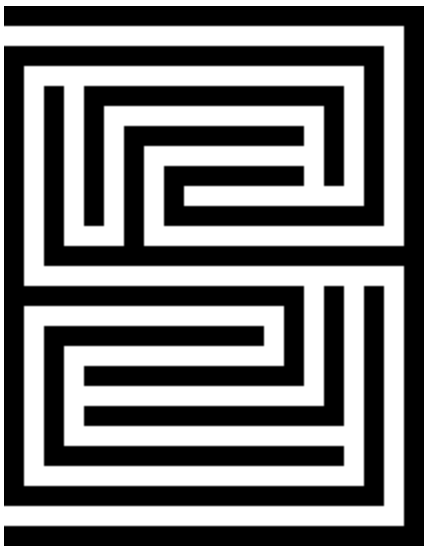

Poof Cash Public Report

PROJECT: Poof Cash Review
Fall 2021

Prepared For:

Brian Li | Poof Labs, Inc.

Prepared By:

Jonathan Haas | Bramah Systems, LLC.

jonathan@bramah.systems

Table of Contents

Poof Cash Security Review

Executive Summary

Scope of Engagement

Bramah Systems, LLC was engaged in Fall of 2021 to perform a comprehensive security review of the Poof Cash Protocol Soliditylang repository. Our review was conducted over a period of five business days by a member of Bramah Systems, LLC. executive staff.

Bramah's review pertains to Solidity code (*.sol) as of commit **e40921742d5d13ca1ab3d8071363e43922bbc8b4.**

Engagement Goals

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

- Is it possible for an attacker to manipulate the code?
- Does the Solidity code match the specification as provided?
- Is there a way to interfere with the software mechanisms?
- Are the arithmetic calculations trustworthy?

Protocol Specification

A basic specification document was compiled by the review team based upon review of the Poof Cash Protocol code and discussion with the team.

Overall Assessment

Bramah Systems was engaged to evaluate and identify multiple security concerns in the codebase of the Poof Cash architecture. During the course of our engagement, Bramah Systems denoted numerous instances wherein the software deviated from established best practices and procedures of secure software development. With limited exceptions (as described in this report), these instances were most commonly a result of structural limitations of Solidity and not due to inactions on behalf of the development team. The codebase benefits from detailed code comments throughout, which allowed for Bramah to review the codebase rapidly and without a deeper formal specification. As the code does make heavy usage of third party library code, the Poof Cash team should stay abreast of any security considerations of

these protocols. At the point of our review, there were no known high or medium severity risks presently outstanding with the third party code that had been implemented into the Poof Cash Protocol codebase.

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

Excess gas consumption

Excess gas consumption may occur when state variables (.balance or .length) are used in the condition of a for or while loop. Every iteration of the “for” loop consumes extra gas with these state variables present. In order to mitigate this excess consumption, if .balance, or .length are used several times, holding their value in a local variable is more gas efficient.

Resolution: This issue has been addressed.

Functions which award or assign a user should emit an event

There are numerous functions (such as **invite, batchInvite**) that augment users' ability to interact with the contracts. Modification to these should be made clear through event emittance.

Resolution: This issue has been addressed.

Error message can be modified to promote clarity

Presently, the error message within Poof.sol#112 lacks context as to which capacity the transfer is unfair in. This may prove critical during debugging of user transactions.

```
require(_fromArgs.amount - _fromArgs.extData.fee == _toArgs.amount, "Transfer is unfair");
```

Resolution: This issue has been addressed.

Toolset Warnings

Unique to the Poof Cash Protocol Protocol

Overview

In addition to our manual review, our process involves utilizing concolic analysis and dynamic testing in order to perform additional verification of the presence security vulnerabilities. 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, in addition to findings generated through manual inspection.

Compilation Warnings

No warnings were found at time of compilation that presented material concern.

Test Coverage

The contract repository possesses substantial unit test coverage throughout. This testing provides a variety of unit tests which encompass the various operational stages of the protocol

Static Analysis Coverage

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

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 (e.g. inherent risk of using native Ethereum constructs such as **timestamp**).

Certain tools, like Oyenete, do not run on newer versions of Solidity. Where applicable, Bramah manually performed validation checks based upon our understanding of the tool. With the

exception of known issues stemming from the usage of third party code, Bramah did not locate any instances of concern within the modified code.

Directory Structure

At time of review, the directory structure of the Poof Cash Protocol appeared as it does below. Our review, at request of Poof Cash Protocol, covers the Solidity code (*.sol) as of commit **e40921742d5d13ca1ab3d8071363e43922bbc8b4.**

.

```
├── LICENSE
├── README.md
├── circuits
│   ├── Deposit.circom
│   ├── DepositMini.circom
│   ├── DepositTemplate.circom
│   ├── MerkleTree.circom
│   ├── MerkleTreeUpdater.circom
│   ├── TreeUpdate.circom
│   ├── TreeUpdateMini.circom
│   ├── Withdraw.circom
│   ├── WithdrawMini.circom
│   └── WithdrawTemplate.circom
├── contracts
│   ├── Migrations.sol
│   ├── Poof.sol
│   ├── PoofLendable.sol
│   ├── PoofMintable.sol
│   ├── PoofMintableLendable.sol
│   ├── VIP.sol
│   ├── interfaces
│   │   ├── ICToken.sol
│   │   ├── ILendingPool.sol
│   │   ├── ISafeBox.sol
```

```
│   │   ├── IVerifier.sol
│   │   └── IWERC20.sol
│   ├── mocks
│   │   ├── ERC20Mock.sol
│   │   ├── FeeManager.sol
│   │   ├── MockLendingPool.sol
│   │   ├── MockSafeBox.sol
│   │   └── WERC20Mock.sol
│   ├── verifiers
│   │   ├── DepositMiniVerifier.sol -> ../../build/circuits/DepositMiniVerifier.sol
│   │   ├── DepositVerifier.sol -> ../../build/circuits/DepositVerifier.sol
│   │   ├── TreeUpdateMiniVerifier.sol -> ../../build/circuits/TreeUpdateMiniVerifier.sol
│   │   ├── TreeUpdateVerifier.sol -> ../../build/circuits/TreeUpdateVerifier.sol
│   │   ├── WithdrawMiniVerifier.sol -> ../../build/circuits/WithdrawMiniVerifier.sol
│   │   └── WithdrawVerifier.sol -> ../../build/circuits/WithdrawVerifier.sol
│   └── wrapped
│       ├── FeeBase.sol
│       ├── c
│       │   └── WrappedCToken.sol
│       └── moola
│           ├── WrappedMToken.sol
│           ├── wmCELO.sol
│           ├── wmcEUR.sol
│           └── wmcUSD.sol
├── hardhat.config.js
├── index.js
├── migrations
│   ├── 1_initial_migration.js
│   ├── 2_deploy_verifiers.js
│   ├── 3_deploy_vip.js
│   ├── 4_deploy_moola_tokens.js
```

```
│   └── 5_deploy_moola_poof.js
├── package.json
├── requirements.txt
├── scripts
│   ├── buildCircuit.sh
│   └── ganacheHelper.js
├── src
│   ├── account.js
│   ├── controller.js
│   ├── note.js
│   └── utils.js
├── test
│   ├── feeBase.test.js
│   ├── poof.aux.test.js
│   ├── poof.base.test.js
│   ├── wrappedCToken.test.js
│   └── wrappedMToken.test.js
├── truffle-config.js
└── yarn.lock

12 directories, 62 files
```