SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

Customer: Juice Protocol

Date: January 6th, 2022

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

Document

Name	Smart Contract Code Review and Security Analysis Report for Juice Protocol.
Approved by	Andrew Matiukhin | CTO Hacken OU
Type	Token, Governance, TimeLock, Defi, Strategies
Platform	Ethereum / Solidity
Methods	Architecture Review, Functional Testing, Computer-Aided Verification, Manual Review
Repository	https://github.com/jbx-protocol/juice-contracts-v1
Commit	3cc599068be591f0b2d0a1d125a4f8de9562de80
Timeline	20 DECEMBER 2021 - 6 JANUARY 2022
Changelog	6 JANUARY 2022 - Initial Audit

Table of contents

Introduction

Hacken OÜ (Consultant) was contracted by Juice Protocol (Customer) to conduct a Smart Contract Code Review and Security Analysis. This report presents the findings of the security assessment of Customer's smart contract and its code review conducted between December 20th 2021 - January 6th 2022.

Scope

The scope of the project is smart contracts in the repository:

Repository: https://github.com/jbx-protocol/juice-contracts-v1
Commit: 3cc599068be591f0b2d0a1d125a4f8de9562de80
File:

contract/Active3DaysFundingCycleBallot.sol
contract/Active7DaysFundingCycleBallot.sol
contract/Active14DaysFundingCycleBallot.sol
contract/DirectPaymentAddress.sol
contract/ExampleETHUSDPriceFeed.sol
contract/ExampleFailingFundingCycleBallot.sol
contract/ExampleJuiceboxProject.sol
contract/ExampleModAllocator.sol
contract/ExampleTreasuryExtension.sol
contract/ExampleYielder.sol
contract/FundingCycles.sol
contract/Governance.sol
contract/ModStore.sol
contract/OperatorStore.sol
contract/Prices.sol
contract/Projects.sol
contract/ProxyPaymentAddress.sol
contract/ProxyPaymentAddressManager.sol
contract/TerminalDirectory.sol
contract/TerminalV1.sol
contract/TicketBooth.sol
contract/Tickets.sol
contract/TokenRepresentationProxy.sol
contract/YearnYielder.sol

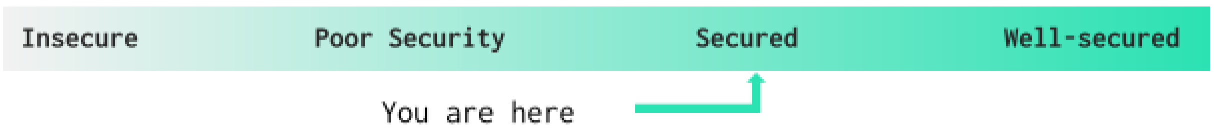
We have scanned this smart contract for commonly known and more specific vulnerabilities. Here are some of the commonly known vulnerabilities that are considered:

Category	Check Item
Code review	▪ Reentrancy ▪ Ownership Takeover

	▪ Timestamp Dependence ▪ Gas Limit and Loops ▪ DoS with (Unexpected) Throw ▪ DoS with Block Gas Limit ▪ Transaction-Ordering Dependence ▪ Style guide violation ▪ Costly Loop ▪ ERC20 API violation ▪ Unchecked external call ▪ Unchecked math ▪ Unsafe type inference ▪ Implicit visibility level ▪ Deployment Consistency ▪ Repository Consistency ▪ Data Consistency
Functional review	▪ Business Logics Review ▪ Functionality Checks ▪ Access Control & Authorization ▪ Escrow manipulation ▪ Token Supply manipulation ▪ Assets integrity ▪ User Balances manipulation ▪ Kill-Switch Mechanism ▪ Operation Trails & Event Generation

Executive Summary

According to the assessment, the Customer's smart contracts are secured.



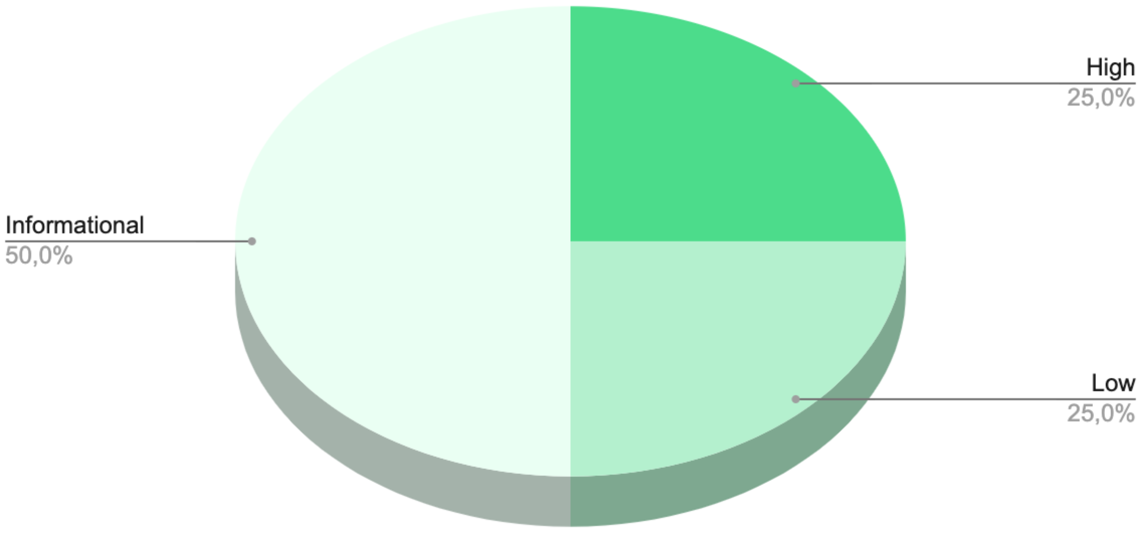
Our team performed an analysis of code functionality, manual audit, and automated checks with Mythril and Slither. All issues found during automated analysis were manually reviewed, and important vulnerabilities are presented in the Audit overview section. A general overview is presented in the AS-IS section, and all found issues can be found in the Audit overview section.

Security engineers found **1** high, **1** low and **2** informational severity issues during the first review.

Notice: Need to admit that the high issue is not a bug, it is a result of contract migration functional existing. But it provides a potential ability for illegal actions, it is protected by multisig governance and obviously it is not an easy and trivial way.

We are not responsible for QA, unit, and integration testing. But need to admit the good level of code test coverage. The code is well-documented: there are a lot of comments in the code and well-prepared documentation.

Graph 1. The distribution of vulnerabilities after the audit.



Severity Definitions

Risk Level	Description
Critical	Critical vulnerabilities are usually straightforward to exploit and can lead to assets loss or data manipulations.
High	High-level vulnerabilities are difficult to exploit; however, they also have a significant impact on smart contract execution, e.g., public access to crucial functions
Medium	Medium-level vulnerabilities are important to fix; however, they can't lead to assets loss or data manipulations.
Low	Low-level vulnerabilities are mostly related to outdated, unused, etc. code snippets that can't have a significant impact on execution
Lowest / Code Style / Best Practice	Lowest-level vulnerabilities, code style violations, and info statements can't affect smart contract execution and can be ignored.

AS-IS overview

Active3DaysFundingCycleBallot.sol

Description

Manages votes towards approving funding cycle reconfigurations.

Imports

- import "./interfaces/ITerminalV1.sol";
- import "./interfaces/IFundingCycleBallot.sol";

Inheritance

Active3DaysFundingCycleBallot has following inherited contracts:

- IFundingCycleBallot

Usages

Active3DaysFundingCycleBallot contract has no custom usages.

Structs

Active3DaysFundingCycleBallot contract has no data structures.

Enums

Active3DaysFundingCycleBallot contract has no enums.

Events

Active3DaysFundingCycleBallot contract has no events.

Modifiers

Active3DaysFundingCycleBallot has no custom modifiers.

Fields

Active3DaysFundingCycleBallot contract has following fields and constants:

- uint256 public constant reconfigurationDelay = 259200;

Functions

Active3DaysFundingCycleBallot has following public functions:

- duration
- state

Active7DaysFundingCycleBallot.sol

Description

Manages votes towards approving funding cycle reconfigurations.

Imports

- import "./interfaces/ITerminalV1.sol";
- import "./interfaces/IFundingCycleBallot.sol";

Inheritance

Active7DaysFundingCycleBallot has following inherited contracts:

- IFundingCycleBallot

Usages

Active7DaysFundingCycleBallot contract has no custom usages.

Structs

Active7DaysFundingCycleBallot contract has no data structures.

Enums

Active7DaysFundingCycleBallot contract has no enums.

Events

Active7DaysFundingCycleBallot contract has no events.

Modifiers

Active7DaysFundingCycleBallot has no custom modifiers.

Fields

Active7DaysFundingCycleBallot contract has following fields and constants:

- uint256 public constant reconfigurationDelay = 604800;

Functions

Active7DaysFundingCycleBallot has following public functions:

- duration
- state

Active14DaysFundingCycleBallot.sol

Description

Manages votes towards approving funding cycle reconfigurations.

Imports

- import "./interfaces/ITerminalV1.sol";
- import "./interfaces/IFundingCycleBallot.sol";

Inheritance

Active14DaysFundingCycleBallot has following inherited contracts:

- IFundingCycleBallot

Usages

Active14DaysFundingCycleBallot contract has no custom usages.

Structs

Active14DaysFundingCycleBallot contract has no data structures.

Enums

Active14DaysFundingCycleBallot contract has no enums.

Events

Active14DaysFundingCycleBallot contract has no events.

Modifiers

Active14DaysFundingCycleBallot has no custom modifiers.

Fields

Active14DaysFundingCycleBallot contract has following fields and constants:

- uint256 public constant reconfigurationDelay = 1209600;

Functions

Active14DaysFundingCycleBallot has following public functions:

- duration
- state

DirectPaymentAddress.sol

Description

A contract that can receive funds directly and forward to a project's current terminal.

Imports

- import "./interfaces/IDirectPaymentAddress.sol";
- import "./interfaces/ITerminalDirectory.sol";

Inheritance

DirectPaymentAddress has following inherited contracts:

- IDirectPaymentAddress

Usages

DirectPaymentAddress contract has no custom usages.

Structs

DirectPaymentAddress contract has no data structures.

Enums

DirectPaymentAddress contract has no enums.

Events

DirectPaymentAddress contract has no events.

Modifiers

DirectPaymentAddress has no custom modifiers.

Fields

DirectPaymentAddress contract has following fields and constants:

- ITerminalDirectory public immutable override terminalDirectory;
- uint256 public immutable override projectId;
- string public override memo;

Functions

DirectPaymentAddress has following public functions:

- terminalDirectory
- terminalDirectory
- memo

FundingCycles.sol

Description

Manage funding cycle configurations, accounting, and scheduling.

Imports

- import "@paulrberg/contracts/math/PRBMath.sol";
- import "./interfaces/IFundingCycles.sol";
- import "./interfaces/IPrices.sol";
- import "./abstract/TerminalUtility.sol";

Inheritance

FundingCycles has following inherited contracts:

- TerminalUtility
- IFundingCycles

Usages

FundingCycles contract has no custom usages.

Structs

FundingCycles contract has following data structures:

- FundingCycle
- FundingCycleProperties

Enums

FundingCycles contract has no enums.

Events

FundingCycles contract has following events:

- Configure
- Tap

- Init

Modifiers

FundingCycles has no custom modifiers.

Fields

FundingCycles contract has following fields and constants:

- uint256 private constant SECONDS_IN_DAY = 86400;
- mapping(uint256 => uint256) private _packedConfigurationPropertiesOf;
- mapping(uint256 => uint256) private _packedIntrinsicPropertiesOf;
- mapping(uint256 => uint256) private _metadataOf;
- mapping(uint256 => uint256) private _targetOf;
- mapping(uint256 => uint256) private _tappedOf;
- uint256 public constant override BASE_WEIGHT = 1E24;
- uint256 public constant override MAX_CYCLE_LIMIT = 32;
- mapping(uint256 => uint256) public override latestIdOf;
- uint256 public override count = 0;

Functions

FundingCycles has following public functions:

- latestIdOf
- count
- BASE_WEIGHT
- MAX_CYCLE_LIMIT
- get
- queuedOf
- currentOf
- currentBallotStateOf
- configure
- tap

Governance.sol

Description

Owner should eventually change to a multisig wallet contract.

Imports

- import "./interfaces/ITerminal.sol";
- import "./interfaces/IPrices.sol";
- import "./abstract/JuiceboxProject.sol";

Inheritance

Governance has following inherited contracts:

- JuiceboxProject

Fields

Governance contract has no fields and constants.

Functions

Governance has following public functions:

- allowMigration
- addPriceFeed
- setFee
- appointGovernance
- acceptGovernance

ModStore.sol

Description

Owner should eventually change to a multisig wallet contract.

Imports

- import "./interfaces/IModStore.sol";
- import "./abstract/Operatable.sol";
- import "./abstract/TerminalUtility.sol";

- import "./libraries/Operations.sol";

Inheritance

ModStore has following inherited contracts:

- IModStore
- Operatable
- TerminalUtility

Fields

ModStore contract has following fields and constants

- mapping(uint256 => mapping(uint256 => PayoutMod[])) private _payoutModsOf;
- mapping(uint256 => mapping(uint256 => TicketMod[])) private _ticketModsOf;
- IProjects public immutable override projects;

Functions

ModStore has following public functions:

- Projects
- payoutModsOf
- ticketModsOf
- setPayoutMods
- setTicketMods

OperatorStore.sol

Description

Addresses can give permissions to any other address to take specific actions throughout the Juicebox ecosystem on their behalf. These addresses are called `operators`.

Permissions are stored as a uint256, with each boolean bit representing whether or not an operator has the permission identified by that bit's index in the 256 bit uint256.

Indexes must be between 0 and 255.

Imports

- import "./interfaces/IOperatorStore.sol";

Inheritance

OperatorStore has following inherited contracts:

- IOperatorStore

Fields

OperatorStore contract has following fields and constants

- mapping(address => mapping(address => mapping(uint256 => uint256))) public override permissionsOf;

Functions

OperatorStore has following public functions:

- permissionsOf
- hasPermission
- hasPermissions
- setOperator
- setOperator
- setOperators

Prices.sol

Description

Manage and normalizes ETH price feeds.

Imports

- import "@openzeppelin/contracts/access/Ownable.sol";
- import "./interfaces/IPrices.sol";

Inheritance

Prices has following inherited contracts:

- IPrices
- Ownable

Fields

Prices contract has following fields and constants

- uint256 public constant override targetDecimals = 18;
- mapping(uint256 => uint256) public override feedDecimalAdjuster;
- mapping(uint256 => AggregatorV3Interface) public override feedFor;

Functions

Prices has following public functions:

- feedDecimalAdjuster
- targetDecimals
- feedFor
- getETHPriceFor
- addFeed

Projects.sol

Description

Stores project ownership and identifying information.

Imports

- import "@openzeppelin/contracts/token/ERC721/ERC721.sol";
- import "./abstract/Operatable.sol";
- import "./interfaces/IProjects.sol";
- import "./libraries/Operations.sol";

Inheritance

Projects has following inherited contracts:

- ERC721,
- IProjects,
- Operatable

Fields

Projects contract has following fields and constants

- uint256 private constant SECONDS_IN_YEAR = 31536000;
- uint256 public override count = 0;
- mapping(uint256 => string) public override uriOf;
- mapping(uint256 => bytes32) public override handleOf;
- mapping(bytes32 => uint256) public override projectFor;
- mapping(bytes32 => address) public override transferAddressFor;
- mapping(bytes32 => uint256) public override challengeExpiryOf;

Functions

Projects has following public functions:

- count
- uriOf
- handleOf
- projectFor
- transferAddressFor
- challengeExpiryOf
- exists
- create
- setHandle
- setUri
- transferHandle
- claimHandle

TerminalV1.sol

Description

This contract manages the Juicebox ecosystem, serves as a payment terminal, and custodies all funds.

Imports

- import '@openzeppelin/contracts/security/ReentrancyGuard.sol';

- import '@openzeppelin/contracts/utils/Address.sol';
- import '@paulrberg/contracts/math/PRBMath.sol';
- import '@paulrberg/contracts/math/PRBMathUD60x18.sol';
- import './interfaces/ITerminalV1.sol';
- import './abstract/JuiceboxProject.sol';
- import './abstract/Operatable.sol';
- import './libraries/Operations.sol';

Inheritance

TerminalV1 has following inherited contracts:

- Operatable,
- ITerminalV1,
- ITerminal,
- ReentrancyGuard

Fields

TerminalV1 contract has following fields and constants

- mapping(uint256 => int256) private _processedTicketTrackerOf;
- mapping(uint256 => uint256) private _preconfigureTicketCountOf;
- IProjects public immutable override projects;
- IFundingCycles public immutable override fundingCycles;
- ITicketBooth public immutable override ticketBooth;
- IModStore public immutable override modStore;
- IPrices public immutable override prices;
- ITerminalDirectory public immutable override terminalDirectory;
- mapping(uint256 => uint256) public override balanceOf;
- uint256 public override fee = 10;
- address payable public override governance;
- address payable public override pendingGovernance;

- mapping(ITerminal => bool) public override migrationIsAllowed;

Functions

TerminalV1 has following public functions:

- governance
- pendingGovernance
- projects
- fundingCycles
- ticketBooth
- prices
- modStore
- reservedTicketBalanceOf
- canPrintPreminedTickets
- balanceOf
- currentOverflowOf
- claimableOverflowOf
- fee
- deploy
- configure
- printPreminedTickets
- tap
- redeem
- printReservedTickets
- setFee
- appointGovernance
- acceptGovernance

Tickets.sol

Imports

- import "@openzeppelin/contracts/token/ERC20/ERC20.sol";
- import "@openzeppelin/contracts/token/ERC20/extensions/draft-ERC20Permit.sol";
- import "@paulrberg/contracts/token/erc20/Erc20Permit.sol";
- import "./interfaces/ITickets.sol";
- import "@openzeppelin/contracts/access/Ownable.sol";

Inheritance

Tickets has following inherited contracts:

- ERC20,
- ERC20Permit,
- Ownable,
- ITickets

Fields

Tickets contract has no fields and constants.

Functions

Tickets has following public functions:

- Print
- Redeem

YearnYielder.sol

Imports

- import "@openzeppelin/contracts/access/Ownable.sol";
- import "@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol";
- import "@paulrberg/contracts/math/PRBMath.sol";
- import "./interfaces/IYielder.sol";
- import "./interfaces/ITerminalV1.sol";
- import "./interfaces/IyVaultV2.sol";
- import "./interfaces/IWETH.sol";

Inheritance

YearnYielder has following inherited contracts:

- IYielder,
- Ownable

Fields

Tickets contract has following fields and constants:

- IyVaultV2 public wethVault = IyVaultV2(0xa9fE4601811213c340e850ea305481afF02f5b28);
- address public weth;
- uint256 public override deposited = 0;
- uint256 public decimals;

Functions

Tickets has following public functions:

- deposited
- getCurrentBalance
- deposit
- withdraw
- withdrawAll

Audit overview

🟥 🟥 🟥 🟥 Critical

No critical issues were found.

🟨 🟨 🟨 High

There is an ability for project owners to replace the TerminalV1 using the migration functionality. TerminalV1 (TerminalV1_1) contract is an implementation of the interface ITerminalV1 and responsible for ecosystem management. The project owners (or anyone who owns or has access to multisig addresses) are able to create their implementation of ITerminalV1 with any kind of logic they want, grant the migration allowance and execute the migration.
So that the owners can reconfigure the main management contract of the project, add logic mint/burn tickets, and control the project funds.
Need to admit that the high issue is not a bug, it is a result of contract migration functional existing. But it provides a potential ability for illegal actions, it is protected by multisig governance and obviously it is not an easy and trivial way.

🟩 🟩 Medium

No medium issues were found.

🟦 Low

There are a lot of cases with redundant gas consumption in the contracts code functions.
For example:

a. There are lots of redundant int overflow checks (solidity compiler version higher than 0.8 will check type overflow for math operations)
b. There are cases of variable initialization with default values (i.e. FundingCycles.sol)

■ Lowest / Code style / Best Practice

1. There is a set of events like: AddToBalance, AllowMigration, Migrate, Configure which don’t exist and don’t emit in the deployed contract: 0xd569D3CCE55b71a8a3f3C418c329A66e5f714431.
2. There is a set of events like Deposit and others which are declared in the ITerminal interface and never emit.

Conclusion

Smart contracts within the scope were manually reviewed and analyzed with static analysis tools. For the contract, high-level description of functionality was presented in As-Is overview section of the report. Audit report contains all found security vulnerabilities and other issues in the reviewed code.

Security engineers found **1** high, **1** low and **2** informational severity issues during the first review.

Notice: Need to admit that the high issue is not a bug, it is a result of contract migration functional existing. But it provides a potential ability for illegal actions, it is protected by multisig governance and obviously it is not an easy and trivial way.

We are not responsible for QA, unit, and integration testing. But need to admit the good level of code test coverage. The code is well-documented: there are a lot of comments in the code and well-prepared documentation.

Disclaimers

Hacken Disclaimer

The smart contracts given for audit have been analyzed in accordance with the best industry practices at the date of this report, in relation to cybersecurity vulnerabilities and issues in smart contract source code, the details of which are disclosed in this report (Source Code); the Source Code compilation, deployment, and functionality (performing the intended functions).

The audit makes no statements or warranties on security of the code. It also cannot be considered as a sufficient assessment regarding the utility and safety of the code, bugfree status or any other statements of the contract. While we have done our best in conducting the analysis and producing this report, it is important to note that you should not rely on this report only - we recommend proceeding with several independent audits and a public bug bounty program to ensure security of smart contracts.

Technical Disclaimer

Smart contracts are deployed and executed on blockchain platform. The platform, its programming language, and other software related to the smart contract can have its vulnerabilities that can lead to hacks. Thus, the audit can't guarantee the explicit security of the audited smart contracts.