Prepared by: Sherlock Security

Customer: Surf Finance

Date: 10/28/2020

Blockchain: Ethereum

Language: Solidity

This document will contain a summary review of Sherlock Security's audit. This report may contain confidential information and is for internal company use, however may be disclosed to public at Client discretion.

Introduction

Sherlock Security (Consultant) was contracted by Surf Finance (Client) to conduct a Smart Contract Code Review and Security analysis report. The summary review will outline Consultant's findings with relation to the smart contract code submitted and any possible security concerns that may exist relating to function or utility, to the best of the review team's ability.

The report will contain notes relating to any vulnerability found in highest-to-lowest priority from: High, Medium, to Low. If no vulnerabilities are discovered, it will be noted as such.

Audit Overview

Surf Finance is a decentralized ‘yield farm’ model. Yield farms utilize a number of contracts to facilitate proper mechanics and compartmentalize sections of code to specific uses and functions. The review will contain audit notes on three (3) different Solidity contracts used in Surf Finance deployed code.

Surf.sol - Governance token contract, able to make changes via governance through submitted proposals

*Governance features used in a fairly distributed ecosystem lessens centralized team control and puts initiative back into community hands through their ability to vote in proposals.

Tito.sol- Primary farm contract where users stake or unstake their Liquidity Pool (LP) tokens, and collect platform rewards

*LP stake has a 10% fee paid upon deposit to the pool, 5% goes to the team, and 5% goes to Whirlpool.sol. A major update from MasterChef.sol, the fee acts as a deterrent against users who jump between pools to farm highest yield without penalty. The fee update intends to increase sustainability by discouraging pool jumping for yield.

Whirlpool.sol – Whirlpool accrues Surf via 5% LP staking fee, to be distributed in the second phase of Surf, once all SURF tokens are minted by Tito.sol.

*Whirlpool has no deposit fee, but a 10% withdrawal fee, structured similar to Tito.sol but no dev fee taken. 5% of withdrawal is permanently locked into SURF/ETH liquidity on Uniswap and half is distributed to Whirlpool staked SURF tokens. The fee update intends

to increase sustainability by permanently locking project tokens into its liquidity pool, while rewarding staked SURF token holders.

Summary

Surf Finance contracts are open-source and available for public review. This provides an added layer of external consensus from community members and outside experts. Forming consensus is important in decentralized development. Although updates have been made, it is important to note that Surf is built upon forked Sushi MasterChef.sol contract code, which has had its own round of audits and returned without critical security error. Surf also utilizes forked code from COMP which has undergone its own reviews which can be found publicly.

Also having been allowed access to Surf's developer chat, Sherlock team was able to review updates to Surf contracts in real-time, allowing for oversight of contract development through their contract version updates. Over the course of contract development, no critical errors or flaws were found in Sherlock's review. In addition to other developers and auditors, Sherlock maintained real-time consultation and communication with Surf's development team.

The team is also utilizing multi-signature (multi-sig) which is a democracy vote in order to decision function use by consensus. There are 6 signers on the multi-sig wallet which control Surf.sol, Tito.sol, and Whirlpool.sol. Without majority vote, functions cannot be utilized. This is a beneficial security measure for any project.

Function Review

The following section will include contract functions, intended uses, and possible vectors for compromise.

High Concern

I. Surf.sol
   a.) **setContractAddresses()** - Permits contract owner ability to update Tito, Whirlpool, and SURF-ETH LP token contracts
   b.) **migrateLockedLPTokens()** - Permits contract owner to transfer contract locked LP tokens to another wallet/contract

II. Tito.sol
   a.) **setWhirlpoolContract()** - Permits contract owner ability to change Whirlpool address
   b.) **addToWhitelist()** - Permits contract owner ability to Whitelist other wallets/contracts

III. Whirlpool.sol
   a.) No High Priority issues found in Whirlpool.sol

Medium Concern

I. Surf.sol
   a.) **setTransferFee()** - Permits contract owner ability to update SURF transfer fee
   b.) **addToTransferWhitelist()** - Permits contract owner ability to add new contract to Transfer Whitelist

II. Tito.sol
   a.) **addPool()** - Permits contract owner ability to add new token pool to contract
   b.) **setApr()** - Permits contract owner ability to increase or decrease token pool APR

III. Whirlpool.sol
   a.) No Medium Priority issues found in Whirlpool.sol

Low Concern

I. Surf.sol
   a.) No Low Priority issues found in Surf.sol
II. Tito.sol
   a.) **removeFromWhitelist()** – Permits contract owner ability to remove a contract from LP whitelist
III. Whirlpool.sol
   a.) No Low Priority issues found in Whirlpool.sol

Function Summary Review

The highest priority concerns lie in the contract owner's ability to make changes/updates to where Surf.sol and Tito.sol are pointed to. LP or SURF tokens could be rerouted to another external contract if the functions are used maliciously.

However, with community diligence, this would easily be detectable on the public ledger, and should be monitored in case any of these functions are utilized by owner without just cause. Another note is that these functions are also necessary for possible future contract deployment, in case of unreported flaw or bug discovered in the current version of the contract.

Additionally, with a governance system in place, ideally many of these changes occur by voting consensus of SURF holders only. In the event these functions are utilized by owner without cause, this would be a major warning of malicious activity. Again, community diligence can be performed by ensuring contract owner is not using these functions maliciously via public ledger.

Medium priority concerns, similarly to the High Priority concerns, stem from contract owner permitted functions: ability to update/edit Whitelist addresses, token pools, and pool APR. As above these functions could potentially be used maliciously to unfairly

advantage one pool over another, add illegitimate token pools, and otherwise affect the SURF economy negatively and/or to benefit an external party.

Again as in the High Priority concerns, community diligence can review public ledger for any of these called functions by the owner. Similarly, there are necessary uses for these functions that allow for ease of use in contract management, should there be needed updates in the future.

Conclusion

The functions outlined as having high or medium abuse concern should always be regarded as concerning in any contract, by any team.

This however, is not indication alone of malicious intent. Often the same functions have legitimate and necessary use in the real time operation of the contracts. In decentralized development, any investor or end-user has the ability to perform due diligence on any open source contract in use, and is encouraged to do so.

With reference to the above functions and their intended uses, the public ledger is the best and first defense. Any functions utilized without notice can be flagged for concern and investigation can be conducted into why it was used. Diligent review of the ledger regarding the referenced contract function use should not be a single action but an ongoing assessment, in order to maintain accountability to their intended use by contract owner.

This report is intended to cover common, known uses of these functions as they pertain to how they could be used against end-users. It is not intended to cover all possible or potential exploits but rather the most commonly used in decentralized applications and contracts.

In conclusion, the audit finds that High and Medium Priority concerns listed could potentially be used maliciously, however is countered by the fact that code is open source and function use can be reviewed on public ledger. With proposed Surf governance in place, ongoing review and assessment can occur to monitor any use of the functions for accountability.

Disclaimer

The smart contract(s) submitted to Sherlock Security for audit have been analyzed by experienced industry developers using current best practices. The audit report pertains to the smart contract code submitted, including: smart contract source code, source code compilation, contract deployment, and functionality of intended use.

The audit makes no statements, guarantees, or warranties regarding the platform, team, or security of the code. A single audit alone cannot be considered sufficient evidence of security regarding safety and utility of the code under review. While the audit has been completed under high standard of review, it is important to note several independent audit reports should be obtained to ensure absolute security of the code, contract, and functions.

Technical Disclaimer

Smart contracts are deployed on a blockchain platform such as ETH. The blockchain, its programming language, and other software related to its use can have vulnerabilities of their own that can lead to exploit, thus Sherlock Security cannot fully guarantee explicit security of the audited smart contracts.