Zwallet Cursory Security Review

Taylor Hornby, December 9th, 2021

This is a short (~1 day) security review of Zwallet for the purpose of performing basic due diligence checks prior to listing it on the z.cash website. Most of the security-critical code is in the zcash-sync dependency project which is included in the scope of this review.

(Note that there is a different project called Zwallet, which is not the subject of this review.)

Since this work is not even close to a complete security review of Zwallet, and since Zwallet implements its own scanning logic and other cryptographic components, I strongly encourage the Zcash community to fund a proper review by cryptographic engineers. Notably, this audit *would not have found any of the following problems, which should be checked in a future audit*:

- Out of date or vulnerable dependencies.
- Subtle security bugs in the majority of the source code.
- Network privacy problems.
- Usable security (UX) problems.
- Differences between the code on GitHub and the binary available on Google Play.
- Correct use of ECC's rust library APIs.
- Platform-specific problems, e.g. correct use of the phone's key storage enclaves.
- Conformance with ZIP standards (e.g. UAs, Payment URIs)
- Memory corruption problems in the FFI bindings.
- Bugs in cryptographic components like the Pedersen hash implementation.
- Double-spend vulnerabilities in the reorg handling logic.
- (And lots more.)

This review only looked for:

- Overtly malicious code.
- Insecure random number (seed) generation.
- Bugs that could lead to use of weak secret keys (enabling theft of funds).

The warp sync specification was also included in the scope of this review.

Issue #1: Reorg-handling logic may be incomplete.

Status: Unconfirmed

When a reorg occurs, zcash-sync will detect it by noticing that the prev_hash of a future block does not match the hash of its previous block. The code for detecting that is here:

https://github.com/hhanh00/zcash-sync/blob/d2c3e8e6c69ff033e50ed20af75a49a8f3c3c3da/src/chain.rs#L67-L69

That error is caught in the zwallet app, which reacts by making a call to rewind the database height:

https://github.com/hhanh00/zwallet/blob/4887184bde7976eb85730e3de8714095269fe2ed/lib/account.dart#L408-L413

The rewinding logic in zcash-sync is simply a series of DELETE FROM statements that remove blocks, sapling_witnesses, received_notes, and transactions with heights greater than or equal to the rewind height:

https://github.com/hhanh00/zcash-sync/blob/d2c3e8e6c69ff033e50ed20af75a49a8f3c3c3da/src/db.rs#L146-L164

This is potentially problematic, as pre-reorg data may still be present in other parts of the app outside of the database, or the rewinding may be incomplete.

One example of incomplete rewinding is in the marking of notes as spent or unspent. In the scanning logic, a revealed nullifier may cause one of the notes to be marked as spent in the database:

https://github.com/hhanh00/zcash-sync/blob/d2c3e8e6c69ff033e50ed20af75a49a8f3c3c3da/src/scan.rs#L167

However, if that revealed nullifier were to be reorged away, the note should be marked as unspent as it is now spendable by the wallet. The logic to do this is missing. As a result, a user may lose funds by deleting a wallet that has unspent notes erroneously marked as spent.

I recommend implementing aggressive automated testing of the reorg handling logic, perhaps taking advantage of darksidewalletd. The reorg-handling logic is the most likely place to find double-spend bugs and therefore deserves much more review and testing.

Good Things

- Key generation happens here, using a secure random number generator: https://github.com/hhanh00/zcash-sync/blob/d2c3e8e6c69ff033e50ed20af75a49a8f3c3c3da/src/wallet.rs#L121-L125
- Secret key data is passed through the FFI layer encoded into ASCII strings, which makes it less likely for keys to be accidentally zeroed or truncated.

Conclusion

Zwallet's code seems to be generally of high quality, but automated testing is lacking. We found one issue in the reorg handling which will not affect most users, but which is a sign that there may be more problems in the handling of reorgs. Users should be cautious about trusting the app with large amounts of funds until it has received a more in-depth security review.