smart state
Web3 security easier than ever

Magic Square

SQRpProRata
Smart Contract Audit Interim Report

Ver. 1.1
June 10, 2024

Table of Contents:

Table of Contents........ 2
Methodology........ 3
Vulnerabilities found........ 4
1. SQRpProRata.sol........ 5
    1.1 Contract structure........ 5
    1.2 Contract methods analysis........ 6
Verification checksums........ 17

Methodology

During the audit process we have analyzed multiple security aspects in line with our methodology, including:

- 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

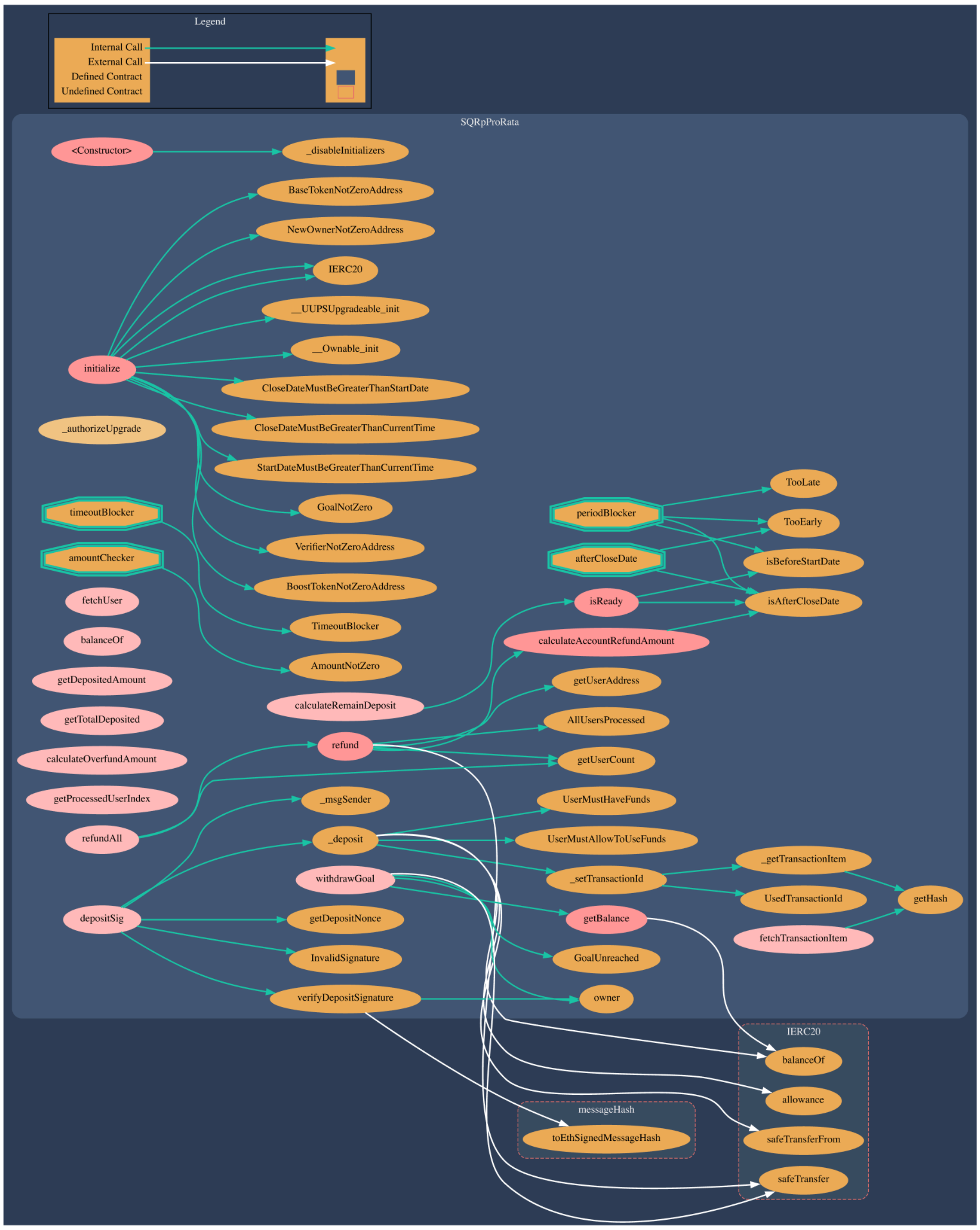
Vulnerabilities we have discovered are listed below.

Vulnerabilities found:

Severity	Amount
INFO	1
LOW	0
MEDIUM	0
HIGH	0
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

`getBalance()`
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

`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

	INFO
`withdrawGoal()`	
getBalance function that relies on contract token balance is used to determine whether the goal is reached or not. We would recommend using totalDeposited instead, since in case someone accidentally sends tokens to the contract, the goal will be reached, but totalDeposited may be lower than its value.	
TOKEN FLOW	Tokens Out, onlyOwner

Verification checksums

Contract name	Bytecode hash(SHA-256)
SQRpProRata.sol	e00db4e7b950a178b8abc3e9802c5743792654 a8a4e4a22f858ce40cbe068425