grapevine
WORLD

SMART CONTRACTS AUDIT

by

VENTURE BOOST

SUMMARY

This document summarizes the assessment and conclusions regarding the overall quality, security and correctness of the Grapevine World smart contracts as evaluated by VentureBoost. The scope of this audit was a second analysis and documentation of the updated Grapevine World smart contracts codebase for quality, security and correctness.

The VentureBoost smart contract team have performed a second audit of the Grapevine smart contracts. Our findings are detailed below.

VentureBoost have no stake or vested interest in Grapevine and have made best efforts to provide an objective analysis.

This audit does not represent an endorsement of the reliability or effectiveness of the smart contracts but constitutes merely an assessment of its logic and implementation. In order to ensure that a secure contract is able to withstand the fast-paced and rapidly changing environment of the Ethereum network, we recommend to put in place a bug bounty program to encourage further and active analysis of the smart contract.

The statements made in this document should not be interpreted as investment or legal advise and its authors shall not be held liable for any decisions made based on them.

AUDIT GOALS AND FOCUS

Sound Architecture

This audit includes assessments of the overall architecture and design choices. Given the subjective nature of these assessments, it will be up to the Grapevine development team to address any of the issues that have been outlined and to make any changes to the smart contracts.

Smart Contract Best Practices

This audit evaluates whether the codebase follows the current established best practices for smart contract development as defined by OpenZeppelin and others.

Code Correctness

This audit evaluates whether the code performs and executes what it is intended to do.

Code Quality

This audit evaluates whether the code has been written in a way that ensures readability and maintainability.

Security

This audit aims to identity any exploitable security vulnerabilities or other potential threats to either the operator of the token sale or its participants.

Testing and Testability

This audit examines how easily the code can be tested and reviews how thoroughly tested the code is.

INTRODUCTION

Source Code

The Grapevine Token and token sale source code was made available in a git (GitHub) repository:

https://github.com/GrapevineWorld/crowdsale-contracts/tree/57394e09306588caaa05dec62807221713bbcd96

The following Solidity source code files (with SHA1 sums) were audited:

sha1sum(crowdsale-contracts-94adf99e098dbf33aa2949869ace99772e74bf5d.zip) =

https://github.com/GrapevineWorld/crowdsale-contracts.git

The source code utilizes the OpenZeppelin solidity library. The audit refers to commit 94adf99 of the library:

https://github.com/GrapevineWorld/crowdsale-contracts/commit/94adf99e098dbf33aa2949869ace99772e74bf5d

We have not audited the library code as a whole, any code for the accounts that will be recipients of the funds of the crowdsale or any code involved in subsequent allocation of tokens to users.

Contracts

The following contracts have been assessed as part of this audit:

- **FlatGrapevineToken.sol**
- **FlatTokenTimelockController.sol**
- **FlatGrapevineCrowdsale.sol**

Testing

Automated unit tests were provided for all contracts with 100% coverage.

ABOUT THE GRAPEVINE SMART CONTRACT

1. The Grapevine token is a fixed supply burnable token.

2. The Grapevine time lockup contract is used for locking participants bonuses and team tokens.

3. Team tokens are released in two rounds.

4. The Grapevine crowdsale is a refundable timed crowdsale with a defined cap.

5. The crowdsale contract is pausable.

6. The crowdsale contract has multiple bonus levels for participants.

7. The highest level of bonus is restricted.

General Notes

- The Grapevine Token Contract is structured in a manner that manages user balances and is compliant with the ERC20 token standard and interfaces.

- The token contract owner can burn tokens.

- The timelock controller contract is designed to lock team and participant bonus tokens for specific durations.

- Participants tokens are locked by the crowdsale contract and cannot be revoked by the owner.

- Team tokens are revocable which means owner can revoke them at any time.

- In case the crowdsale fails, all locked tokens will be returned to the owner.

- The Grapevine crowdsale contract is based on the standard OpenZeppelin library.

- The crowdsale will open for a specified duration. If the target of the crowdsale has not been met within the specified duration, all funds will be returned to the participants. If the target of the crowdsale is met, the funds will be transferred to the crowdsale owner.

- While the owner can pause the crowdsale, this will not give him ultimate control over the funds.
- The crowdsale allows participants to withdraw tokens if the crowdsale successfully meets its cap.
- There are different levels of bonuses for participants.
- The highest level of bonus is restricted.

TERMINOLOGY

This audit uses the following terminology:

Likelihood

How likely a bug is to be encountered or exploited in the wild, as specified by the OWASP Risk Rating Methodology:

https://www.owasp.org/index.php/OWASP_Risk_Rating_Methodology#Step_2:_Factors_for_Estimating_Likelihood

Impact

The impact a bug would have if exploited, as specified by the OWASP Risk Rating Methodology:

https://www.owasp.org/index.php/OWASP_Risk_Rating_Methodology#Step_3:_Factors_for_Estimating_Impact

Severity

How serious an issue is, derived from Likelihood and Impact, as specified by the OWASP Risk Rating Methodology:

https://www.owasp.org/index.php/OWASP_Risk_Rating_Methodology#Step_4:_Determining_the_Severity_of_the_Risk

AUDIT RESULTS AND FINDINGS

Note Issues

Invalid function visibility

- Likelihood: low
- Impact: low

Only those functions are marked as public functions that are accessed externally and internally.
In **TimeLockController.sol** all functions are public but none is being accessed by another contract internally.

We recommend to change their visibility to external rather than public.

Invalid Interface

- Likelihood: low
- Impact: low

The crowdsale contract needs the Time Lock Controller contract interface for making function calls rather than complete contract, like ERC20 Interface: **https://github.com/GrapevineWorld/crowdsale-contracts/blob/master/flatContracts/FlatGrapevineCrowdsale.sol#L136**

For optimization, we recommend to replace this contract:
https://github.com/GrapevineWorld/crowdsale-contracts/blob/master/flatContracts/FlatGrapevineCrowdsale.sol#L157
with the following code:

```
contract TimeLockControllerInterface{
        function createInvestorTokenTimeLock(
                address _beneficiary,
                uint256 _amount,
                uint256 _start,
                address _tokenHolder
        ) returns (bool);
          function activate();
          function setCrowdsaleEnded();
}
```

Similarly, we recommend to replace Burnable contract:

https://github.com/GrapevineWorld/crowdsale-contracts/blob/master/flatContracts/FlatGrapevineCrowdsale.sol#L1313
with the following:

```
contract BurnableToken{
        function burn(uint256 _value);
}
```

Low Severity Issues

None found.

Medium Severity Issues

Possible Perpetual Token Lockup

- Likelihood: medium
- Impact: low

According to the code semantics of the Time lockup controller contract:
https://github.com/GrapevineWorld/crowdsale-contracts/blob/master/flatContracts/FlatTokenTimelockController.sol#L447
in case the crowdsale does not reach its goal, it should return all tokens to the owner. However, this may not work as intended.

We recommend to move this call:
https://github.com/GrapevineWorld/crowdsale-contracts/blob/master/flatContracts/FlatGrapevineCrowdsale.sol#L1470
outside of the **if** condition to ensure that in case if failure, the owner can get the tokens back.

High Severity Issues

None found.

Critical Issues

None found.

CONTRACT ISSUE RATING

CONCLUSIONS AND RECOMMENDATIONS

The Grapevine smart contracts code is heavily based on the OpenZeppelin open source library which is best practice. We did not find any high severity or critical issues. Two note issues and one medium issue have been identified which we recommend to correct. We also recommend to remove conditional restrictions on bonus levels since this is generally not advisable and not good practice. Overall, the code is sophisticated and well commented.