

BIP: 156
Layer: Peer Services
Title: Dandelion - Privacy Enhancing Routing
Author: Brad Denby <bdenby@cmu.edu>
Andrew Miller <soc1024@illinois.edu>
Giulia Fanti <gfanti@andrew.cmu.edu>
Surya Bakshi <sbakshi3@illinois.edu>
Shaileshh Bojja Venkatakrisnan <shaileshh.bv@gmail.com>
Pramod Viswanath <pramodv@illinois.edu>
Comments-URI: <https://github.com/bitcoin/bips/wiki/Comments:BIP-0156>
Status: Rejected
Type: Standards Track
Created: 2017-06-09
License: CC0-1.0

Abstract

Bitcoin's transaction spreading protocol is vulnerable to deanonymization attacks. Dandelion is a transaction routing mechanism that provides formal anonymity guarantees against these attacks. When a node generates a transaction without Dandelion, it transmits that transaction to its peers with independent, exponential delays. This approach, known as diffusion in academia, allows network adversaries to link transactions to IP addresses.

Dandelion mitigates this class of attacks by sending transactions over a randomly selected path before diffusion. Transactions travel along this path during the "stem phase" and are then diffused during the "fluff phase" (hence Dandelion). We have shown that this routing protocol provides near-optimal anonymity guarantees among schemes that do not introduce additional encryption mechanisms.

Motivation

Transaction diffusion in Bitcoin is vulnerable to deanonymization attacks. Because transactions are sent to peers with independent, exponential delays, messages spread through the network in a statistically symmetric manner. This pattern allows colluding spy nodes to infer the transaction source. Breaking this symmetry prevents the attack. However, we have shown that an adversary with knowledge of the network topology can launch a much more effective "fingerprint" attack if the symmetry breaking is not done properly.

Consider a botnet-style adversary with access to the P2P graph. Botnets of size comparable to the Bitcoin P2P network are common and cheap, and these adversaries can learn the network structure with probe messages. We have shown that such an adversary can achieve total deanonymization of the entire network after observing less than ten transactions per node.

Dandelion is a practical, lightweight privacy solution that provides the Bitcoin

network formal anonymity guarantees. While other privacy solutions aim to protect individual users, Dandelion protects anonymity by limiting the capability of adversaries to deanonymize the entire network.

How Dandelion Works

Dandelion enhances user privacy by sending transactions through an anonymity phase before diffusing them throughout the network. At a high level, Dandelion enhances privacy by (i) breaking the symmetry of diffusion and (ii) mixing transactions by forwarding messages from different sources along the same path.

Dandelion routing can be conceptualized in three phases. First, a privacy graph is constructed. In practice, this privacy graph is constructed in a fully decentralized manner and is a subgraph of the existing Bitcoin P2P network. Next, transactions are forwarded along this privacy graph during the "stem phase." Finally, messages are broadcast to the network during the "fluff phase" using the typical method of diffusion.

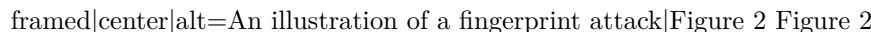
Figure 1

In order to select the privacy graph in a decentralized manner, each node selects a subset of its outbound peers to be Dandelion destinations. Dandelion transactions (transactions in their stem phase) that arrive at this node via inbound connections are forwarded to these Dandelion destinations.

In an ideal setting, we have found that a Hamiltonian circuit provides near-optimal privacy guarantees. However, constructing a Hamiltonian circuit through the Bitcoin P2P network in a decentralized, trustless manner is not feasible. Thus, we recommend that each node select two Dandelion destinations uniformly at random without replacement from its list of outbound peers. Our tests have shown that this method provides comparable privacy with increased robustness.

During stem phase routing, there is a question of how to route messages in order to protect privacy. For example, if two Dandelion transactions arrive at a node from different inbound peers, to which Dandelion destination(s) should these transactions be sent? We have found that some choices are much better than others.

Consider the case in which each Dandelion transaction is forwarded to a Dandelion destination selected uniformly at random. This approach results in a fingerprint attack allowing network-level botnet adversaries to achieve total deanonymization of the P2P network after observing less than ten transactions per node.

Figure 2

During a fingerprint attack, a botnet-style adversary with knowledge of the graph structure first simulates transaction propagation. This offline step lets the adversary generate fingerprints for each network node. During the online attack, the adversary collects transactions at its spy nodes and matches these

observations to the simulated fingerprints. Our simulations have shown that this attack results in devastating, network-wide deanonymization.

framed|center|alt=A plot illustrating total deanonymization|Figure 3 Figure 3

To avoid this issue, we suggest "per-inbound-edge" routing. Each inbound peer is assigned a particular Dandelion destination. Each Dandelion transaction that arrives via this peer is forwarded to the same Dandelion destination. Per-inbound-edge routing breaks the described attack by blocking an adversary's ability to construct useful fingerprints. Fingerprints arise when routing decisions are made independently per transaction at each node. In this case, two transactions from the same node generally take different paths through the network. Crucially, this results in multiple, unique data points that are aggregated to match with a fingerprint.

Dandelion ensures that two transactions from the same node take the same network path, limiting adversaries to the far-left of the graph in Figure 3. In other words, adversary knowledge is limited to the case of one observed message rather than a rich profile of multiple transaction paths. Dandelion also breaks the symmetry of diffusion, making the source of the transaction difficult to infer.

framed|center|alt=A plot illustrating limited deanonymization|Figure 4 Figure 4

After a transaction has traveled along a Dandelion stem for a random number of hops, it transitions into the fluff phase of routing. The transaction is shared with the network through the existing process of diffusion. In practice, this fluff mechanism is enforced by a weighted coin flip at each node. If the random value is below some threshold, the Dandelion transaction is transformed into a typical transaction. In our testing, we have chosen a probability of ten percent that a given Dandelion transaction enters fluff phase when leaving a given node. This value strikes a good balance between stem path length and transaction spreading latency.

Note that Dandelion's expected precision guarantees are a population-level metric, whereas the expected recall guarantees can be interpreted as an individual-level metric. Expected recall is equivalent to the probability that an adversary associates a single transaction with a given source. These guarantees are probabilistic. They do not address scenarios in which a node has been eclipsed by other nodes, or when a node is specifically targeted by an ISP-like adversary. Individuals who are concerned about targeted deanonymization should still use Tor.

At a high level, Dandelion is like an "anonymity inoculation" for the public at large - including users who are not aware of Bitcoin's privacy issues. Higher adoption leads to greater benefits, even for users who do not use Tor. Early adopters of Dandelion still receive privacy benefits. In the worst case when no neighbors support Dandelion, transactions make at least one hop before diffusing. Note that any solution based only on routing cannot be perfectly anonymous due to the fundamental lower bounds on precision and recall shown in the original Dandelion paper. Dandelion provides near-optimal anonymity guarantees among

such solutions.

Specification

Dandelion can be specified with a handful of features: Dandelion transaction support, Dandelion routing data and logic, periodic Dandelion route shuffling, memory pool logic, the fluff mechanism, transaction embargoes, and Dandelion transaction logic. Specification details are summarized below.

Dandelion transaction support

During the stem phase, transactions are "Dandelion transactions." When a Dandelion transaction enters fluff phase, it becomes a typical Bitcoin transaction. Dandelion transactions and typical transactions differ only in their `NetMsgType`.

Dandelion (stem phase) transactions MUST be differentiable from typical Bitcoin transactions.

Dandelion routing data and logic

Dandelion routing during the stem phase requires notions of inbound peers, outbound peers, Dandelion destinations, and Dandelion routes. Inbound peers consist of all currently connected peers that initiated the peer connection. Outbound peers consist of all currently connected peers that were connected to by this node. Dandelion destinations are a subset of outbound peers. The number of Dandelion destinations is limited by the `DANDELION_MAX_DESTINATIONS` parameter. In the reference implementation, this parameter is set to two. Our tests have shown that this value provides both privacy and robustness (see the reference paper for more details on the parameter tradeoffs). Dandelion routes are a map of inbound peers to Dandelion destinations. Every inbound peer is mapped to a Dandelion destination.

Note that a Dandelion node may choose a different `DANDELION_MAX_DESTINATIONS` parameter without splitting from the privacy graph. When mapping inbound connections to outbound connections for Dandelion routes, we implement the following routing logic. First, select a set of Dandelion destinations from the set of outbound peers. This set of Dandelion destinations is of size less than or equal to `DANDELION_MAX_DESTINATIONS`. For each inbound connection, first identify the subset of Dandelion destinations with the least number of routes. For example, some subset of Dandelion destinations may be affiliated with zero routes while all other Dandelion destinations are affiliated with one or more routes. From this subset, select one Dandelion destination uniformly at random. Establish a Dandelion route from the inbound connection to this Dandelion destination.

For a given Dandelion routing epoch, two distinct Dandelion destinations SHOULD be selected uniformly at random from the set of outbound connections. All Dandelion transactions that arrive via a given inbound connection

MUST be transmitted to the same Dandelion destination. When choosing a Dandelion destination for a given inbound connection, the destination MUST be selected uniformly at random from the set of Dandelion destinations with the least number of inbound connections mapped to them.

Periodic Dandelion route shuffling

The map of Dandelion routes is cleared and reconstructed every ten minutes on average. We have chosen the value of ten minutes heuristically in order to make privacy graph learning difficult for adversaries. Note that a Dandelion node may choose a different average shuffle time without splitting from the privacy graph.

Dandelion routes MUST be cleared and reconstructed at random intervals. Dandelion routes SHOULD be cleared and reconstructed every ten minutes on average.

Memory pool logic

Dandelion transactions are segregated from typical transactions. The `mempool` remains unchanged. Another instance of the `CTxMemPool` class, called the `stempool`, is used for Dandelion transactions. Information flows from `mempool` to `stempool` in order to ensure proper transaction propagation. Information does not flow from `stempool` to `mempool`, except when a Dandelion transaction fluffs into a typical transaction.

When a Dandelion transaction arrives, the transaction MUST be added to the `stempool` and MUST NOT be added to the `mempool`. When a typical Bitcoin transaction arrives, the transaction MUST be added to the `mempool` and MUST be added to the `stempool`. When a Dandelion transaction fluffs, the transaction MUST be added to the `mempool`.

The fluff mechanism

When relaying a Dandelion transaction along a Dandelion route, there is a 10% chance that the Dandelion transaction becomes a typical Bitcoin transaction and is therefore relayed via diffusion. In our testing, this value strikes a good balance between stem path length and transaction spreading latency. Note that a Dandelion node may choose a different chance of fluffing without splitting from the privacy graph.

When a node prepares to transmit a Dandelion transaction, the node MUST flip a biased coin. If the outcome is "Dandelion transaction," then the node MUST transmit the transaction to the appropriate Dandelion destination. Otherwise, the node MUST convert the Dandelion transaction into a typical Bitcoin transaction. A Dandelion transaction SHOULD fluff into a typical Bitcoin transaction with a 10% probability.

Transaction embargoes

During the stem phase, transactions are relayed along a single path. If any node in this path were to receive the Dandelion transaction and go offline, then the transaction would cease to propagate. To increase robustness, every node that forwards a Dandelion transaction initializes a timer at the time of reception. If the Dandelion transaction does not appear in the memory pool by the time the timer expires, then the transaction enters fluff phase and is forwarded via diffusion.

When a Dandelion transaction arrives, the node **MUST** set an embargo timer for a random time in the future. If the Dandelion transaction arrives as a typical Bitcoin transaction, the node **MUST** cancel the timer. If the timer expires before the Dandelion transaction is observed as a typical Bitcoin transaction, then the node **MUST** fluff the Dandelion transaction.

Dandelion transaction logic

The following cases define a node's behavior when receiving network packets referencing Dandelion transactions.

- Receive INV for Dandelion TX: If the peer is inbound and the Dandelion transaction has not been received from this peer, then reply with GETDATA.
- Receive GETDATA for Dandelion TX: If the peer is not inbound and the Dandelion transaction has been advertised to this peer, then reply with the Dandelion transaction.
- Receive Dandelion TX: If the peer is inbound, then relay the Dandelion TX to the appropriate Dandelion destination.

Implementation

A reference implementation is available at the following URL: <https://github.com/dandelion-org/bitcoin/tree/dandelion-feature-commits>

All features have been compressed into a single commit at the following URL: <https://github.com/dandelion-org/bitcoin/tree/dandelion>

Compatibility

Dandelion does not conflict with existing versions of Bitcoin. A Bitcoin node that supports Dandelion appears no differently to Bitcoin nodes running older software versions. Bitcoin nodes that support Dandelion can identify feature support through a probe message. Obviously, older nodes are not capable of Dandelion routing. If a Bitcoin node supporting Dandelion has no peers that also support Dandelion, then its behavior naturally decays to that of a Bitcoin node without Dandelion support due to the Dandelion transaction embargoes.

Acknowledgements

We would like to thank the Bitcoin Core developers and Gregory Maxwell in particular for their insightful comments, which helped to inform this implementation and some of the follow-up work we conducted. We would also like to thank the Mumblewimble development community for coining the term "stempool," which we happily adopted for this implementation.

References

1. An Analysis of Anonymity in Bitcoin Using P2P Network Traffic http://fc14.ifca.ai/papers/fc14_submission_71.pdf
2. Deanonymisation of clients in Bitcoin P2P network <https://arxiv.org/abs/1405.7418>
3. Discovering Bitcoin's Public Topology and Influential Nodes <https://cs.umd.edu/projects/coinscope/coinscope.pdf>
4. (Sigmetrics 2017) Dandelion: Redesigning the Bitcoin Network for Anonymity <https://arxiv.org/abs/1701.04439>
5. (Sigmetrics 2018) Dandelion++: Lightweight Cryptocurrency Networking with Formal Anonymity Guarantees <https://arxiv.org/pdf/1805.11060.pdf>

Copyright

To the extent possible under law, the author(s) have dedicated all copyright and related and neighboring rights to this work to the public domain worldwide. This work is distributed without any warranty.

You should have received a copy of the CC0 Public Domain Dedication with this work. If not, see <https://creativecommons.org/publicdomain/zero/1.0/>.