smart state

Web3 security easier than ever

Magic Square

SQRpProRata
Smart Contract Audit Interim Report

Ver. 2.2
July 25, 2024

smartstate.tech

Table of Contents:

Methodology

During the audit process we have analyzed various security aspects in line with our methodology, which includes:

- Manual code analysis
- Best code practices
- ERC20/BEP20 compliance (if applicable)
- Locked ether
- Pool Asset Security (backdoors in the underlying ERC-20)
- FA2 compliance (if applicable)
- Logical bugs & code logic issues
- Error handling issues
- General Denial Of Service(DOS)
- Cryptographic errors
- Weak PRNG issues
- Protocol and header parsing errors
- Private data leaks
- Using components with known vulnerabilities
- Unchecked call return method
- Code with no effects
- Unused vars
- Use of deprecated functions
- Authorization issues
- Reentrancy
- Arithmetic Overflows / Underflows
- Hidden Malicious Code
- External Contract Referencing
- Short Address/Parameter Attack
- Race Conditions / Front Running
- Uninitialized Storage Pointers
- Floating Points and Precision
- Signatures Replay

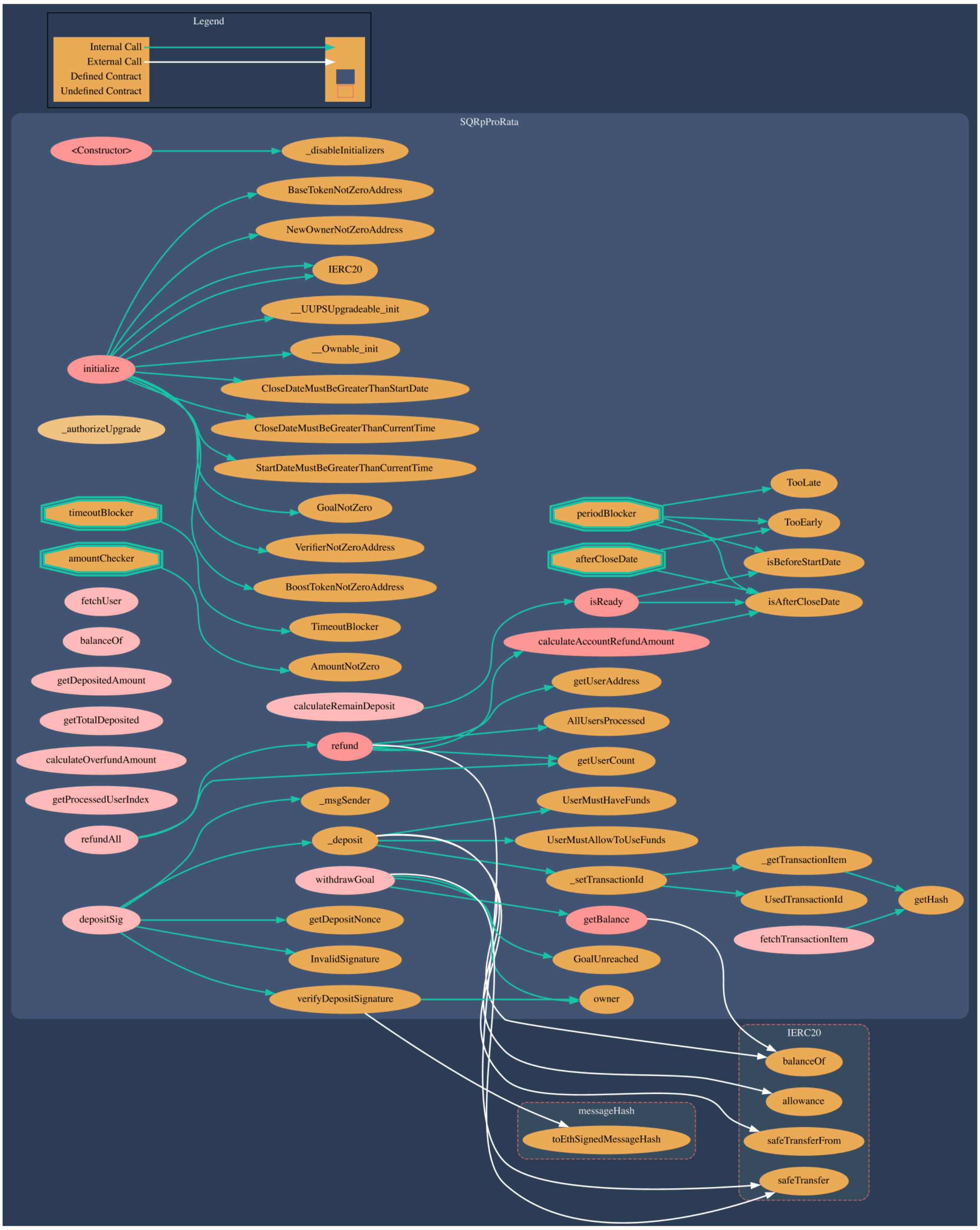
Vulnerabilities we have discovered are listed below.

Vulnerabilities found:

Severity	Amount
INFO	0
LOW	0
MEDIUM	0
HIGH	1
CRITICAL	0
TOTAL:	1

1. SQRpProRata.sol

1.1 Contract structure



Pic.1.1 SQRpProRata.sol structure

1.2 Contract methods analysis

`constructor() returns()`
Vulnerabilities not detected
Math issues not detected

`initialize(SQRpProRata.ContractParams) returns()`
Vulnerabilities not detected
Math issues not detected

`_authorizeUpgrade(address) returns()`
Vulnerabilities not detected
Math issues not detected

`getContractName() returns(string)`
Vulnerabilities not detected
Math issues not detected

`getContractVersion() returns(string)`
Vulnerabilities not detected
Math issues not detected

`getBaseGoal() returns(uint256)`
Vulnerabilities not detected
Math issues not detected

`getStartDate() returns(uint32)`
Vulnerabilities not detected
Math issues not detected

`getCloseDate() returns(uint32)`
Vulnerabilities not detected
Math issues not detected

`getDepositRefundFetchReady() returns(bool)`
Vulnerabilities not detected
Math issues not detected

`getAccountCount() returns(uint32)`
Vulnerabilities not detected
Math issues not detected

`getAccountByIndex(uint32) returns(address)`
Vulnerabilities not detected
Math issues not detected

`getDepositRefundTokensInfo()`
`returns(IDepositRefund.DepositRefundTokensInfo)`

Vulnerabilities not detected
Math issues not detected

`getDepositRefundAllocation(address) returns(uint256)`
Vulnerabilities not detected
Math issues not detected

`getDepositRefundAccountInfo(address)`
`returns(IDepositRefund.DepositRefundAccountInfo)`

Vulnerabilities not detected
Math issues not detected

`getDepositRefundContractInfo()`
`returns(IDepositRefund.DepositRefundContractInfo)`

Vulnerabilities not detected
Math issues not detected

`isBeforeStartDate() returns(bool)`
Vulnerabilities not detected
Math issues not detected

`isAfterCloseDate() returns(bool)`
Vulnerabilities not detected
Math issues not detected

`isDepositReady() returns(bool)`
Vulnerabilities not detected
Math issues not detected

`isReachedBaseGoal() returns(bool)`
Vulnerabilities not detected
Math issues not detected

`fetchAccountInfo(address) returns(SQRpProRata.AccountInfo)`
Vulnerabilities not detected
Math issues not detected

`getBaseBalance() returns(uint256)`
Vulnerabilities not detected
Math issues not detected

`getBoostBalance() returns(uint256)`
Vulnerabilities not detected
Math issues not detected

`balanceOf(address) returns(uint256,uint256)`
Vulnerabilities not detected
Math issues not detected

`getHash(string) returns(bytes32)`
Vulnerabilities not detected
Math issues not detected

`getAccountDepositNonce(address) returns(uint32)`
Vulnerabilities not detected
Math issues not detected

`calculateRemainDeposit() returns(uint256)`
Vulnerabilities not detected
Math issues not detected

`calculateAccidentAmount() returns(uint256)`
Vulnerabilities not detected
Math issues not detected

`calculateOverfundAmount() returns(uint256)`
Vulnerabilities not detected
Math issues not detected

`divisionRoundUp(uint256,uint256) returns(uint256)`
Vulnerabilities not detected
Math issues not detected

`calculateAccountBaseAllocation(address) returns(uint256)`
Vulnerabilities not detected
Math issues not detected

`calculateAccountBaseRefund(address) returns(uint256)`
Vulnerabilities not detected
Math issues not detected

`calculateAccountBoostRefund(address) returns(uint256)`
Vulnerabilities not detected
Math issues not detected

`calculateAccountBoostAverageExchangeRate(address) returns(uint256)`
Vulnerabilities not detected
Math issues not detected

`calculateAccountShare(address) returns(uint256)`
Vulnerabilities not detected
Math issues not detected

`fetchTransactionItem(string)`
`returns(SQRpProRata.TransactionItem)`

Vulnerabilities not detected
Math issues not detected

`_getTransactionItem(string)`
`returns(bytes32,SQRpProRata.TransactionItem)`

Vulnerabilities not detected
Math issues not detected

`getProcessedAccountIndex() returns(uint32)`
Vulnerabilities not detected
Math issues not detected

`calculatedTotalBoostRefundAmount() returns(uint256)`
Vulnerabilities not detected
Math issues not detected

`calculatedRequiredBoostAmount() returns(uint256)`
Vulnerabilities not detected
Math issues not detected

`calculateExcessBoostAmount() returns(uint256)`
Vulnerabilities not detected
Math issues not detected

`calculateBaseSwappedAmount(uint32) returns()`
Vulnerabilities not detected
Math issues not detected

`calculateDecimalsFactors(uint8,uint8)`
`returns(uint256,uint256)`

Vulnerabilities not detected
Math issues not detected

`calculateRemainProcessedAccountAmount() returns(uint256)`
Vulnerabilities not detected
Math issues not detected

`_setTransactionId(string,uint256) returns()`
Vulnerabilities not detected
Math issues not detected

`_deposit(address,uint256,bool,uint256,string,uint32) returns()`
Vulnerabilities not detected
Math issues not detected

`verifyDepositSignature(address,uint256,bool,uint256,uint32,string,uint32,bytes) returns(bool)`
Vulnerabilities not detected
Math issues not detected

`depositSig(SQRpProRata.DepositSigParams) returns()`	
Vulnerabilities not detected	
Math issues not detected	
TOKEN FLOW	Tokens In, public

`refund(uint32) returns()`	
Vulnerabilities not detected	
Math issues not detected	
TOKEN FLOW	Tokens Out, onlyOwner

refundAll() returns()	
Vulnerabilities not detected	
Math issues not detected	
TOKEN FLOW	Tokens Out, onlyOwner

withdrawBaseGoal() returns()	
Vulnerabilities not detected	
Math issues not detected	
TOKEN FLOW	Tokens Out, onlyOwner

calculateBaseSwappedAmountAll() returns()
Vulnerabilities not detected
Math issues not detected

`withdrawBaseSwappedAmount() returns()`	
Vulnerabilities not detected	
Math issues not detected	
TOKEN FLOW	Tokens Out, onlyOwner

`withdrawExcessTokens() returns()`	
Vulnerabilities not detected	
Math issues not detected	
TOKEN FLOW	Tokens Out, onlyOwner

Status: Acknowledged	HIGH
`forceWithdraw(address,address,uint256) returns()`	
Centralization risks. Owner can sweep any token from the contract.	
TOKEN FLOW	Tokens Out, onlyOwner

Verification checksums

Contract	Bytecode hash(SHA-256)
SQRpProRata.sol	5255a45364c5aecc7bc6d7a9690094a1a6994acb9ff440ef4b9a0cd5680a9333