Dust: A Blocking-Resistant Internet Transport Protocol

Brandon Wiley
School of Information, University of Texas at Austin

Abstract

Censorship of free speech on the Internet has been an increasing problem as the methods of identifying and filtering traffic have become more sophisticated. A number of approaches to counteract Internet censorship have been implemented, from censorship-resistant publishing systems to anonymizing proxies. While existing systems provide protection against some attacks, they fail to provide resistance to blocking of the transport protocol by modern techniques such as Deep Packet Inspection. Dust is proposed as a blocking-resistant Internet protocol designed to be used in conjunction with existing systems to add resistance to a number of attacks currently in active use to censor Internet communication.

1. Introduction

The evolution of Internet censorship has been a cycle of increasingly sophisticated filtering techniques inspiring new circumvention techniques. Shallow Packet Inspection filtered primarily based on matching the IP addresses in packet headers to a blacklist of known IP addresses. This attack led to the development of anonymizing proxy networks that hide the true destination IP address by first routing through proxies with unknown, and therefore not blacklisted, IPs. Filtering technology now employs Deep Packet Inspection (DPI) techniques that can filter out specific Internet protocols by detecting certain characteristic properties. This has resulted in censorship-resistant services such as anonymizing proxies being specifically targeted to be blocked or throttled. In order to provide censorship resistance, blocking resistance is now necessary to defeat DPI filtering.

Dust is an Internet protocol designed to provide blocking resistance against DPI techniques. Dust uses a novel out-of-band handshake to establish a secure, blocking-resistant channel for communication over a filtered channel. Once a secure channel has been established, Dust packets are indistinguishable from random packets and so cannot be filtered by normal techniques. For attackers that specifically filter random packets an optional "unrandomization" step is used to give the packets arbitrary statistical properties.

2. Related Work

While a network of proxy nodes can provide protection against destination IP blacklists, they are still vulnerable to various forms of DPI protocol fingerprinting. This problem is dealt with by Kopsell, who proposes a method to extend existing anonymous publishing systems to bypass blocking, a property referred to as "blocking resistance" [10]. Kopsell's threat model assumes that the attacker has control of only part of the Internet and that some small amount of unblockable inbound information can enter, perhaps out of band. The nodes in Kopsell's system are volunteer anonymizing proxies that clients communicate with over in order to obtain access to a censorship-resistant publishing system. Clients obtain an invitation to the network, including the IP addresses of some proxy nodes, through a low-bandwidth, unblockable channel.

Kopsell's proposal used SSL as the communication channel. Unfortunately, SSL does not offer blocking resistance when SSL traffic is specifically targeted. Tor has suffered blocking by two attacks because of its use of SSL. One attack specifically targeted unique characteristics of Tor's SSL handshake and the other was a general throttling of all SSL traffic [2][9]. While Tor has subsequently increased the steganographic strength of its SSL handshake by making it resemble Apache's SSL handshake, this protects against only the first attack and not the second.

2.1. Obfuscated Protocols

An obfuscated protocol, in contrast to a secure protocol, provides protection from the attacker only so long as the attacker does not know the details of the encoding. For instance, BitTorrent clients have implemented three obfuscating protocols in order to prevent filtering and throttling of the BitTorrent protocol, the most common of which is Message Stream Encryption (MSE) [7]. Analysis of packet sizes and the direction of packet flow have been shown to identify connections obfuscated with MSE with 96% accuracy, primarily through analysis of the statistical properties of the key exchange [4].

Obfuscated TCP (ObsTCP) has gone through several versions, the last of which used DNS records to transmit the encryption keys [8]. This required the attacker to correlate separate communication streams, extracting the keys from the DNS packets and then applying them to the TCP packets. However, an analogous attack has already been demonstrated in the blocking of BitTorrent traffic through monitoring of the tracker protocol traffic to obtain the ports of the BitTorrent protocol connections [10][7]. A similar proposal called tcpcrypt is also easily defeated by looking for static strings in the handshake [1].

Obfuscated-openssh obfuscates the handshake for an existing secure protocol by replacing the handshake portion of an SSH protocol connection with a minimal blocking-resistant encrypted protocol [6]. This handshake is encrypted with a key that is generated by iterated hashes of a seed that is added to the beginning of the encrypted part of the handshake. The iteration number is chosen to be high enough that key generation is slow, so the blocking resistance of this technique relies on key generation being too expensive to scale to all connections simultaneously. However, modern filters are capable of statistically sampling packets and processing them offline, allowing for the obfusation to be defeated at least probabilistically. [11].

3. Design

Unlike an obfuscated or steganographic protocol, Dust is secure even against attackers that know the full details of the protocol. The difficulty is in negotiating a secure session key without an unencrypted, filtered handshake. Fortunately, Kopsell's model allows for a single out-of-band invitation to be sent prior to the establishment of the data connection. This affordance is the key to creating a protocol which is not merely obfuscated, but rather secure even if the attacker monitors all other traffic and is aware of the design of the protocol.

The Dust protocol implements such a design in order to provide censorship resistance through protocol unobservability. Dust uses out-of-band invitations to perform the first part of the cryptographic handshake. In order to establish protocol unobservability, all packets consist entirely of encrypted or random, single-use bytes so as to be indistinguishable from each other and from random packets. Theoretical attackers that target random packets are dealt with by an optional second stage of "unrandomization" using a reverse Huffman encoder. Table 1 summarizes the various attacks that Dust is designed to defend against.

As in Kopsell's model, a peer must first receive an out-of-band invitation to join the network. This invitation contains the IP address and public key of the receiver. The sender can then complete the handshake by sending a single in-band intro packet. The handshake is now complete and each side now has the public key of the other, allowing for a secure session key to be computed. The parties can now communicate bidirectionally with any number of data packets encrypted with the session key that was computed in the handshake. The minimal Dust conversation therefore consists of two logical in-band packets: one intro packet, and one data packet. The protocol specification allows for these packets to be chained together inside a single UDP or TCP packet. The use of a single UDP or TCP packet for communication of short messages prevents timing attacks when the payload is sufficiently small. For larger payloads, Dust does not protect against timing attacks based on the timing of the application layer protocol. However, Dust does not add its own unique timing characteristics in the handshake as the handshake consists of only one packet and therefore is not vulnerable to timing attacks except those directed at the timing added by applications using Dust as the underlying transport protocol. It is the responsibility of the application protocol to control its timing in such a way that it is difficult to fingerprint.

3.1. Protocol

In order to accept a connection from a new client, a Dust server must first complete a key exchange with that client. The Dust server first creates an invite packet containing the server's IP, port, and public key, and a newly generated ID and shared secret pair. The invite must then be communicated to the client out-of-band by any means that is effective. The invite is encrypted with a password and so is indistinguishable from random bytes. It can therefore be safely transmitted, along with the password, over an out-of-band channel such as email or instant messaging. It will not be susceptible to attacks which block email communication containing IP addresses because only the password is transmitted unencrypted. If the invitation channel is under observation by the attacker, and only in the case that the attacker is specifically attempting to filter Dust packets, then the password should be sent by another channel that, while it can still be observed by the attacker, should be uncorrelated with the invitation channel. For instance, the password could be delivered in a non-digital form such as voice or writing.

An important thing to note about the out-of-band channel for distribution of the invite packet is that anonymizing proxy networks already require such a channel for the purpose of proxy discovery. It is already necessary for communicating the proxy IP and port to the client without this communication being blocked by the attacker. Modifying the proxy discovery protocol to use Dust invite packets instead of plain-text IP and port pairs is a minimal change and allows for a truly secure protocol instead of mere obfuscation. The format for Dust invites is also more resistant to filtering than the invitations currently in use for anonymizing the proxies. The worst case for Dust invites is the one in which both the invite and the password are intercepted, they are correlated, and the invite is decrypted. The IP of the receiver and then by added to a blacklist and subsequently blocked. This worst cast scenario is equivalent to the current situation for anonymizing proxies which use unencrypted invitations.

In order to complete the handshake, the client uses the IP and port information from the invite packet to send an intro packet to the server. The first few bytes of the intro packet contain the random, single-use ID from the invite packet. The rest of the intro packet is encrypted with the secret from the invite packet. The payload of the intro packet is the public key of the client.

When the server receives a packet from an unknown IP address, it assumes it to be an intro packet and retrieves the ID from the beginning of the packet. This is used to look up the associated stored secret. The server uses the secret to decrypt the packet, retrieves the public key of the client, and generates a shared session key. It adds the session key to its list of known hosts, associated with the IP and port from which the intro packet was sent. This completes the second phase of the public key exchange. The client and server can now send and receive encrypted data packets freely. Since Dust packets can be chained inside of TCP or UDP packets, the intro packet may be followed immediately by a data packet. If the message to be communicated is short, then this single packet may constitute the entirety of the conversation.

3.2. Packet Format

There are three types of Dust packets: invite, intro, and data. All three types of packets build upon the basic Dust packet format. In a basic Dust packet, the MAC is computed using the ciphertext, initialization vector (IV), and a key. The type of key used differs depending on the type of packet. Using a MAC allows for the contents of the packet to be verified against corruption or tampering. The IV is a single-use random value used to encrypt the ciphertext and compute the MAC. This ensures that the ciphertext and MAC values will be different even when sending the same data. Since the IV is random and the MAC is computed using the IV, both values are effectively random to an observer. The rest of the packet, excluding the padding, are encrypted to form the ciphertext. The ciphertext includes a timestamp, lengths for the data and padding, and the data itself. A separate padding length (PL) value is needed because several Dust packets may be contained inside a single UDP or TCP packet. Finally, a random number of random bytes of padding are added in order to randomize the packet length.

An invite packet contains all of the basic fields such as MAC, IV, and padding. The key used in an invite packet to encrypt the ciphertext and compute the MAC is a derived from a password and random salt value using a password-based key derivation function (PBKDF). The salt value is prepended to the encrypted packet. The use of both salt and a PBKDF makes it difficult to decrypt the packet by brute force. This protects the contents of the invite packet against decryption unless the password is known.

The invite packet includes the information necessary for the client to connect to the server and complete the handshake. It contains the server's public key, the IP and port where the server can be contacted, a flags byte which specifies if the connection to the server should be made using UDP or TCP and whether the IP address is an IPv4 or IPv6 address, and an ID and secret pair to be used in the construction of an intro packet.

In an intro packet, the ID is the same as the one received in the invite packet. This is effectively a single-use random value as when it was contained in the invite packet it was encrypted and it is only seen in plain text in the intro packet. The ID is used by the server to link the intro packet to the stored single-use random secret. This secret is used to encrypt the ciphertext and to compute the MAC for the intro packet. Since each ID is a single-use value, only one intro packet can be sent for each invite packet received by the client. The rest of the fields in an intro packet are the same as a general Dust packet. The content of an intro packet is the public key of the client.

Once the server has obtained the client's public key from the intro packet, the key exchange is complete and a shared session key is computed by both sides for use in encrypting future data packets. In a data packet, the content is the data to be sent and the key used to encrypt the ciphertext and to compute the MAC is the shared session key derived from the exchanged public key and locally stored private key.

4. Discussion

Table 1 summarizes the attacks against the protocol and the protocol features that protect against those attacks. The common DPI attacks are covered as well as common attacks against general protocols such as corruption and replay of packets. The main theoretical attack that has been raised against Dust so far is that of statistical sampling and filtering of protocols with high entropy. Since Dust cannot be differentiated by normal means from other entropy maximizing protocols the attack must be against all such protocols.

To defend against this attack, recent versions of Dust include an optional "unrandomization". The high entropy output of the normal Dust encoder is given to a reverse Huffman encoder [12], which uses the frequency table from an existing protocol and "decompresses" the Dust packet to match that distribution. Through this process a Dust packet can be given an arbitrary entropy, thereby defeating high entropy filtering.

Attack	Defense
Static strings	All fields encrypted or randomized
Length	Randomized padding
Timing	Single packet conversations
Corruption	Encrypted MAC
Replay	Timestamp
Insertion	Encrypted MAC
Brute force	Salt and PBKDF
High entropy	Reverse Huffman Encoding

Table 1. Attacks and defenses

5. Conclusion

Dust allows for secure communication to be established without leaking information to the attacker that could be used to identify and block the protocol. For attackers that specifically seek to block Dust-like protocols, an optional second stage of steganographic encoding can be used, allowing Dust to resemble existing protocols (when an appropriate encoder is available) while maintaining its core properties of security and lack of information flow to the attacker.

Those wishing to examine or use the Dust protocol for academic or practical purposes can find the source code for its implementation at http://github.com/blanu/Dust. Although the general Dust protocol can use a variety of algorithms for its cryptography operations, the reference Dust implementation uses Skein-256-256 for hash, MAC, PRNG, PBKDF, and cipher functions and the curve25519 ECDH implementation for generate session keys from public and private keys. Further specific details about the implementation of the protocol can be found in the README at https://github.com/blanu/Dust/raw/master/README.

6. Acknowledgements

This work was made possible by the generous intellectual contributions of Drake Wilson, Greg Hazel, and Luis Francisco-Revilla, as well as a grant from the University of Texas at Austin iSchool CDLF program and the IMLS.

7. References

1. Bittau, A., Hamburg, M., Handley, M., Mazieres, D., and Boneh, D. The case for ubiquitous transport-level encryption. 19th USENIX Security Symposium., (2008).

2. Dingledine, R. Tor and circumvention: Lessons learned. The 26th Chaos Communication Congress, (2009).

3. Harrison, D. BEP 008: Tracker Peer Obfuscation. Retrieved from: http://www.bittorrent.org/beps/bep_0008.html.

4. Hjelmvik, E and John, W. Breaking and Improving Protocol Obfuscation. Department of Computer Science and Engineering, Chalmers University of Technology, Technical Report No. 2010-05, ISSN 1652- 926X. (2010)

5. Kopsell, S., Hilling, U.: How to Achieve Blocking Resistance for Existing Systems Enabling Anonymous Web Surfing. In: Proceedings of the Workshop on Privacy in the Electronic Society. pp. 103-115. ACM Press, New York (2004)

6. Leidl, B. Obfuscated-OpenSSH README. Retrieved from: https://github.com/brl/obfuscated-openssh/blob/master/README.obfuscation. (2010)

7. Message Stream Encryption. http://wiki.vuze.com/w/Message_Stream_Encryption (2006)

8. Obfuscated TCP. Wikipedia. Retrieved from: http://en.wikipedia.org/wiki/Obfuscated_TCP. (2010)

9. Sennhauser, M.: The State of Iranian Communication. http://diode.mbrez.com/docs/SoIN.pdf (2009)

10. Topolsky, R. Comments of Robert M. Topolsky In the Matter of Petition of Free Press et al. for Declaratory Ruling that Degrading an Internet Application Violates the FCC's Internet Policy Statement and Does Not Meet an Exception for "Reasonable Network Management". Federal Communications Commission WC Docket No. 07-52, 08-7. (2008)

11. Using NetFlow Filtering or Sampling to Select the Network Traffic to Track. Retrieved from: http://www.cisco.com/en/US/docs/ios/netflow/configuration/guide/nflow_filt_samp_traff.html#wp1064305. (2006)

12. Wayner, P. Basic Mimicry. In *Disappearing Cryptography: Information Hiding: Steganography & Watermarking*. Morgan Kaufmann, 2008, 87-92.