WingsDAO Token Audit

OPENZEPPELIN SECURITY | MAY 10, 2017

Security Audits

The Wings team asked us to review and audit their new Token contract code. We looked at their code and now publish our results.

The audited contract is at their contracts GitHub repo. The version used for this report is commit 1b308105a31c5b005c21fbbda3b1ff5b3fac9bae. The main contract file is Token.sol.

Code quality is good. We're very happy to audit a project using OpenZeppelin.

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

Update: The Wings team implement most recommendations in their master branch.

Severe

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

Potential problems

Be careful with types

Avoid declaring variables using *var* if possible. The type-deduction system can bring some surprises if you don't think about types very carefully. For example, for ints, it will choose the smallest type that is required to hold assigned constants. This can lead to endless loops and gas depletion. Better use the explicit type *uint* for no surprises and higher limits. Consider replacing all occurrences of `var` with a specific type.

There's a problem with using timestamps and **now** (alias for **block.timestamp**) for contract logic, based on the fact that miners can perform some manipulation. In general, it's better not to rely on timestamps for contract logic. The solutions is to use **block.number** instead, and approximate dates with expected block heights and time periods with expected block amounts.

The Token.sol contract uses timestamps in line 212. The risk of miner manipulation, though, is really low. The potential damage is also limited: miners could only slightly manipulate when each preallocation can be done. We recommend the team to consider the potential risk of this manipulation and switch to **block.number** if necessary.

For more info on this topic, see this stack exchange question.

Warnings

Usage of magic constants

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

- https://github.com/WingsDao/contracts/blob/1b308105a31c5b005c21fbbda3b1ff5b3fac9bae/contracts/Token.sol#L94
- https://github.com/WingsDao/contracts/blob/1b308105a31c5b005c21fbbda3b1ff5b3fac9bae/contracts/Token.sol#L150-L151

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

Fix: https://github.com/WingsDao/contracts/blob/master/contracts/Token.sol#L53

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 contract's invariants. For more info on how to implement automated bug bounties with OpenZeppelin, see this guide.

Avoid duplicated code

The logic in transfer, transferFrom, and approve is very similar to the provided methods in StandardToken and could be refactored to avoid repetition. There is no need to rewrite the parent's contract logic: consider using the super keyword to access StandardToken implementation of methods, like so:

```
function transfer(address _to, uint _value) whenAllocation(fals
  return super.transfer(_to, _value);
}
```

Fixed in https://github.com/WingsDao/contracts/blob/master/contracts/Token.sol#L131

Naming suggestions

- All references to premine (Preminer, PreminerAdded, PremineAllocationAdded, PremineRelease, etc.) are confusing. There is no mining of these tokens. Consider using another name like "initial assignment".
- **checkUserAllocation** and **checkPreminerAllocation** are not very good names, as they don't clearly indicate what they are supposed to do. Consider renaming it for example to** whenHasntAllocated(user)**.
- **monthlyPayment uint** field of Preminer struct is confusing. It's name implies it's a monthly payment but there's nothing enforcing the payment is only done monthly. Consider renaming the field or adding time checks.

Fixed in https://github.com/WingsDao/contracts/blob/master/contracts/Token.sol#L86

Use latest version of Solidity

Current code is written for old versions of solc (0.4.2). With the 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 ^0.4.10;`) to enforce latest compiler version to be used.

Fixed in https://github.com/WingsDao/contracts/blob/master/contracts/Token.sol#L1

- Comment on line 27 has a typo: should say “structure” instead of “scturcture”.
- Same in line 39: “perminers”.
- Same in line 202: “monthes”.
- Same in line 140: “functiona**lity**”.
- No uses of `send` and `call.value`.
- Good work using OpenZeppelin! 🙂
- Good work using safe math.
- Why is totalSupply assigned a fixed magic number?

Conclusions

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

Code quality is good. We’re very happy to audit a project using OpenZeppelin.

Update: The Wings team implement most recommendations in their master 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 Wings Token contract. We have not reviewed the related Wings project. The above should not be construed as investment advice or an offering of Wings tokens. For general information about smart contract security, check out our thoughts here.

Related Posts

Beefy

Zap Audit

BRUSHFAM

OpenBrush Contracts Library Security Review

Linea

Bridge Audit

Beefy Zap Audit

BeefyZapRouter serves as a versatile intermediary designed to execute users' orders through routes...

Security Audits

OpenBrush Contracts Library Security Review

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

Security Audits

Linea Bridge Audit

Linea is a ZK-rollup deployed on top of Ethereum. It is designed to be EVM-compatible and aims to...

Security Audits