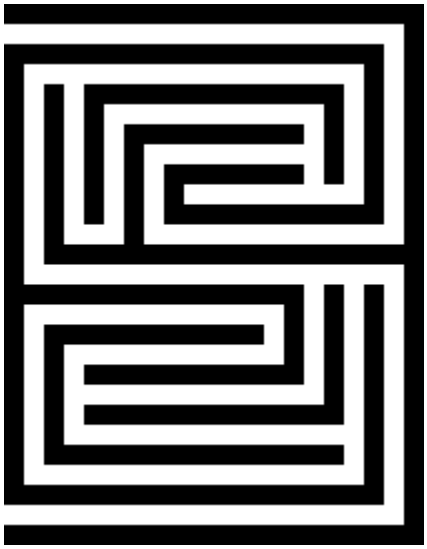

Ethlings Security Review Public Report

PROJECT: Ethlings Security Review

March 2021

Prepared For:

Ethlings

Ethlings

Prepared By:

Jonathan Haas | Bramah Systems, LLC.

jonathan@bramah.systems

Table of Contents

Ethlings Protocol Review

Executive Summary

Scope of Engagement

Bramah Systems, LLC was engaged in April of 2021 to perform a comprehensive security review of the Ethlings smart contracts (specific contracts denoted within the appendix). Our review was conducted over a period of two business days by both members of the Bramah Systems, LLC. executive staff.

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

Over time, certain elements of the protocol were modified, leading to a number of differences in initial pricing schema, control mechanisms, and overall performance of the protocol. Due to the frequency of these changes, Bramah and Ethlings engaged in numerous back and forth sessions resolving minor errors

Timeline

Review Commencement: April 9, 2021

Report Delivery: May 19th, 2021

Engagement Goals

The primary scope of the engagement was to evaluate and establish the overall security of the Ethlings 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. The contracts were provided via GitHub (commit hash **b608d9a485cc2a4177cb6be47b79abb9808bdbc9**)

Overall Assessment

Bramah Systems was engaged to evaluate and identify any potential security concerns within the codebase of the Ethlings protocol. 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. The team has since addressed these issues with a resolution or risk acceptance.

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 Ethlings 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 Ethlings protocol or any other relevant product, service or asset of Ethlings or otherwise. This report is not and should not be relied upon by Ethlings 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 Ethlings 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

Usage of weak PRNG and Insufficient Randomness

The contract makes use of a weak pseudo-random number generator, hashing values as replicated below:

```
// generate entropy using block values, sender, and serial number
bytes32 hash = keccak256(abi.encodePacked(block.number, blockhash(block.number - 1), msg.sender, avatarNumber));
```

We suggest using the Chainlink VRF in order to generate random numbers for usage, as the above approach lacks sufficient entropy.

Resolution: The Ethlings team has determined that the relative cost of changing to a purer randomization function (such as Chainlink VRF) outweighs the benefits of implementation.

Library code uses hardcoded gas-values

A hardcoded gas amount exists in ERC165Checker, function _callERC165SupportsInterface(address,bytes4). The function states the following:

```
- (success,result) = account.staticcall{gas: 30000}(encodedParams)
```

Hard-coded gas values should be avoided, as gas values can and do change over time (such as in the case of transfer)

Resolution: This library has been removed.

Constant not declared constant

Despite being constant (seconds per year), and under the “constant” comment, SECONDS_IN_A_YEAR is not declared constant.

Resolution: This change has been implemented.

Multiple external calls inside a loop could result in revert

External calls inside a loop may revert if one call triggers the fallback function of an external contact. One must validate that all external calls are to trusted contracts.

Resolution: Validation was performed and the relevant external calls are to trusted contracts.

Unused variables in library code

_balances, _allowances, _totalSupply, _name, _symbol, and **_decimals** are never used in ChangeToken.

Resolution: These items have been removed.

Sensitive changes should emit an event

Sensitive protocol changes, such as changing the team address or lock state (**setTeamAddress**, **setLocked**) should **emit** an event.

Resolution: Events are now emitted.

setEthlingsAddress should provide clearer error messaging

Presently, setEthlingsAddress function when set to the 0 address provides a notification that the ethlingsAddress cannot be changed. This is not functionally true, as it can be changed, just not to the zero address. The error message should be updated accordingly.

Resolution: While an error message has not been introduced, an event has been emitted in the event the address has been changed.

Test suite does not exist for EarningsRouter contract

The EarningsRouter contract does not presently have test coverage, nor is it referenced within any testing files.

Resolution: Tests have been introduced.

Specific Recommendations

Unique to the Ethlings Protocol

Batch mint can result in resource exhaustion

As the batchMint function performs quite a few computationally heavy calculations within a loop, it is suggested that an upper bound be placed on the number of iterations the loop may go through. This will allow for avoidance of resource exhaustion, which is encountered for sufficiently high iterations through the loop (although Bramah encountered slow-downs as soon as thirty iterations).

Resolution: An upper bound limit has been placed on the various computationally heavy calculations to limit gas exhaustion.

Toolset Warnings

Unique to the Ethlings Protocol

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 were encountered during the course of our audit.

Test Coverage

The contracts possess a number of functional unit tests encompassing various stages of the application lifecycle.

Static Analysis Coverage

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

- Securify
- MAIAN
- Mythril
- Oyente
- Slither

Directory Structure

At time of review, the directory structure of the Ethlings smart contracts repository appeared

as it does below. Our review, at request of Ethlings, covers the Solidity code (*.sol) as of commit hash **b608d9a485cc2a4177cb6be47b79abb9808bdbc9**

```
.
├── LICENSE
├── README.md
├── contracts
│   ├── ChangeToken.sol
│   ├── EarningsRouter.sol
│   ├── Ethlings.sol
│   ├── IChangeToken.sol
│   ├── IEthlings.sol
│   ├── IWearables.sol
│   ├── Wearables.sol
│   └── lib
│       ├── Address.sol
│       ├── BaseRelayRecipient.sol
│       ├── Context.sol
│       ├── ERC1155.sol
│       ├── ERC165.sol
│       ├── ERC165Checker.sol
│       ├── ERC20.sol
│       ├── ERC721.sol
│       ├── EnumerableMap.sol
│       ├── EnumerableSet.sol
│       ├── EthlingUtils.sol
│       ├── IERC1155.sol
│       ├── IERC1155MetadataURI.sol
│       ├── IERC1155Receiver.sol
│       ├── IERC165.sol
```

```
│       ├── IERC20.sol
│       ├── IERC20Metadata.sol
│       ├── IERC721.sol
│       ├── IERC721Enumerable.sol
│       ├── IERC721Metadata.sol
│       ├── IERC721Receiver.sol
│       ├── IRelayRecipient.sol
│       ├── Ownable.sol
│       ├── ReentrancyGuard.sol
│       ├── SafeERC20.sol
│       ├── SafeMath.sol
│       └── Strings.sol
├── migrations
│   └── 1_deploy_token.js
├── networks.js
├── package-lock.json
├── package.json
├── test
│   ├── avatar.js
│   ├── util
│   │   └── helper.js
│   └── wearables.js
└── test-environment.config.js

5 directories, 44 files
```