Magic Square

SQRpProRata
Smart Contract Audit Interim Report

Ver. 1.2
June 15, 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

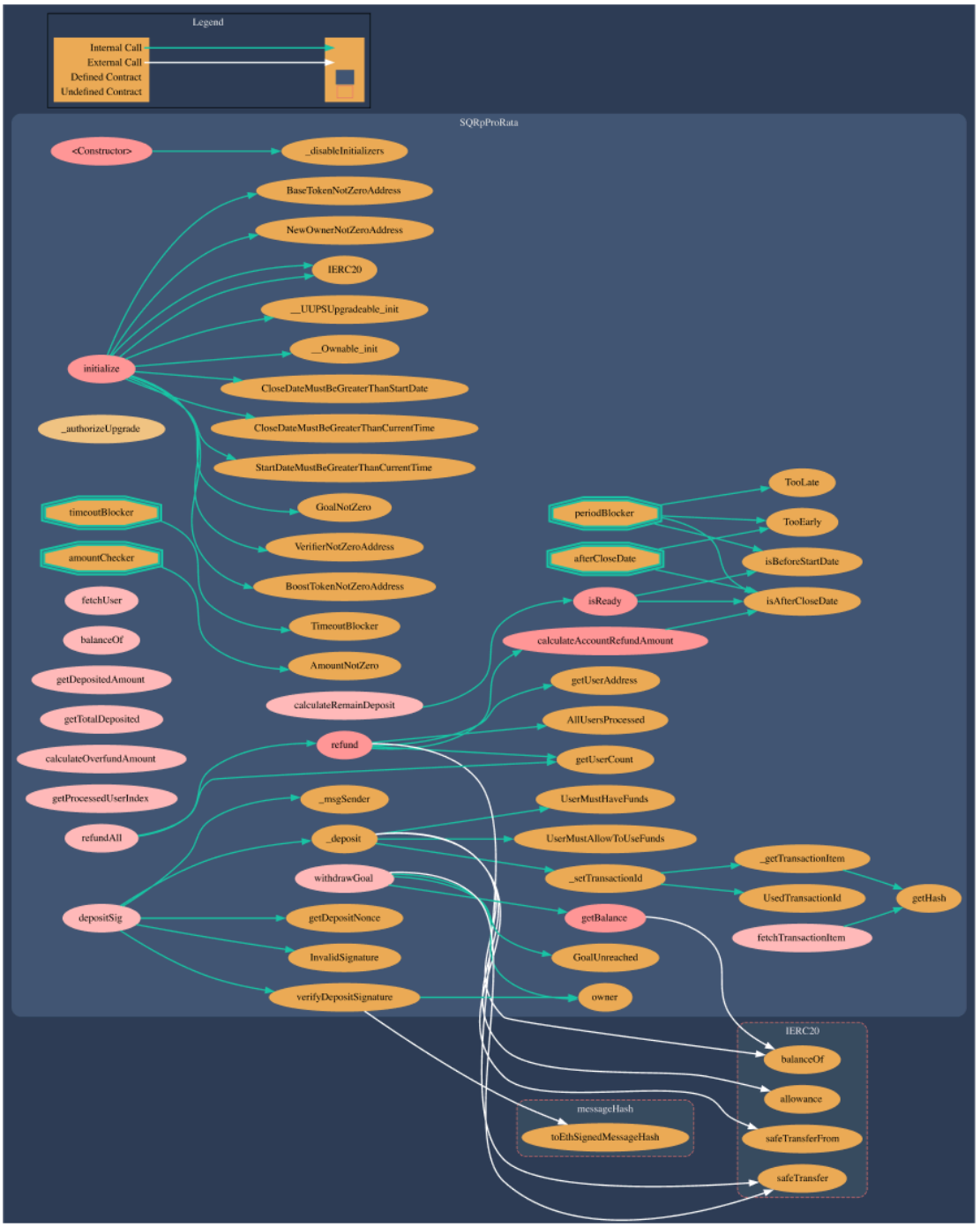
Vulnerabilities we have discovered are listed below.

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()`
Vulnerabilities not detected
Math issues not detected

`initialize(`
`address _newOwner,`
`address _baseToken,`
`address _boostToken,`
`address _verifier,`
`uint256 _goal,`
`uint32 _startDate, //0 - skip`
`uint32 _closeDate`
`)`

Vulnerabilities not detected
Math issues not detected

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

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

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

`isReady()`
Vulnerabilities not detected
Math issues not detected

`getUserCount()`
Vulnerabilities not detected
Math issues not detected

`fetchUser(address account)`
Vulnerabilities not detected
Math issues not detected

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

`balanceOf(address account)`
Vulnerabilities not detected
Math issues not detected

`getHash(string calldata value)`
Vulnerabilities not detected
Math issues not detected

`getDepositNonce(address account)`
Vulnerabilities not detected
Math issues not detected

`getUserAddress(uint32 index)`
Vulnerabilities not detected
Math issues not detected

`getDepositedAmount(address account)`
Vulnerabilities not detected
Math issues not detected

`getTotalDeposited()`
Vulnerabilities not detected
Math issues not detected

	HIGH
`calculateAccidentAmount()`	
Function incorrectly calculates accident amount. Let's assume we have a goal of 15k tokens. We have a total deposited of 20k and totalOutput of 15k (we didn't refund yet). And someone has accidently sent 1k tokens to contract, so it's baseBalance for now is 6k. Function will try to subtract 20k of totalDeposited from 6k of baseBalance, which will result in underflow and revert. We assume that this function can be rewritten the following way: return baseBalance - (totalDeposited - totalOutput) without a need in any if conditions.	

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

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

`calculateAccountRefundAmount(address account)`
Vulnerabilities not detected
Math issues not detected

`fetchTransactionItem(`
`string calldata transactionId`
`)`

Vulnerabilities not detected
Math issues not detected

`_getTransactionItem(`
`string calldata transactionId`
`)`

Vulnerabilities not detected
Math issues not detected

`getProcessedUserIndex()`
Vulnerabilities not detected
Math issues not detected

`_setTransactionId(uint256 amount, string calldata transactionId)`
Vulnerabilities not detected
Math issues not detected

```
_deposit(
address account,
uint256 amount,
string calldata transactionId,
uint32 timestampLimit
)
```

Vulnerabilities not detected

Math issues not detected

```
verifyDepositSignature(
address account,
uint256 amount,
bool boost,
uint32 nonce,
string calldata transactionId,
uint32 timestampLimit,
bytes calldata signature
)
```

Vulnerabilities not detected

Math issues not detected

depositSig(uint256 amount, bool boost, string calldata transactionId, uint32 timestampLimit, bytes calldata signature)	
Vulnerabilities not detected	
Math issues not detected	
TOKEN FLOW	Tokens In, public

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

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

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

Verification checksums

Contract name	Bytecode hash(SHA-256)
SQRpProRata.sol	fd5858006fce0f111914a45c6aa805062ab5ae9befe5296d7b4f7f700e051040