HACKEN

SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

Customer: Startbahn Inc
Date: August 17th, 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 Startbahn Inc
Approved By	Evgeniy Bezuglyi | SC Audits Department Head at Hacken OU
Type	ERC721 token
Platform	EVM
Network	Polygon network
Language	Solidity
Methods	Manual Review, Automated Review, Architecture Review
Website	https://startrail.io
Timeline	20.06.2022 - 17.08.2022
Changelog	21.07.2022 - Initial Review 17.08.2022 - Second Review

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

Table of contents

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

Introduction

Hacken OÜ (Consultant) was contracted by Startbahn Inc (Customer) to conduct a Smart Contract Code Review and Security Analysis. This report presents the findings of the security assessment of the Customer's smart contracts.

Scope

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

Initial review scope
Repository:
https://github.com/startbahn/startrail
Commit:
99fe4ed8122e697dc07b737cdaebfa075f948542
Technical Documentation:
Type: Whitepaper (partial functional requirements provided)
https://whitepaper.startrail.io/

Type: Technical description
https://github.com/startbahn/startrail/tree/develop/docs

Integration and Unit Tests: Yes
Contracts:
File: ./contracts/bulk/Bulk.sol
SHA3: bb326e1347d83c8a7bdf8307ea4ab1523c9e5fafb56aadf0aa41d5343321355a

File: ./contracts/common/SignatureDecoder.sol
SHA3: dee31a995f76cccca739118c9b830befe4f35609c8c01d64f5ea4cc8e96e681a

File: ./contracts/lib/IDGeneratorV2.sol
SHA3: b85b45ff1294354c801d390df11eaec9561be9038c283ae24e7af6d8da2db921

File: ./contracts/lib/MultiSend.sol
SHA3: 1ab67c8e81ffec8dc63773b244bb81793ee2869467fc0e788c581bea4ba34a08

File: ./contracts/licensedUser/LicensedUserManager.sol
SHA3: afab5eaec0d181df1db6cf5ba1afd2f21cc1bb9df782c9dc25d267fb812986d6

File: ./contracts/licensedUser/OwnerManager.sol
SHA3: 20ec416a0bca62a2e5bb34d60c43f3c2899a0c0c77b5a69c67c146c4c9c8c331

File: ./contracts/licensedUser/proxy/AdminUpgradeabilityProxyLUM.sol
SHA3: 44a76a009f14dabc4d948aa8591151d5d5f9ea2ada6aef4c8bb837febdfeb1d9

File: ./contracts/licensedUser/proxy/StartrailProxyLUM.sol
SHA3: a9c7842464d22528b24612d0659719a451d59c34d74aaec7e9189645c7f971d8

File: ./contracts/licensedUser/proxy/WalletProxyMinimal.sol
SHA3: cf91cbf262f184473370cbf3a4374628e56aa335676cb8b926e2bf076e36eab1

```
File: ./contracts/licensedUser/SignatureChecker.sol
SHA3: 8c3fc86c3f298ef97084a4173b265d3d72dca8816a872a3b008d510f74030581

File: ./contracts/metaTx/eip2771/EIP2771BaseRecipient.sol
SHA3: 87cecd3c1a96d1c2b4c5e038d60c96c30dcaee53f772d37b79b3f0726c545238

File: ./contracts/metaTx/MetaTxForwarderV2.sol
SHA3: bea046d63b35377f32498e94613c32c86f8a88dd09b9c72a4c7c9144e103a7df

File: ./contracts/metaTx/MetaTxRequestManager.sol
SHA3: 9c8e7e7446b51aea56092316003a5dd6ac3c8b779a5652347e2fbe3f976f95f7

File: ./contracts/metaTx/replayProtection/ReplayProtection.sol
SHA3: e3deda5b7ab623256d6114f131d5a665b79d4227dee956f0e199a7dbbe4e9e7c

File: ./contracts/metaTx/TransactionExecutor.sol
SHA3: 92544c62e9edc2a2cdd536f2ef5cc4035d5f47ada65e1f713111eadf1d975902

File: ./contracts/name/Contracts.sol
SHA3: 38263058577aeb287482e9ee25461d97a0d6b51aa8e80d36f3efd0f7c54be47f

File: ./contracts/name/NameRegistry.sol
SHA3: f796778863ad88c7540a9f4d360ae5120890d1f6931157fb093e75ea248ed4e3

File: ./contracts/proxyAdmin/AdminUpgradeabilityProxy.sol
SHA3: 4c3752f17b77e304056fc97bda714bf6c6094cd473dbfd4ed08066bab4dca5b2

File: ./contracts/proxyAdmin/BaseAdminUpgradeabilityProxy.sol
SHA3: 5138e1b3ffe200daa79d3deaeb32a1f0db7fb0a53bdea14833f614658fec5117

File: ./contracts/proxyAdmin/BaseUpgradeabilityProxy.sol
SHA3: 69dcbcd30dd9c88237b843c4b8e91a027780f4bdc12ae9e2ffd33c0a6c112d75

File: ./contracts/proxyAdmin/Ownable.sol
SHA3: 34a05780e303c0b203dfbee7a169416500ab681e4d675f62820bcdc21cc06dc3

File: ./contracts/proxyAdmin/Proxy.sol
SHA3: 478ff05478604699116d754f1af900228d0c6b87ea0a81ec2c1bfaa3ddcf8657

File: ./contracts/proxyAdmin/ProxyAdmin.sol
SHA3: 54656f872d6c23c8062b450833dcd97fdffda6ac0a4b019734453239bdb90bb6

File: ./contracts/proxyAdmin/StartrailProxy.sol
SHA3: e2cae87a5b3845e092c40e9072c8411d8ce477a794b659420b89e38d25bc2090

File: ./contracts/proxyAdmin/StartrailProxyAdmin.sol
SHA3: b44818d2dd998c57bdb1c917f5d1bf8417e668d9b4dd4fb033ed2ba249b76771

File: ./contracts/proxyAdmin/UpgradeabilityProxy.sol
SHA3: 89043c500865115b1f066cf9291244f61381637fe55f08a837561120c5dec89f

File: ./contracts/proxyAdmin/UpgradeableProxiable.sol
SHA3: 6fd8b84245693c4ecf4fbcbeb279f9df1331dcdfbc18c6ec1004b52ce870edc9
```

File: ./contracts/startrailregistry/ERC721.sol
SHA3: bc1ea1fc220aab23343e72510c0d69b2ae4b02d37ed04e916c5f571ca0285e65

File: ./contracts/startrailregistry/OpenSeaMetaTransactionLibrary.sol
SHA3: 805f134005579aa728f237d51e55a5810fe02d298ae30b48ab00527b211730f6

File: ./contracts/startrailregistry/StartrailRegistryLibraryV1.sol
SHA3: a2215283c616f2aec80dd56b84727971d1b6e3ad76c6387737157ab582af001b

File: ./contracts/startrailregistry/StartrailRegistryV15.sol
SHA3: 45e03d8617b05ca252532b2546ee6b910049720eb039a8f0f32af993d8339fe9

File: ./contracts/startrailregistry/Storage.sol
SHA3: 05403bad4e489fc5ac248f1ec6b56762f9a8a140a87c6f4d881993a349187125

Second review scope

Repository:
https://github.com/startbahn/startrail/tree/develop

Commit:
9911089c1c912c9dfbcfa9873ee8990853128698

Technical Documentation:
Type: Whitepaper (partial functional requirements provided)
https://whitepaper.startrail.io/

Type: Technical description
https://github.com/startbahn/startrail/tree/develop/docs

Integration and Unit Tests: Yes

Contracts:
File: ./contracts/bulk/Bulk.sol
SHA3: bb326e1347d83c8a7bdf8307ea4ab1523c9e5fafb56aadf0aa41d5343321355a

File: ./contracts/common/SignatureDecoder.sol
SHA3: dee31a995f76cccca739118c9b830befe4f35609c8c01d64f5ea4cc8e96e681a

File: ./contracts/lib/IDGeneratorV2.sol
SHA3: b85b45ff1294354c801d390df11eaec9561be9038c283ae24e7af6d8da2db921

File: ./contracts/lib/MultiSend.sol
SHA3: 1ab67c8e81ffec8dc63773b244bb81793ee2869467fc0e788c581bea4ba34a08

File: ./contracts/licensedUser/LicensedUserManager.sol
SHA3: 5819d07a3edd54c617314457842709cdc443b4baca98c33163eabb6d525bf042

File: ./contracts/licensedUser/OwnerManager.sol
SHA3: 20ec416a0bca62a2e5bb34d60c43f3c2899a0c0c77b5a69c67c146c4c9c8c331

File: ./contracts/licensedUser/proxy/AdminUpgradeabilityProxyLUM.sol
SHA3: 44a76a009f14dabc4d948aa8591151d5d5f9ea2ada6aef4c8bb837febdfeb1d9

File: ./contracts/licensedUser/proxy/StartrailProxyLUM.sol
SHA3: a9c7842464d22528b24612d0659719a451d59c34d74aaec7e9189645c7f971d8

File: ./contracts/licensedUser/proxy/WalletProxyMinimal.sol
SHA3: cf91cbf262f184473370cbf3a4374628e56aa335676cb8b926e2bf076e36eab1

File: ./contracts/licensedUser/SignatureChecker.sol
SHA3: 8c3fc86c3f298ef97084a4173b265d3d72dca8816a872a3b008d510f74030581

File: ./contracts/metaTx/eip2771/EIP2771BaseRecipient.sol
SHA3: 3c365bfbc47be13e47ec6d15fbd998a9019c11ee9ea2c3ede074c77115a9fa64

File: ./contracts/metaTx/MetaTxForwarderV2.sol
SHA3: bea046d63b35377f32498e94613c32c86f8a88dd09b9c72a4c7c9144e103a7df

File: ./contracts/metaTx/MetaTxRequestManager.sol
SHA3: 9c8e7e7446b51aea56092316003a5dd6ac3c8b779a5652347e2fbe3f976f95f7

File: ./contracts/metaTx/replayProtection/ReplayProtection.sol
SHA3: e3deda5b7ab623256d6114f131d5a665b79d4227dee956f0e199a7dbbe4e9e7c

File: ./contracts/metaTx/TransactionExecutor.sol
SHA3: 92544c62e9edc2a2cdd536f2ef5cc4035d5f47ada65e1f713111eadf1d975902

File: ./contracts/name/Contracts.sol
SHA3: 38263058577aeb287482e9ee25461d97a0d6b51aa8e80d36f3efd0f7c54be47f

File: ./contracts/name/NameRegistry.sol
SHA3: f796778863ad88c7540a9f4d360ae5120890d1f6931157fb093e75ea248ed4e3

File: ./contracts/proxyAdmin/AdminUpgradeabilityProxy.sol
SHA3: 4c3752f17b77e304056fc97bda714bf6c6094cd473dbfd4ed08066bab4dca5b2

File: ./contracts/proxyAdmin/BaseAdminUpgradeabilityProxy.sol
SHA3: 5138e1b3ffe200daa79d3deaeb32a1f0db7fb0a53bdea14833f614658fec5117

File: ./contracts/proxyAdmin/BaseUpgradeabilityProxy.sol
SHA3: 69dcbcd30dd9c88237b843c4b8e91a027780f4bdc12ae9e2ffd33c0a6c112d75

File: ./contracts/proxyAdmin/Ownable.sol
SHA3: 34a05780e303c0b203dfbee7a169416500ab681e4d675f62820bcdc21cc06dc3

File: ./contracts/proxyAdmin/Proxy.sol
SHA3: 478ff05478604699116d754f1af900228d0c6b87ea0a81ec2c1bfaa3ddcf8657

File: ./contracts/proxyAdmin/ProxyAdmin.sol
SHA3: 54656f872d6c23c8062b450833dcd97fdffda6ac0a4b019734453239bdb90bb6

File: ./contracts/proxyAdmin/StartrailProxy.sol
SHA3: e2cae87a5b3845e092c40e9072c8411d8ce477a794b659420b89e38d25bc2090

File: ./contracts/proxyAdmin/StartrailProxyAdmin.sol
SHA3: b44818d2dd998c57bdb1c917f5d1bf8417e668d9b4dd4fb033ed2ba249b76771

File: ./contracts/proxyAdmin/UpgradeabilityProxy.sol
SHA3: 89043c500865115b1f066cf9291244f61381637fe55f08a837561120c5dec89f

File: ./contracts/proxyAdmin/UpgradeableProxiable.sol
SHA3: 6fd8b84245693c4ecf4fbcbeb279f9df1331dcdfbc18c6ec1004b52ce870edc9

File: ./contracts/startrailregistry/ERC721.sol
SHA3: bc1ea1fc220aab23343e72510c0d69b2ae4b02d37ed04e916c5f571ca0285e65

File: ./contracts/startrailregistry/OpenSeaMetaTransactionLibrary.sol
SHA3: 805f134005579aa728f237d51e55a5810fe02d298ae30b48ab00527b211730f6

```
File: ./contracts/startrailregistry/StartrailRegistryLibraryV1.sol
SHA3: a2215283c616f2aec80dd56b84727971d1b6e3ad76c6387737157ab582af001b

File: ./contracts/startrailregistry/StartrailRegistryV16.sol
SHA3: 65be09d138df227dfbce93818985843e629a31de9d32c0c04332acfbb06e873b

File: ./contracts/startrailregistry/Storage.sol
SHA3: 05403bad4e489fc5ac248f1ec6b56762f9a8a140a87c6f4d881993a349187125
```

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

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 cannot lead to assets loss or data manipulations.
Low	Low-level vulnerabilities are mostly related to outdated, unused, etc. code snippets that cannot have a significant impact on execution.

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

Executive Summary

The score measurement details can be found in the corresponding section of the methodology.

Documentation quality

The total Documentation Quality score is **10** out of **10**. The Customer provided a whitepaper. Technical and functional requirements are provided.

Code quality

The total CodeQuality score is **9** out of **10**. The code is covered with unit tests and followed by comments. The code contains some library code duplications.

Architecture quality

The architecture quality score is **10** out of **10**. The project architecture follows conventional practices.

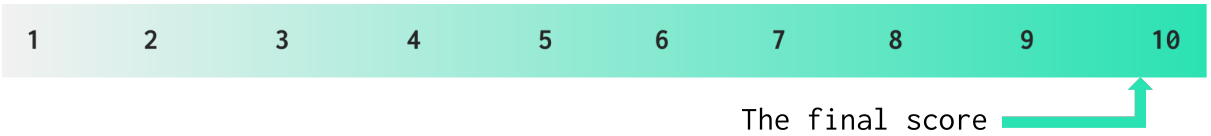
Security score

As a result of the audit, the code contains **3** low severity issues. The security score is **10** out of **10**.

All found issues are displayed in the “Findings” section.

Summary

According to the assessment, the Customer's smart contract has the following score: **9.9**.



Checked Items

We have audited provided smart contracts for commonly known and more specific vulnerabilities. Here are some of the items that are considered:

Item	Type	Description	Status
Default Visibility	SWC-100 SWC-108	Functions and state variables visibility should be set explicitly. Visibility levels should be specified consciously.	Passed
Integer Overflow and Underflow	SWC-101	If unchecked math is used, all math operations should be safe from overflows and underflows.	Passed
Outdated Compiler Version	SWC-102	It is recommended to use a recent version of the Solidity compiler.	Failed
Floating Pragma	SWC-103	Contracts should be deployed with the same compiler version and flags that they have been tested thoroughly.	Passed
Unchecked Call Return Value	SWC-104	The return value of a message call should be checked.	Passed
Access Control & Authorization	CWE-284	Ownership takeover should not be possible. All crucial functions should be protected. Users could not affect data that belongs to other users.	Passed
SELFDESTRUCT Instruction	SWC-106	The contract should not be self-destructible while it has funds belonging to users.	Not Relevant
Check-Effect-Interaction	SWC-107	Check-Effect-Interaction pattern should be followed if the code performs ANY external call.	Passed
Assert Violation	SWC-110	Properly functioning code should never reach a failing assert statement.	Not Relevant
Deprecated Solidity Functions	SWC-111	Deprecated built-in functions should never be used.	Passed
Delegatecall to Untrusted Callee	SWC-112	Delegatecalls should only be allowed to trusted addresses.	Not Relevant
DoS (Denial of Service)	SWC-113 SWC-128	Execution of the code should never be blocked by a specific contract state unless it is required.	Passed
Race Conditions	SWC-114	Race Conditions and Transactions Order Dependency should not be possible.	Not Relevant
Authorization	SWC-115	tx.origin should not be used for	Passed

through tx.origin		authorization.	
Block values as a proxy for time	SWC-116	Block numbers should not be used for time calculations.	Not Relevant
Signature Unique Id	SWC-117 SWC-121 SWC-122 EIP-155	Signed messages should always have a unique id. A transaction hash should not be used as a unique id. Chain identifier should always be used. All parameters from the signature should be used in signer recovery	Passed
Shadowing State Variable	SWC-119	State variables should not be shadowed.	Passed
Weak Sources of Randomness	SWC-120	Random values should never be generated from Chain Attributes or be predictable.	Not Relevant
Incorrect Inheritance Order	SWC-125	When inheriting multiple contracts, especially if they have identical functions, a developer should carefully specify inheritance in the correct order.	Passed
Calls Only to Trusted Addresses	EEA-Level-2 SWC-126	All external calls should be performed only to trusted addresses.	Passed
Presence of unused variables	SWC-131	The code should not contain unused variables if this is not justified by design.	Passed
EIP standards violation	EIP	EIP standards should not be violated.	Passed
Assets integrity	**Custom**	Funds are protected and cannot be withdrawn without proper permissions.	Passed
User Balances manipulation	**Custom**	Contract owners or any other third party should not be able to access funds belonging to users.	Passed
Data Consistency	**Custom**	Smart contract data should be consistent all over the data flow.	Passed
Flashloan Attack	**Custom**	When working with exchange rates, they should be received from a trusted source and not be vulnerable to short-term rate changes that can be achieved by using flash loans. Oracles should be used.	Not Relevant
Token Supply manipulation	**Custom**	Tokens can be minted only according to rules specified in a whitepaper or any other documentation provided by the customer.	Passed
Gas Limit and Loops	**Custom**	Transaction execution costs should not depend dramatically on the amount of	Passed

		data stored on the contract. There should not be any cases when execution fails due to the block Gas limit.	
Style guide violation	**Custom**	Style guides and best practices should be followed.	Passed
Requirements Compliance	**Custom**	The code should be compliant with the requirements provided by the Customer.	Passed
Environment Consistency	**Custom**	The project should contain a configured development environment with a comprehensive description of how to compile, build and deploy the code.	Passed
Secure Oracles Usage	**Custom**	The code should have the ability to pause specific data feeds that it relies on. This should be done to protect a contract from compromised oracles.	Not Relevant
Tests Coverage	**Custom**	The code should be covered with unit tests. Test coverage should be 100%, with both negative and positive cases covered. Usage of contracts by multiple users should be tested.	Passed
Stable Imports	**Custom**	The code should not reference draft contracts, that may be changed in the future.	Passed

System Overview

Startrail is a blockchain infrastructure supporting the development of the fine arts ecosystem by furthering its trust and credibility. It is designed and managed by Startbahn:

- *StartrailRegistry* – this is the main ERC-721 NFT contract where the SRRs are registered. Startrail Registry Record (SRR) certificate stores such information:
  - Basic artwork information
  - Provenance data
  - Detailed information to enrich artwork authenticity
- *NameRegistry* – basic contract configuration which stores the addresses of the contracts in the system.
- *Administrator* – the contract responsible for administrators functions:
  - Contract upgrades
  - Licensed User creation and maintenance
  - Meta transaction request type maintenance
  - Contract level configuration property maintenance
- *LicensedUserManager* – is a contract which manages Licensed User wallets. A Licensed User wallet is a single address controlled by one or more owners. The owners register with the Administrator as known and KYC'd entities. This contract manages all Licensed User wallets with a level of security equal to a proxy with bytecode.
- *MetaTxForwarder* – this contract verifies and forwards meta transactions to other Startrail contracts Startrail meta transaction types are defined in the MetaTxRequestManager. Transactions are forwarded if they are one of these types.
- *Bulk* – the contract performs the bulk of actions. The Merkle root of a tree of actions to be performed in bulk is stored with the contract, as are hashes of leaves that have been processed. Leaf actions are submitted through the Bulk contract, which verifies they are part of the Merkle tree before forwarding them to an appropriate Startrail destination contract.

Privileged roles

- The *Administrator* controls the system of the *Startrail* contracts. Admin is authorized to update contracts in the system, create new SRRs and update existing SRRs information, add Licensed Users.
- The *Licensed User* is able to issue new SRRs and control previously created tokens.
- The *Licensed User Wallet* owner participates in governance of the Licensed User entity.
- An *EOA* (Externally Owned Account) specifically, on Startrail, an EOA is an account type for artwork owners. An EOA has access to SRRs it owns and can trigger transfers.

Risks

- The contracts in the system are upgradable.
- Admin may issue new tokens and update SRR (ERC-721) token information.
- Admin and token issuer may lock all the transfers for a specific token.

Findings

Critical

No critical severity issues were found.

High

No high severity issues were found.

Medium

1. **Invalid event data.**

   The function emits the *UpdateSRR* event and passes msgSender as a third parameter to the SRR data structure. The third parameter should be an issuer address, but the function may be called by the admin of the contract, which means that the wrong value may be passed.

   This may confuse users with the wrong data specified during the event emitting.

   File: ./contracts/startrailregistry/StartrailRegistryV16.sol

   Contract: StartrailRegistryV16

   Function: updateSRR

   Recommendation: Specify the correct data during the event emitting.

   Status: Fixed
   (Revised commit: 21aab76a3325952503f7fb77d2222486ebd0890c)

Low

1. **Typical library function declaration.**

   The contract has the typical *isEmptyString* math function that may be loaded from the library.

   This may lead to unnecessary Gas usage during the contract deployment.

   File: ./contracts/licensedUser/OwnerManager.sol

   Contract: OwnerManager

   Function: isEmptyString

   Recommendation: Use the library function.

   Status: Reported

2. **Redundant nesting level.**

   The contract has the state mapping *_addressStorage* with the redundant nesting level. Only _addressStorage[tokenId][_SRR] is passed, which means that the second nesting level is redundant.

This may lead to redundant Gas usage.

File: ./contracts/startrailregistry/Storage.sol

Contract: Storage

Function: isEmptyString

Recommendation: Remove redundant nesting level.

Status: Reported

3. **Missing zero address validation.**

   The owner address parameter is not validated if a zero owner address is specified in the single item array. Address parameters are being used without checking against the possibility of *0x0*.

   This can lead to useless Gas usage during the creation of a wallet or can lead to unwanted external calls to *0x0*

   Files:
   ./contracts/startrailregistry/StartrailRegistryV16.sol
   ./contracts/metaTx/eip2771/EIP2771BaseRecipient.sol

   Functions: setNameRegistryAddress, _setTrustedForwarder

   File: ./contracts/licensedUser/LicensedUserManager.sol

   Contract: LicensedUserManager

   Function: createWalletInternal

   Recommendation: Implement zero address validations.

   Status: Fixed
   (Revised commit: 8b3312cfd931b59359561fc5eb1f73fb8f4f322d)

4. **Comparisons of boolean values.**

   Boolean constants can be used directly and do not need to be compared with *true* or *false*.

   File: ./contracts/licensedUser/LicensedUserManager.sol

   Functions: onlyActiveWallet, isActiveWallet

   File: ./contracts/startrailregistry/StartrailRegistryV16.sol

   Function: onlyLicensedUserOrAdministrator

   Recommendation: Remove the equality to the boolean constant.

   Status: Fixed
   (Revised commit: 6d09e065e026fd3aff0edffe160b665d30148464)

5. **Potential Out-of-Gas exception.**

Iterating over arrays to create tokens in `createSRRWithProofMulti` function may lead to enormous Gas consumption due to the arrays' size.

File: ./contracts/bulk/Bulk.sol

Contract: Bulk

Function: createSRRWithProofMulti

Recommendation: Implement array size limitations.

Status: Mitigated (Array size limitations are implemented in the off-chain logic. The Startrail-api server's restriction is set to 50 records per call for the functions. Additionally, the relayed tx Gas limit is set at 12M, which is more than enough to process the 50 records).

6. **Outdated Solidity version.**

Using an old version prevents access to new Solidity security checks.

Recommendation: Consider using one of these versions: *0.8.6, 0.8.9, 0.8.11*, or 0.8.13.

Status: Reported

Disclaimers

Hacken Disclaimer

The smart contracts given for audit have been analyzed by the best industry practices at the date of this report, with 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 report contains no statements or warranties on the identification of all vulnerabilities and security of the code. The report covers the code submitted to and reviewed, so it may not be relevant after any modifications. Do not consider this report as a final and sufficient assessment regarding the utility and safety of the code, bug-free status, or any other contract statements.

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 the security of smart contracts.

English is the original language of the report. The Consultant is not responsible for the correctness of the translated versions.

Technical Disclaimer

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