# COSMOS Fundraiser Audit

APRIL 12, 2017 | IN SECURITY AUDITS | BY OPENZEPPELIN SECURITY

The COSMOS team asked us to review and audit their new COSMOS Fundraiser code. We looked at their contracts and tools and now publish our results.

The audited projects are at their cosmos GitHub organization. The list of projects included in this audit are:

- Fundraiser.sol solidity contract.
- cosmos/fundraiser-lib project.
- cosmos/fundraiser project.

- cosmos/fundraiser-cli project.
- cosmos/fundraiser-server project.

Our overall impression of the projects is that code quality is good, and that most security best-practices were followed.

**Update**: The COSMOS team implemented most of our recommendations in the latest version of their projects.

Here's our assessment and recommendations, in order of importance:

## Severe

We haven't found any severe security problems with the code.

## Potential problems

### Use of send and call

Use of **send** and **call** primitives is always risky and should be analyzed in detail. One occurrence of call found in line 91 of Fundraiser.sol.

Remove unneeded `call.value()`

Using `call.value()` is potentially dangerous, and was responsible for the TheDAO hack. We couldn't find a reason to use `treasury.call.value(msg.value)()` instead of the simpler `treasury.send(msg.value)`. We recommend making this change.

For more info on this problem, see this note.

### Use safe math

There are many unchecked math operations in the code. It's always better to be safe and perform checked operations. Consider using a safe math library, or performing pre-condition checks on any math operation.

### Usage of blockchain.info API

blockchain.info has had problems in the past, and as the COSMOS team mentions on this note, their nodes don't recognize OP_RETURN outputs as standard. Consider running your own nodes of insight-api or bitcoin-abe and connecting to those, to reduce the risk of fundraiser problems because of third party problems.

### Use of static bitcoin fees

Line 10 in bitcoin.js of cosmos/fundraiser-lib defines a fee rate of 200 satoshis per byte. That's the current cheapest and fastest fee rate according to https://bitcoinfees.21.co/, but it's still not recommended to use a static fee rate. If there's network congestion during the fundraiser, transactions could be stuck. Consider integrating bitcoinfees' API to get recommended fee rate dynamically.

### Add preconditions to important functions

Some functions could use more precondition checking. This helps prevent critical bugs coming from programming or user input handling errors. For example, the function getAddress in line 20 of bitcoin.js of fundraiser-lib performs no checks on the `pub` argument. If there's a bug in the caller code and and invalid value is sent, an address will be returned, corresponding to no public key. Example:

### bitcoin.waitForPayment function callback could be called multiple times

waitForPayment function of bitcoin.js module in fundraiser-lib doesn't handle correctly requests that take longer than 6 seconds, and could call the callback multiple times. Consider adding a reentrancy guard, as shown on this sample code:

**maraoz/fundraiser-lib**
*fundraiser-lib – JS module for participating in Cosmos Fundraiser* github.com

## Warnings

### Naming suggestions

- The **totalEther** uint variable in line 43 of Fundraiser.sol is not an ether amount, but a wei amount. Consider renaming **totalEther** to **totalSupply** for better resemblance to the ERC20 Standard Token too.

### Use latest version of Solidity

Fundraiser.sol is written for an old version of solc (0.4.7). We recommend changing the solidity version pragma for the latest version (`pragma solidity ^0.4.10;`) to enforce latest compiler version to be used.

### Hardcoded ATOMS per BTC

The rate of ATOMS per BTC is fixed in line 12 of bitcoin.js of fundraiser-lib. If fundraiser lasts the full length (14 days), price could change a significant amount. Consider adding dynamic ATOMS/BTC rate based on USD/BTC rate from a public API.

### Key generation scheme

The key generation scheme proposed is not standard, consider using BIP32 and BIP44 to derive COSMOS, Bitcoin and Ethereum keys using standard paths. See how DFINITY did their key derivations for more info.

### COSMOS address recording method not recommended

As mentioned in these notes, COSMOS address recording mechanism uses pay-to-pubkey-hash outputs, which can be seen in lines 102 to 104 of bitcoin.js of fundraiser-lib. This is a very inefficient method, because it creates unspendable outputs, bloating the UTXO set forever. A better approach is to use OP_RETURN outputs, as mentioned in the notes linked above. See how DFINITY did their address recording for more info.

## Additional Information and Notes

- Good work using block heights (beginBlock and endBlock) instead of timestamps, as this protects against miner timestamp manipulation.
- Consider extracting the halting mechanism into a superclass, for better code clarity.
- Lines 80 to 82 (the fallback function implementation) are unnecessary and can be removed. Since solidity version 0.4.0, a non-payable fallback function will always throw.
- only_before_period is never used. Consider removing it.
- Admin has complete control over atomRate. Consider changing that to a predictable pre-fixed rate curve.
- randombytes module used for generating random buffers looks good.

### Update:

COSMOS team asked us to further review their latest commits. The reviewed changes are:

- https://github.com/cosmos/fundraiser/compare/042878f156e2b0764bdf92c99c9f3b4d9d7d9276...5c2cc150b32a3194b4ecb6c41c7358c8bbdbb1d3
- https://github.com/cosmos/fundraiser-lib/compare/426425dfc296060a9b87830e69e19ae8a6d444c0...354463946665802d4f248ec1c5ddb1e47642abc9
- https://github.com/cosmos/fundraiser-cli/compare/ccaeca38b22c1cb06c1ddb9f3abed358284c8beb...d75c41395043e4d3bc5499301a19828ef9060a6c

We have the following observations on those changes:

- No severe or potentially problematic changes found. Mostly positive changes and implementations of our recommendations. Good work!

- The file content/fundraiser-agreement.md seems to be a placeholder.
- Good work replacing blockchain.info with insight, as suggested.
- Good work fixing the precondition problem of getAddress.
- **pushTx** in src/bitcoin.js of fundraiser-lib could try to use insight and fallback to blockchain.info if it fails. Given that code for both options is written, adding this behavior should have low development cost.
- Current code still uses pay-to-pubkey-hash outputs to signal COSMOS address, which is a far from ideal method. Now that fundraiser-lib doesn't rely on blockchain.info, the method should be changed to OP_RETURN outputs.
- Good job implementing dynamic fees using https://bitcoinfees.21.co/api/v1/fees/recommended
- Consider returning a default fee rate if the rate received by bitcoinfees.21.co API is out of range in fetchFeeRate of src/bitcoin.js in fundraiser-lib. This reduces reliance on that 3rd party.
- Be careful with left-pad, used in src/ethereum.js in fundraiser-lib. 😉
- eth.js still uses etherscan.io API, consider switching to a self-hosted solution (such as a geth node) for less reliance on 3rd parties.
- Fix this TODO in eth.js of fundraiser-lib.
- Good job implementing the BIP44 key derivation scheme as recommended, in src/wallet of fundraiser-lib.
- Fix this TODO in fundraiser-cli.

## Conclusions

No severe security issues were found. Some changes were recommended to follow best practices and reduce potential attack surface.

**Update**: The COSMOS team implemented most of our recommendations in the latest version of their projects.

Overall, code quality is good, it's well commented, and most well-known security good practices were followed. 👍

*Note that as of the date of publishing, the above review reflects the current understanding of known security patterns as they relate to the COSMOS fundraiser. We have not reviewed the related COSMOS network project. The above should not be construed as investment advice or an offering of tokens. For general information about smart contract security, check out our thoughts here.*

**RELATED POSTS**