Security Warnings

Security Audit

Zcash has been subjected to a formal third-party security review. For security announcements, audit results and other general security information, see https://z.cash/support/security.html

x86-64 Linux Only

There are known bugs which make proving keys generated on 64-bit systems unusable on 32-bit and big-endian systems. It's unclear if a warning will be issued in this case, or if the proving system will be silently compromised.

Wallet Encryption

Wallet encryption is disabled, for several reasons:

- Encrypted wallets are unable to correctly detect shielded spends (due to the nature of unlinkability of JoinSplits) and can incorrectly show larger available shielded balances until the next time the wallet is unlocked. This problem was not limited to failing to recognize the spend; it was possible for the shown balance to increase by the amount of change from a spend, without deducting the spent amount.
- While encrypted wallets prevent spending of funds, they do not maintain the shielding properties of JoinSplits (due to the need to detect spends). That is, someone with access to an encrypted wallet.dat has full visibility of your entire transaction graph (other than newly-detected spends, which suffer from the earlier issue).

- We were concerned about the resistance of the algorithm used to deri wallet encryption keys (inherited from Bitcoin) to dictionary attacks by a powerful attacker. If and when we re-enable wallet encryption, it is likely to be with a modern passphrase-based key derivation algorithm designed for greater resistance to dictionary attack, such as Argon2i.

You should use full-disk encryption (or encryption of your home directory) to protect your wallet at rest, and should assume (even unprivileged) users who are running on your OS can read your wallet.dat file.

Side-Channel Attacks

This implementation of Zcash is not resistant to side-channel attacks. You should assume (even unprivileged) users who are running on the hardware, or who are physically near the hardware, that your `zcashd` process is running on will be able to:

- Determine the values of your secret spending keys, as well as which notes you are spending, by observing cache side-channels as you perform a JoinSplit operation. This is due to probable side-channel leakage in the libsnark proving machinery.
- Determine which notes you own by observing cache side-channel information leakage from the incremental witnesses as they are updated with new notes.
- Determine which notes you own by observing the trial decryption process of each note ciphertext on the blockchain.

You should ensure no other users have the ability to execute code (even unprivileged) on the hardware your `zcashd` process runs on until these vulnerabilities are fully analyzed and fixed.

REST Interface

The REST interface is a feature inherited from upstream Bitcoin. By default, it is disabled. We do not recommend you enable it until it has undergone a security review.

RPC Interface

Users should refrain from changing the default setting that only allows RPC connections from localhost. Allowing connections from remote hosts would enable a MITM to execute arbitrary RPC commands, which could lead to compromise of the account running zcashd and loss of funds. For multi-user services that use one or more zcashd instances on the backend, the parameters passed in by users should be controlled to prevent confused-deputy attacks which could spend from any keys held by that zcashd.

Block Chain Reorganization: Major Differences

Users should be aware of new behavior in Zcash that differs significantly from Bitcoin: in the case of a block chain reorganization, Bitcoin's coinbase maturity rule helps to ensure that any reorganization shorter than the maturity interval will not invalidate any of the rolled-back transactions. Zcash keeps Bitcoin's 100-block maturity interval for generation transactions, but because JoinSplits must be anchored within a block, this provides more limited protection against transactions becoming invalidated. In the case of a block chain reorganization for Zcash, all JoinSplits which were anchored within the reorganization interval and any transactions that depend on them will become invalid, rolling back transactions and reverting funds to the original owner. The transaction rebroadcast mechanism inherited from Bitcoin will not successfully rebroadcast transactions depending on invalidated JoinSplits if the anchor needs to change. The creator of an invalidated JoinSplit, as well as the creators of all transactions dependent on it, must rebroadcast the transactions themselves.

Receivers of funds from a JoinSplit can mitigate the risk of relying on funds received from transactions that may be rolled back by using a higher minconf (minimum number of confirmations).

Logging z_* RPC calls

The option `-debug=zrpc` covers logging of the `z_*` calls. This will reveal information about private notes which you might prefer not to disclose. For example, when calling *z_sendmany* to create a shielded transaction, input notes are consumed and new output notes are created.

The option `-debug=zrpcunsafe` covers logging of sensitive information in `z_*` calls which you would only need for debugging and audit purposes. For example, if you want to examine the memo field of a note being spent.

Private spending keys for z addresses are never logged.

Potentially-Missing Required Modifications

In addition to potential mistakes in code we added to Bitcoin Core, and potential mistakes in our modifications to Bitcoin Core, it is also possible that there were potential changes we were supposed to make to Bitcoin Core but didn't, either because we didn't even consider making those changes, or we ran out of time. We have brainstormed and documented a variety of such possibilities in issue #826 , and believe that we have changed or done everything that was necessary for the 1.0.0 launch. Users may want to review this list themselves.