Weaknesses

Unfortunately, the overhead of the HTTP module is very high and as a result performance is low. Currently, HTTP protocol compliance is not sufficiently complete to permit traffic being processed through a HTTP proxy.

1.6 bananaphone

bananaphone [2] is an encoding scheme which transforms a stream of binary data into a stream of tokens (eg, something resembling natural language text) such that the stream can be decoded by concatenating the hashes of the tokens.

1.6.1 Strengths

The strength of bananaphone is that its output resembles arbitrarily formatted data. For example, its output can look like a natural language, or the chunks of an MP3 file.

Furthermore, bananaphone can also be used to generate payloads for other pluggable transports (e.g. HTTP or IRC transports).

1.6.2 Weaknesses

bananphone is not able to produce truly realistic looking output. That is, bananaphone output emulating a natural language will look wrong upon human inspection.

Standalone bnanaphone is not able to withstand adversaries willing to whitelist specific network protocols.

1.7 Easy-Come-Easy-Go Transports (ECEGT)

Another deployment idea is to create a number of easy-to-implement but easy-to-block pluggable transports. Pluggable transport proxies are designed to be modular and to allow fast development of pluggable transports, while most DPI systems are not currently designed with the same modular mindset.

1.7.1 Strengths

The main strength of this idea is that by carefully selecting the strengths of each ECEGT, censors might be forced to build distinguishers for each one of them.

For example, a simple obfs2-over-base64 transport would reduce the entropy of obfs2 and also obfuscate the packet sizes a bit. This will probably force censors to add more rules to their DPI systems, which not only takes time but also increases their false-positive rates.

1.7.2 Weaknesses

The obvious weakness of ECEGT is that censors can easily find ways to block them.

2 Protocol trade-offs

The discussion above has shown some trade-offs which must be made when designing a protocol.

2.0.3 Perfect forward secrecy

To achieve perfect forward secrecy, a Diffie-Hellman key exchange needs to be performed. This requires public key cryptography, and also one more round-trip than otherwise necessary. While desirable, Tor already provides perfect forward secrecy, so it is not essential that the pluggable transport does so.

2.0.4 Resistance to passive and active attacks

A goal of any blocking resistance scheme is resistance to being reliably fingerprinted by a passive attack. However, some (such as obfs2) only try to avoid identification by string matching rather than an arbitrary passive attack. To reliably resist passive attacks, public key cryptography can be used.

Resisting active attacks is harder. Unless there is out-of-band exchange of an authentication key, a man-in-the-middle can always impersonate both sides of the communication. However, while resisting active attacks is desirable, the underlying Tor protocol already provides resistance.

2.0.5 Entropy fingerprinting

Tor traffic is encrypted and therefore inevitably is high-entropy. Any transport which attempts to reduce the entropy of data must therefore significantly increase the payload size. This restricts efficient blocking resistant protocols to impersonating encrypted protocols, or perhaps compressed data (on the assumption that DPI equipment will not decode the compressed data).

2.0.6 Padding and splitting

Even if traffic payload is disguised, directly encoding Tor packets will lead to a characteristic set of packet sizes. By basing network traffic on observed network traces, StegoTorus resists blocking based on packet sizes but at the cost of a high padding overhead.

3 Potential censorship techniques

It is difficult to predict the approach which will be taken by censors, especially as some are willing to block all encrypted traffic, at least temporarily. Some potential techniques (e.g. entropy detection, protocol whitelisting, etc.) are implicit in the development of fingerprinting resistant transports.

However one approach which could work, even against the StegoTorus HTTP module, is to require all data be sent via a HTTP proxy (which itself blocks encrypted traffic). This is done in some high-security corporate environments so it is likely that commercially available filtering equipment is capable of enforcing this policy.

4 Further avenues of research

In case HTTP proxy filtering is deployed, it would be useful to have a pluggable transport which was compatible enough with HTTP to make it through unscathed.

Further efficiency improvements could be made to all protocols. Some possibilities are simple (e.g. compression of Tor data, taking advantage of the predictable TLS headers), but more sophisticated approaches could lead to better results.

obfs2 could be extended to resist passive attack by performing a public-key handshake.

A better understanding of how to reliably embed encrypted data in compressed fields would make it more difficult to develop filtering equipment capable for distinguishing common compression algorithms from ciphertext.

Padding to disguise packet size is expensive, but simply joining or splitting packets would disguise Tor's protocol signature at little or no cost. This approach is not capable of matching an arbitrary packet-size probability distribution, but it would defeat simple packet-size fingerprinting attacks.

5 Changes to Tor

There is no transport protocol proposed which is both efficient and very likely to remain unblocked, so the decision to move obfuscation out of Tor still seems appropriate.

References

[1] George Kadianakis and Nick Mathewson. obfs2 (the twobfuscator). Technical report. `https://gitweb.torproject.org/obfsproxy.git/blob/HEAD:/doc/obfs2/protocol-spec.txt`.

[2] Leif Ryge. bananaphone: Reverse hash encoding. Technical report. `https://github.com/leif/bananaphone`.

[3] Brandon Wiley. Dust: A blocking-resistant internet transport protocol. Technical report. `http://blanu.net/Dust.pdf`.