SuperDAO Promissory Token audit

OPENZEPPELIN SECURITY | FEBRUARY 7, 2017

Security Audits

We've been asked by the SuperDAO team to review and audit their new token code. We looked at their contracts and now publish our results.

The audited contracts are at their Superdao-Seed-Stage1 GitHub repo. The version used for this report is commit c20d6d45d911f59003122d97df701848cf597cab. The main contract files are PromissoryToken.sol and ConstitutionalDNA.sol.

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

EDIT: Most recommendations have been addressed by the team in this branch.

Severe

We have found two severe security problems with the code.

Use of passwords as function authentication mechanisms

The use of `founderHash`, `privatePhrase` and `_oneTimesharedPhrase` as an authentication mechanism does not add any security. Even non-public variables can be read by anyone from the blockchain, and anyone can check transaction data to recover parameter values. Checks using `founderHash`, `privatePhrase` and `_oneTimesharedPhrase` are useless and can thus be removed with an equivalent result.

Consider switching to `msg.sender`-based authentication as used here.

of PromissoryToken.sol.

– Always check send return value: OK.

– Consider calling send at the end of the function: OK.

– Favor pull payments over push payments: Severe problem.

If there is more than one destination in a withdrawal proposal, one of the destination addresses can prevent all others from getting the funds. If one of the destination addresses throws in the fallback function, the whole `approveWithdraw` function call will fail, causing all payments to fail. This gives control to any malicious payee to block payments to all other payees.

For more info on this problem, see this note.

Potential problems

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.

A particular attack on the PromissoryToken contract can be done by the founder when calling the function `setPrepaid`. On line 161 of PromissoryToken.sol, the check can be skipped if a big enough `_tokenAmount` is chosen by the founder.

Use latest version of Solidity

Current code is written for old versions of solc (0.4.0). With the latest storage overwriting vulnerability found on versions of solc prior to 0.4.4, there's a risk this code can be compiled with vulnerable versions. We recommend changing the solidity version pragma for the latest version (`pragma solidity ^0.4.7;`) to enforce latest compiler version to be used.

Warnings

Usage of magic constants

There are several magic constants in the contract code. Some examples are:

- https://github.com/Superdao-DAO/Superdao-Seed-Stage1/blob/c20d6d45d911f59003122d97df701848cf597cab/contracts/PromissoryToken.sol#L276
- https://github.com/Superdao-DAO/Superdao-Seed-Stage1/blob/c20d6d45d911f59003122d97df701848cf597cab/contracts/ConstitutionalDNA.sol#L233
- https://github.com/Superdao-DAO/Superdao-Seed-Stage1/blob/c20d6d45d911f59003122d97df701848cf597cab/contracts/ConstitutionalDNA.sol#L228

Use of magic constants reduces code readability and makes it harder to understand code intention. We recommend extracting magic constants into contract constants.

Bug Bounty

Formal security audits are not enough to be safe. We recommend implementing an automated contract-based bug bounty and setting a period of time where security researchers from around the globe can try to break the token's invariants. For more info on how to implement automated bug bounties with OpenZeppelin, see this guide.

Avoid name reuse

The `constructor` and `founderSwitchRequest` function in PromissoryToken.sol use the name `_founderHash`, for two different semantic meanings. Using the same name for two different things is confusing and can bring unintended behaviors. Consider changing the function parameter name to `founderPassword` to differenciate it's specific meaning.

Additional Information and Notes

- Consider changing the style of claimPrepaid to fail-first style. Instead of throwing on the else clause, negate the conditionals, throw on the if clause, and extract the rest outside of the if statement.
- `backerCheck` modifier is only used once. Consider removing the modifier and adding the check at the beginning of approveWithdraw.
- Same with foundationNotSet.

- Same thing with EarliestBackersSet, and the name shouldn't start with an uppercase letter preferrably.
- Same thing with MinimumBackersClaimed.
- The safeguard fallback function is not needed if using solidity >0.4.0, as the `payable` keyword was introduced.

Conclusions

Two severe security issues were found. These should be fixed as soon as possible. Some additional changes were recommended to follow best practices and reduce potential attack surface.

EDIT: Most recommendations have been addressed by the team in this branch.

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

Related Posts

Beefy
Zap Audit
OpenZeppelin

BRUSHFAM
OpenBrush Contracts Library Security Review
OpenZeppelin

Linea
Bridge Audit
OpenZeppelin

intermediary designed to execute users' orders through routes...

Security Audits

OpenBrush is an open-source smart contract library written in the Rust programming language and the...

Security Audits

Ethereum. It is designed to be EVM-compatible and aims to...

Security Audits