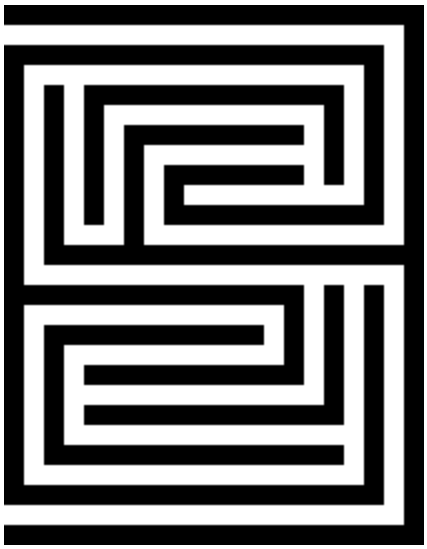

Superfarm Audit Public Report

PROJECT: Superfarm Audit

January 2021

Prepared For:

Superfarm Team | Superfarm, Inc.

https://superfarm.io

Prepared By:

Jonathan Haas | Bramah Systems, LLC.

jonathan@bramah.systems

Table of Contents

Superfarm Protocol Review

Executive Summary

Scope of Engagement

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

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

Timeline

Review Commencement: January 17th, 2021

Report Delivery: January 20th, 2021

Engagement Goals

The primary scope of the engagement was to evaluate and establish the overall security of the Superfarm 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

Contract specification was provided in the form of code comments and functional unit tests, along with a verbose specification document which provided justification for infrastructure decisions and structural fundamentals. This documentation was used to refine our general understanding of the protocol, including around certain implementation changes made -- for

example, adding an additional variable to the ERC-1155 invocation, which is used to assign a "**FeeOwner**".

Overall Assessment

Bramah Systems was engaged to evaluate and identify any potential security concerns within the codebase of the Superfarm Protocol. During the course of our engagement, Bramah Systems only few instances wherein the team deviated materially from established best practices and procedures of secure software development within DLT, as our report details.

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

TODO items remain within source code

TODO items still remain within the source code, indicating certain structural elements that should be decided before publication to the blockchain.

Resolution: Bramah confirmed with the team that the remaining TODO comments have been removed and the concerns they regarded have been addressed.

Unbounded loop length controlled by user supplied input

There are multiple loops throughout the protocol that are influenced by user supplied values. As these loops have no inherent limitation to size (there exists no function to validate that loops do not exceed a certain length), potential concern for resource exhaustion exists (namely, that a loop will be iterated through to the point of exhausting available gas for execution, causing the operation to fail.

File: SuperFarm-master/superfarm-contracts/contracts/Fee1155.sol

Lines: 202-211

File: SuperFarm-master/superfarm-contracts/contracts/Fee1155.sol

Lines: 140-159

File: SuperFarm-master/superfarm-contracts/contracts/Shop1155.sol

Lines: 198-200

File: SuperFarm-master/superfarm-contracts/contracts/Shop1155.sol

Lines: 122-124

File: SuperFarm-master/superfarm-contracts/contracts/Shop1155.sol

Lines: 145-174

File: SuperFarm-master/superfarm-contracts/contracts/Shop1155.sol

Lines: 155-169

File: SuperFarm-master/superfarm-contracts/contracts/Shop1155.sol

Lines: 165-167

File: SuperFarm-master/superfarm-contracts/contracts/Staker.sol

Lines: 376-380

File: SuperFarm-master/superfarm-contracts/contracts/Staker.sol

Lines: 180-188

File: SuperFarm-master/superfarm-contracts/contracts/Staker.sol

Lines: 269-280

File: SuperFarm-master/superfarm-contracts/contracts/Staker.sol

Lines: 311-320

File: SuperFarm-master/superfarm-contracts/contracts/Staker.sol

Lines: 360-364

File: SuperFarm-master/superfarm-contracts/contracts/Token.sol

Lines: 181-181

File: SuperFarm-master/superfarm-contracts/contracts/ShopEtherMinter1155.sol

Lines: 94-109

File: SuperFarm-master/superfarm-contracts/contracts/ShopEtherMinter1155.sol

Lines: 63-67

File: SuperFarm-master/superfarm-contracts/contracts/ShopEtherMinter1155.sol

Lines: 78-81

Resolution: The team provided the following risk mitigation, which we feel provides adequate justification risk mitigation. Where appropriate, this could be communicated to users (such as in developer documentation)

“We've decided that there's no good way for us to limit this one at the smart-contract level. In all of the situations where the loop length was unbounded we would happily allow users to operate with as many elements as would possibly fit within the gas limit. Our interface will have proper sanity-checks for situations where this one could get hairy and risk users wasting gas, but for the situation of users who choose to interact with our contracts directly we are going to leave the matter of the gas limit in their hands.”

Sensitive parameter changing functions should emit an event

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

Resolution: An event has been added.

External is preferable to public for gas optimization

As noted by multiple other static code analysis tools, the usage of external function visibility is preferable to public, in that external functions are prevented from being called internally, whereas public functions can be called internally. This results in a gas optimization that is due to the fact that Solidity copies arguments to memory on a public function whereas external functions read from calldata (which is cheaper than memory allocation)

File: SuperFarm-master/superfarm-contracts/contracts/FeeOwner.sol

Lines: 41-44

File: SuperFarm-master/superfarm-contracts/contracts/Fee1155.sol

Lines: 76-78

File: SuperFarm-master/superfarm-contracts/contracts/Fee1155.sol

Lines: 197-213

File: SuperFarm-master/superfarm-contracts/contracts/Fee1155.sol

Lines: 124-167

File: SuperFarm-master/superfarm-contracts/contracts/Fee1155.sol

Lines: 178-186

File: SuperFarm-master/superfarm-contracts/contracts/Fee1155.sol

Lines: 87-89

File: SuperFarm-master/superfarm-contracts/contracts/Shop1155.sol

Lines: 121-125

File: SuperFarm-master/superfarm-contracts/contracts/Shop1155.sol

Lines: 183-189

File: SuperFarm-master/superfarm-contracts/contracts/Shop1155.sol

Lines: 213-260

File: SuperFarm-master/superfarm-contracts/contracts/Shop1155.sol

Lines: 197-202

File: SuperFarm-master/superfarm-contracts/contracts/Shop1155.sol

Lines: 136-175

File: SuperFarm-master/superfarm-contracts/contracts/Staker.sol

Lines: 158-167

File: SuperFarm-master/superfarm-contracts/contracts/Staker.sol

Lines: 175-189

File: SuperFarm-master/superfarm-contracts/contracts/Staker.sol

Lines: 134-139

File: SuperFarm-master/superfarm-contracts/contracts/Staker.sol

Lines: 397-416

File: SuperFarm-master/superfarm-contracts/contracts/Staker.sol

Lines: 146-148

File: SuperFarm-master/superfarm-contracts/contracts/Staker.sol

Lines: 373-382

File: SuperFarm-master/superfarm-contracts/contracts/Staker.sol

Lines: 196-198

File: SuperFarm-master/superfarm-contracts/contracts/Staker.sol

Lines: 388-390

File: SuperFarm-master/superfarm-contracts/contracts/Staker.sol

Lines: 423-440

File: SuperFarm-master/superfarm-contracts/contracts/Staker.sol

Lines: 448-450

File: SuperFarm-master/superfarm-contracts/contracts/Staker.sol

Lines: 238-255

File: SuperFarm-master/superfarm-contracts/contracts/Staker.sol

Lines: 457-465

File: SuperFarm-master/superfarm-contracts/contracts/Token.sol

Lines: 42-45

File: SuperFarm-master/superfarm-contracts/contracts/Token.sol

Lines: 50-54

File: SuperFarm-master/superfarm-contracts/contracts/FarmRecords.sol

Lines: 49-51

File: SuperFarm-master/superfarm-contracts/contracts/FarmRecords.sol

Lines: 41-43

File: SuperFarm-master/superfarm-contracts/contracts/FarmRecords.sol

Lines: 25-27

File: SuperFarm-master/superfarm-contracts/contracts/FarmRecords.sol

Lines: 33-35

File: SuperFarm-master/superfarm-contracts/contracts/ShopEtherMinter1155.sol

Lines: 87-110

File: SuperFarm-master/superfarm-contracts/contracts/ShopEtherMinter1155.sol

Lines: 56-68

File: SuperFarm-master/superfarm-contracts/contracts/ShopEtherMinter1155.sol

Lines: 73-82

Resolution: Functions have been appropriately tagged for the circumstances in which they are called.

ETH transfer occurring within a loop

If at least one address cannot receive ETH (e.g. a contract with a default fallback function), the whole transaction will be reverted. In purchaseItems, this consideration should be taken into account.

File: SuperFarm-master/superfarm-contracts/contracts/ShopEtherMinter1155.sol

Lines: 94-109

Resolution: The Superfarm team provided the following risk mitigation: “From looking at the instances where this can occur, namely when purchasing multiple items from a Shop contract, I see your concern that a single malformed FeeOwner's fee owner on an item would render the entire purchase transaction inoperable. However, much like the unbounded loop length example, we only foresee a FeeOwner entering such a state if a user interacted with their FeeOwner outside of our interface and updated the owner to a broken contract. We leave the concern of whether or not such a contract can actually receive Ether to the user performing such advanced operations.

In response to your feedback we considered different fixes for this issue:

1. ignoring the malformed item and letting purchase succeed with all other items

2. simply ignoring malformed fees and letting the purchase continue

Ultimately, we chose not to pursue the first option due to its perceived impact on users; if a user attempts to purchase ten different items we would rather the entire transaction fail than allow them to only receive nine of those items and worry about having been scammed. We chose not to pursue the second option due to our extreme sensitivity to the purchase function's gas costs and how rarely we anticipate this error happening in the first place. Ultimately, we can identify such broken items and conceal them on our application front-end.”

Bramah feels this risk mitigation adequately addresses such concerns, and agrees that advanced usage outside of the team developed interface carries an understandable element of risk. Bramah believes think the situation as presented by the team does accurately represent potential user feedback, and that the mitigation as presented adequately would address such feedback.

Hardhat/console.sol should be removed before deployment

When running your contracts and tests on Hardhat Network you can print logging messages and contract variables calling console.log() from your Solidity code. To use it you have to import Hardhat's console.log from your contract code. This debug information should be removed from any production deployments.

File: SuperFarm-master/superfarm-contracts/contracts/FeeOwner.sol

Lines: 7

File: SuperFarm-master/superfarm-contracts/contracts/Shop1155.sol

Lines: 10

File: SuperFarm-master/superfarm-contracts/contracts/ShopEtherMinter1155.sol

Lines: 10

File: SuperFarm-master/superfarm-contracts/contracts/Staker.sol

Lines: 9

File: SuperFarm-master/superfarm-contracts/contracts/Token.sol

Lines: 6

Resolution: The team has removed these references.

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. The contract presently makes use of **transfer** throughout.

Resolution: The Superfarm team replaced all instances of transfer with .call{ value: amount }(""), while introducing requirements to validate successful execution of the transfer. ReentrancyGuard was additional added to mitigation concern regarding potentially reentrant functions.

Specific Recommendations

Unique to the Superfarm Protocol

Highly permissive owner account and centralization of power

The deploying account possesses a number of highly actions (namely, changing various distribution and reward preferences -- notably mentioned in the scope of time-based promotions). This deploying account should (where possible) minimize usage of the associated key (e.g. performing transactions, using as a regular user account) and perform other operational security best practices. Potentially, this could involve transferring ownership to a MultiSignature governance.

Resolution: The Superfarm team provided the following details regarding their anticipated ownership plans: “We plan to follow the footsteps of many other successful projects and migrate owner control to a multisignature wallet behind a time lock; ultimately we would like to move fast at this stage and will need time to thoroughly test such a multisignature and timelock setup before we deploy it.”

Considerations of “data” field for ERC1155 tokens

As the **data** field allows for relatively arbitrary data to be attached to ERC1155 tokens, it is useful to remember that any untrusted tokens which are processed could potentially have gas limitations and considerations.

Resolution: The Superfarm team provided the following details regarding the usage of the **data** element: “We again approach this with the same freedom afforded to users who might run into gas limitations on our unbounded arrays: if a user creates an ERC-1155 token with data that causes gas limitation issues, then we plan to leave that matter to the user creating such an item.”

Performance of a “closed” system

As the protocol makes multiple calls to external contracts (namely, minting), one should be careful to ensure that the specification of the remote token matches that of the ERC1155. This avoids potential instances in which reentrancy could be abused (alternatively, a

ReentrancyGuard may be added at the cost of gas increase).

Resolution: ReentrancyGuard has been added to mitigate concerns regarding potential function reentrancy.

Toolset Warnings

Unique to the Superfarm 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 warnings were present at time of compilation.

Test Coverage

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

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-1155 standard).

Directory Structure

At time of review, the directory structure of the Superfarm smart contracts repository appeared as it does below. Our review, at request of Superfarm, covers the Solidity code (*.sol) as of commit-hash **f9f41eb24e638c77189ef3d630a62eb54e8d1551** of the Superfarm repository.

.

```
├── contracts
│   ├── FarmRecords.sol
│   ├── Fee1155.sol
│   ├── FeeOwner.sol
│   ├── Shop1155.sol
│   ├── ShopEtherMinter1155.sol
│   ├── Staker.sol
│   └── Token.sol
├── docs
│   ├── FarmRecords.md
│   ├── Fee1155.md
│   ├── FeeOwner.md
│   ├── Shop1155.md
│   ├── ShopEtherMinter1155.md
│   ├── Staker.md
│   ├── Token.md
│   └── design-rationale.pdf
├── hardhat-ganache-tests.config.js
├── hardhat.config.js
├── package-lock.json
├── package.json
├── scripts
│   └── deploy.js
```

```
├── test
│   ├── Fee1155.test.js
│   ├── Shop1155.test.js
│   ├── ShopEtherMinter1155.test.js
│   ├── Staker.test.js
│   └── Token.test.js
└── yarn.lock

4 directories, 26 files
```