smart state

Web3 security easier than ever

MAGIC SQUARE
SQRPaymentGateway.sol
Smart contract audit report

April 10, 2024

Table of contents

Methodology

- 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 / Random number generators 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
- Re-entrancy
- 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

Summary

This audit encompasses the examination of the MAGIC SQUARE SQRPaymentGateway.sol smart contract.

Disclaimer

This is a final public security audit report version and does not include vulnerabilities that might have been found and addressed during the audit process.
An audit does not provide any warranties regarding the code security. We presume that a single audit cannot be considered totally sufficient and always recommend several independent audits and a public bug bounty program to ensure code security.
Please do not consider this report as investment and / or financial advice of any kind.

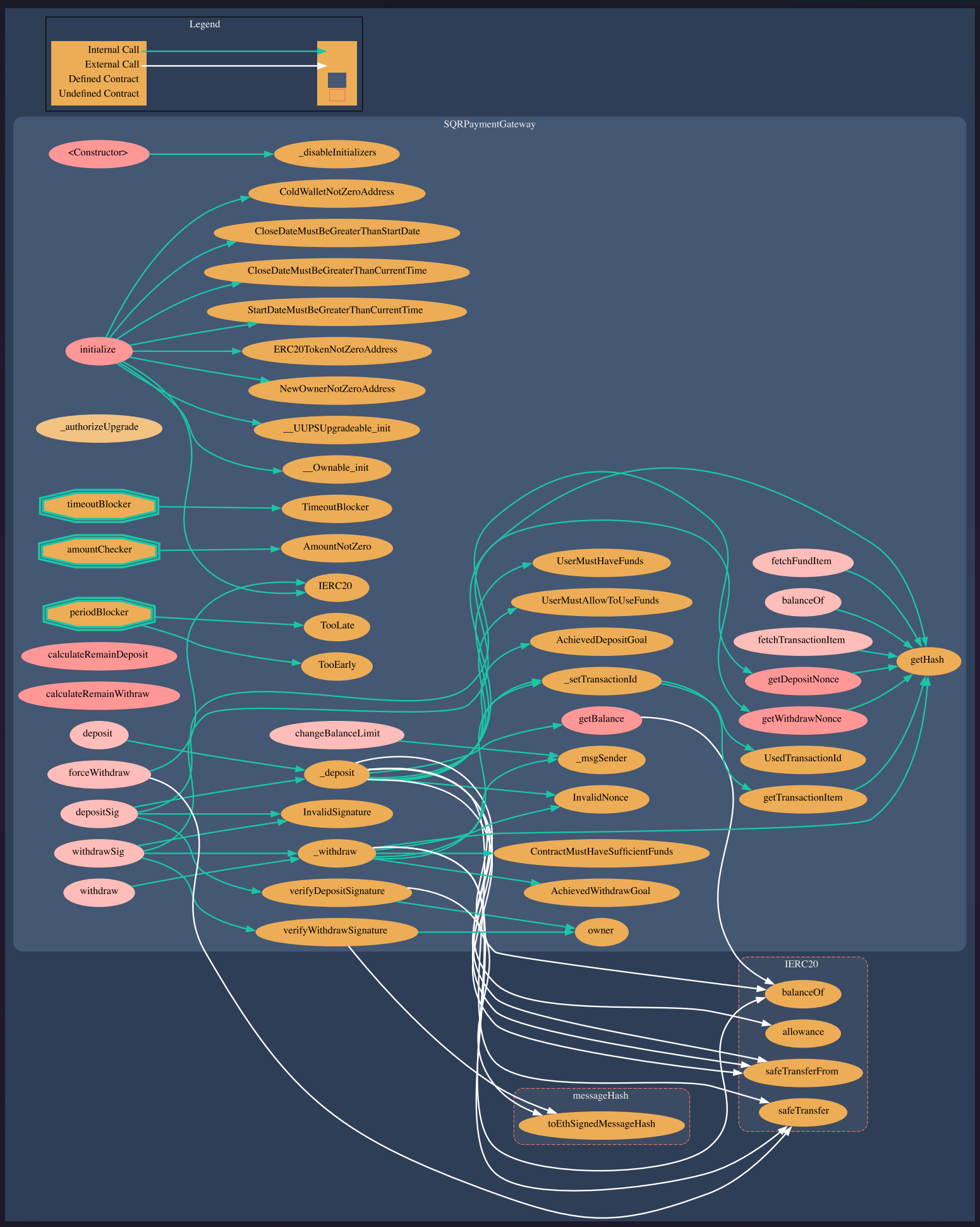
Vulnerabilities found by type

Info	0
Warning	0
Warning	0
Total	0

ACKNOWLEDGED	WARNING
High centralization risk	
There is a high centralization risk due to the contract structure.	

1.1 Structure of contract:

SQRPaymentGateway.sol



pic.1.1 SQRPaymentGateway.sol

1.2 SQRPaymentGateway.sol contract methods analysis:

initialize(address _newOwner, address _erc20Token, address _depositVerifier, uint256 _depositGoal, address _withdrawVerifier, uint256 _withdrawGoal, uint32 _startDate, uint32 _closeDate, address _coldWallet, uint256 _balanceLimit)
Vulnerabilities not detected

_authorizeUpgrade(address newImplementation)
Vulnerabilities not detected

changeBalanceLimit(uint256 _balanceLimit)
Vulnerabilities not detected

fetchFundItem(string memory userId)
Vulnerabilities not detected

getBalance()
Vulnerabilities not detected

balanceOf(string memory userId)
Vulnerabilities not detected

getHash(string memory value)
Vulnerabilities not detected

1.2 SQRPaymentGateway.sol contract methods analysis:

getDepositNonce(string memory userId)
Vulnerabilities not detected

getWithdrawNonce(string memory userId)
Vulnerabilities not detected

calculateRemainDeposit()
Vulnerabilities not detected

calculateRemainWithraw()
Vulnerabilities not detected

fetchTransactionItem(string memory transactionId)
Vulnerabilities not detected

getTransactionItem(string memory transactionId)
Vulnerabilities not detected

_setTransactionId(uint256 amount, string memory transactionId)
Vulnerabilities not detected

1.2 SQRPaymentGateway.sol contract methods analysis:

_deposit(string memory userId, string memory transactionId, address account, uint256 amount, uint32 nonce, uint32 timestampLimit)
Vulnerabilities not detected
TOKEN FLOW: Tokens in

deposit(string memory userId, string memory transactionId, address account, uint256 amount, uint32 nonce, uint32 timestampLimit)
Vulnerabilities not detected

verifyDepositSignature(string memory userId, string memory transactionId, address account, uint256 amount, uint32 nonce, uint32 timestampLimit, bytes memory signature)
Vulnerabilities not detected

depositSig(string memory userId, string memory transactionId, address account, uint256 amount, uint32 timestampLimit, bytes memory signature)
Vulnerabilities not detected

_withdraw(string memory userId, string memory transactionId, address to, uint256 amount, uint32 nonce, uint32 timestampLimit)
Vulnerabilities not detected
TOKEN FLOW: Tokens out

1.2 SQRPaymentGateway.sol contract methods analysis:

withdraw(string memory userId, string memory transactionId, address to, uint256 amount, uint32 nonce, uint32 timestampLimit)
Vulnerabilities not detected

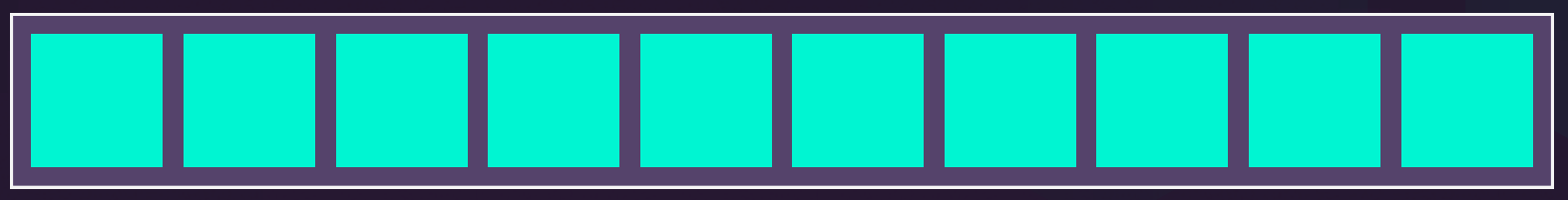
verifyWithdrawSignature(string memory userId, string memory transactionId, address to, uint256 amount, uint32 nonce, uint32 timestampLimit, bytes memory signature)
Vulnerabilities not detected

withdrawSig(string memory userId, string memory transactionId, address to, uint256 amount, uint32 timestampLimit, bytes memory signature)
Vulnerabilities not detected

Verification checksums

Contract name	Bytecode hash(SHA-256)
SQRPaymentGateway.sol	0268f3c26a9ab365d65fcbaae4d7512c4a33ab2aa293ef532c6106b36405ebb7

Project evaluation



10/10

Get in touch 👋

@smartstatetech

@smartstate

@SmartStateAudit

@smartstatetech

@smartstate.tech

View this report on Smartstate.tech

info@smartstate.tech

smartstate.tech