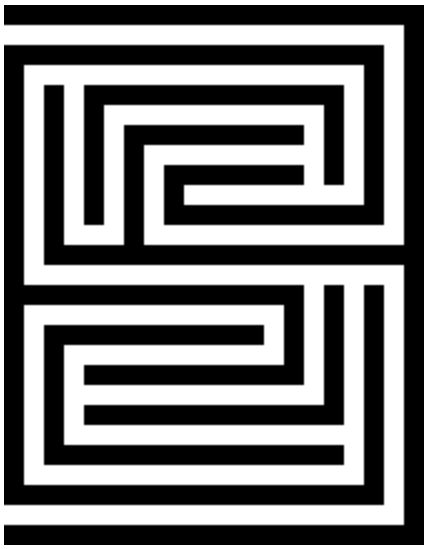

Vanilla Network Audit Public Report

PROJECT: Vanilla Network Audit

December 2020

Prepared For:

Vanilla Network Team | Vanilla Network

info@vanilla.network

Prepared By:

Jonathan Haas | Bramah Systems, LLC.

jonathan@bramah.systems

Table of Contents

Vanilla Network Protocol Security Review

Executive Summary

Scope of Engagement

Bramah Systems, LLC was engaged in December of 2020 to perform a comprehensive security review of the Vanilla Network smart contracts (specific contracts denoted within the appendix). Our review was conducted over a period of two business days by a member of the Bramah Systems, LLC. executive staff.

Bramah Systems completed the assessment using manual, static and dynamic analysis techniques.

Timeline

Review Commencement: December 20th, 2020

Report Delivery: December 22nd, 2020

Engagement Goals

The primary scope of the engagement was to evaluate and establish the overall security of the Vanilla Network protocol, with a specific focus on trading actions. In specific, the engagement sought to answer the following questions:

- Is it possible for an attacker to steal or freeze tokens?
- Does the Solidity code match the specification as provided?
- Is there a way to interfere with the contract mechanisms?
- Are the arithmetic calculations trustworthy?

Contract Specification

Specification was provided in the form of code comments and a README. The contracts were provided via GitHub, and our review focuses on the contracts of commit hash **82d0bd1701673d2fe9d5d47b016e45d4ffa25176**.

The README describes the protocol as follows:

Staking option pools yielding a fixed return. The Chocolate and Strawberry staking pools, equally are to reward long term stakeholders however "guarantees" a fixed and known return, and those wanting to increase their net ownership of the Vanilla token whilst supply continues to burn. More Details (1) There will be X number of 'seats' available for the pools. This represents the number of tokens that can be pooled into this option in totality before the pool closes. (2) There are limits applied that can be pooled per user. This will be a minimum and maximum limit. (3) A user needs to remain in this pool for a set number of days before they can claim rewards; the user has the ability to un-stake whenever they like during this period but would forfeit rewards if they were to do so. (4) There is a 5% fee for joining, and claiming rewards. This is in line with the deflationary protocol which is central to the supply slide economics of the $VNLA token. (5) To reward users for staking and remaining in this pool, a fixed equivalent annual percentage return (equivalent) will be awarded. This represents a tremendous return on investment, without damaging the projects ecosystem. (6) There will be a subscription period before the claim period for rewards commences. This should allow users sufficient opportunity to pool their tokens or aid their decision making process.

Overall Assessment

Bramah Systems was engaged to evaluate and identify any potential security concerns within the codebase of Vanilla Network. During the course of our engagement, Bramah Systems found multiple instances wherein the team deviated materially from established best practices and procedures of secure software development within DLT. In collaboration with the team, these issues have since been addressed.

These aside, the team otherwise used thoroughly reviewed and vetted components and provided details as to the token structure, economics, and intent, which helped Bramah highlight any potential concerns with their approach.

Disclaimer

As of the date of publication, the information provided in this report reflects the presently held, commercially reasonable understanding of Bramah Systems, LLC.'s knowledge of security patterns as they relate to the Vanilla Network Protocol, with the understanding that distributed ledger technologies ("DLT") remain under frequent and continual development, and resultantly carry with them unknown technical risks and flaws. The scope of the review provided herein is limited solely to items denoted within "Scope of Engagement" and contained within "Directory Structure". The report does NOT cover, review, or opine upon security considerations unique to the Solidity compiler, tools used in the development of the protocol, or distributed ledger technologies themselves, or to any other matters not specifically covered in this report.

The contents of this report must NOT be construed as investment advice or advice of any other kind. This report does NOT have any bearing upon the potential economics of the Vanilla Network protocol or any other relevant product, service or asset of Vanilla Network or otherwise. This report is not and should not be relied upon by Vanilla Network or any reader of this report as any form of financial, tax, legal, regulatory, or other advice.

To the full extent permissible by applicable law, Bramah Systems, LLC. disclaims all warranties, express or implied. The information in this report is provided "as is" without warranty, representation, or guarantee of any kind, including the accuracy of the information provided. Bramah Systems, LLC. makes no warranties, representations, or guarantees about the Vanilla Network Protocol. Use of this report and/or any of the information provided herein is at the users sole risk, and Bramah Systems, LLC. hereby disclaims, and each user of this report hereby waives, releases, and holds Bramah Systems, LLC. harmless from, any and all liability, damage, expense, or harm (actual, threatened, or claimed) from such use.

Timeliness of Content

All content within this report is presented only as of the date published or indicated, to the commercially reasonable knowledge of Bramah Systems, LLC. as of such date, and may be superseded by subsequent events or for other reasons. The content contained within this report is subject to change without notice. Bramah Systems, LLC. does not guarantee or warrant the accuracy or timeliness of any of the content contained within this report, whether accessed through digital means or otherwise.

Bramah Systems, LLC. is not responsible for setting individual browser cache settings nor can it ensure any parties beyond those individuals directly listed within this report are receiving the most recent content as reasonably understood by Bramah Systems, LLC. as of the date this

report is provided to such individuals.

General Recommendations

Best Practices & Solidity Development Guidelines

Hardcoded addresses without update mechanism

The smart contracts contain multiple hardcoded addresses (the ERC20 address for the token, as well as an owner address). We suggest instead the usage of the Ownable smart contract from OpenZeppelin, which in addition to establishing access control, allows for transfer of ownership as may be necessary in the event of compromise.

Resolution: The team has resolved this issue by allowing the constructor to instantiate the variable value.

Events should be “emit” to differentiate between function calls

Events should be “emitted” to stand out with regular function calls, as per Solidity release notes:

General: Support and recommend using `emit EventName();` to call events explicitly.

In order to make events stand out with regards to regular function calls, `emit EventName()` as opposed to just `EventName()` should now be used to "call" events.

In particular, events that perform certain global sensitive actions (such as disabling swapping) should emit an event.

Resolution: The team has resolved this issue by adding the “**emit**” keyword.

Usage of send and transfer considered against best-practice

Following EIP-1884, the usage of transfer and send is no longer suggested, due to changing gas costs in their usage. Use .call.value(...)("") instead.

Resolution: As the transfer function is being called on the token directly, it is up to the team to assess the transfer methods of ERC20 token contracts they may wish to interface with.

Avoid state changes following transfer due to reentrancy concerns

As concisely explained by Consensys, "...the major dangers of calling external contracts is that they can take over the control flow, and make changes to your data that the calling function wasn't expecting". Function **UnStake**, which calls transfer and then later sets **stakers[msg.sender].stakedAmount** to 0 should be rewritten to incorporate the state change beforehand, or utilize a library such as **ReentrancyGuard**.

Resolution: The team has resolved this issue through following the checks-effects-interactions pattern.

Address visibility should be made explicit

The visibility in the below addresses should be explicitly defined for clarity.

IERC20 vanilla = IERC20(0x286F04ae08d18097d6b5cc2d96D7719972365a5d);

address owner = 0xFa50b82cbf2942008A097B6289F39b1bb797C5Cd;

Resolution: The team has resolved this issue through inclusion of visibility modifiers.

Specific Recommendations

Unique to the Vanilla Network Protocol

Sensitive parameter changing functions should emit an event

As the parameter setting functions (**changeMinTokensPerUser**, **changeMaxTokensPerUser**, **changeAPY**, **changeMaxSlots**) all allow for modification of potential rewards flow to users, it is suggested that these functions emit an event on invocation.

Resolution: The team has added in event emittance.

Faulty logic within sendRewards function

sendRewards within **RewardsWallet.sol** possesses a require function that utilizes the successful transfer as a determinant of whether or not the require was successful. The error message emitted, which references volumetric concerns, may not necessarily be true (as there are a multitude of reasons a potential send could revert).

Resolution: The team has removed this output from the **require** statement.

Code comments may lead to confusion

The **STAKE** function features a code block (replicated below) which enforces temporal boundaries of the function's execution. The code comment, while related to the variable at hand, does not adequately explain what is actually happening to the user. This may concern may be mitigated through more verbose failure messaging within the dApp.

require(block.timestamp <= subscriptionEnds, "subscription time expires");

Resolution: The team has added clarifying comments.

Transfer function called on token could have unintended consequences

Notably, as the token is transferred utilizing the transfer function of the provided ERC20 token, tokens written to not follow the ERC-20 specification may have unintended consequences, as

there is no guarantee that these tokens will perform as anticipated.

Resolution: This is an accepted risk of interfacing with external ERC20 tokens, and the team will exercise caution in the inclusion of new tokens.

Solidity style guide not followed

The protocol presently does not follow the Solidity style guide. Functions should be grouped according to their visibility and ordered as follows:

- constructor
- fallback function (if exists)
- external
- public
- internal
- private

Occurrences:

Numerous

Resolution: The team has resolved this error as of commit hash **5510949397a0c2de61fc2f5cf2451d4c4dde8e47**.

Adherence to Specification

Evaluating protocol adherence to specification

Specification Evaluation

The specification definition details information about the construction of a controlled APY value. While the functionality to control the APY is present within the codebase, high APY projects have often been associated with unrealistic or impossible financial expectations. We suggest individuals perform their own research, and as always suggest a thorough examination of token-economics when assessing a new platform.

Resolution: The team has stated intent to limit the number of “seats” to reduce the outflow of tokens into the public domain. Therefore, ensuring this protocol is sustainable

Toolset Warnings

Unique to the Vanilla Network

Overview

In addition to our manual review, our process involves utilizing static analysis and formal methods in order to perform additional verification of the presence of security vulnerabilities (or lack thereof). An additional part of this review phase consists of reviewing any automated unit testing frameworks that exist.

The following sections detail warnings generated by the automated tools and confirmation of false positives where applicable.

Compilation Warnings

No compilation warnings are generated by the contract as of present commit.

Test Coverage

The contract repository lacks substantial unit test coverage throughout. This may be due to the contracts stemming from pre-existing contracts, associated with the BREE protocol.

Static Analysis Coverage

The contract repository underwent heavy scrutiny with multiple static analysis agents, including:

- Securify
- MAIAN
- Mythril
- Oyente
- Slither

In each case, the team had either mitigated relevant concerns raised by each of these tools or provided adequate justification for the risk (such as adhering to the ERC-20 token standard). Notably, the basis for many of the contracts (specifically SafeMath) have previously been audited as well, in which static analyzers were also independently run.

Directory Structure

At time of review, the directory structure of the Vanilla Network smart contracts repository appeared as it does below. Our review, at request of Vanilla Network, covers the Solidity code (*.sol) as of commit hash **82d0bd1701673d2fe9d5d47b016e45d4ffa25176**.

.

├── ChocolatePool.sol

├── README.md

├── RewardsWallet.sol

└── StrawberryPool.sol

0 directories, 4 files