AliumSwap

SMART CONTRACT AUDIT

22.03.2021

Made in Germany by Chainsulting.de

Table of contents

1. Disclaimer

The audit makes no statements or warrantees about utility of the code, safety of the code, suitability of the business model, investment advice, endorsement of the platform or its products, regulatory regime for the business model, or any other statements about fitness of the contracts to purpose, or their bug free status. The audit documentation is for discussion purposes only.

The information presented in this report is confidential and privileged. If you are reading this report, you agree to keep it confidential, not to copy, disclose or disseminate without the agreement of AliumSwap. If you are not the intended receptor of this document, remember that any disclosure, copying or dissemination of it is forbidden.

Major Versions / Date	Description
0.1 (15.03.2021)	Layout
0.5 (16.03.2021)	Verify Claims and Test Deployment
0.6 (17.03.2021)	Testing SWC Checks
0.8 (17.03.2021)	Automated Security Testing Manual Security Testing
0.9 (18.03.2021)	Summary and Recommendation
1.0 (19.03.2021)	Final document
1.1 (22.03.2021)	Added fixed commits
1.2 (23.03.2021)	Added deployed contracts

2. About the Project and Company

Company address: NA (ANON)

Website: http://alium.finance

Twitter: http://twitter.com/alium.finance

Medium: https://aliumswap.medium.com

Telegram (ENG): https://t.me/aliumswap_official

Telegram (RU): https://t.me/aliumswap_ru

Telegram (JP): https://t.me/aliumswap_jp

Reddit: https://www.reddit.com/user/AliumSwap_Official

Discord: https://discord.gg/BU6m6zgpvZ

LinkedIn: https://www.linkedin.com/company/75861509

2.1 Project Overview

AliumSwap is a decentralized AMM Exchange with multi-blockchain option and NFTs. Aliumswap claims to be the first AMM DEX with multi-chain support, which starts on the Binance Smart Chain blockchain (BSC). BSC is very similar in its user properties to Ethereum, but the fees are currently hundreds of times lower.

AMM DEX Alium (AliumSwap) allows users to reduce the costs of commissions, they have improved and added some features that are not available on other exchanges on Binance Smart Chain. For example, they allow the user the ability to create any trading pair on their own, without unnecessary coordination with the exchange support.

3. Vulnerability & Risk Level

Risk represents the probability that a certain source-threat will exploit vulnerability, and the impact of that event on the organization or system. Risk Level is computed based on CVSS version 3.0.

Level	Value	Vulnerability	Risk (Required Action)
Critical	9 – 10	A vulnerability that can disrupt the contract functioning in a number of scenarios, or creates a risk that the contract may be broken.	Immediate action to reduce risk level.
High	7 – 8.9	A vulnerability that affects the desired outcome when using a contract, or provides the opportunity to use a contract in an unintended way.	Implementation of corrective actions as soon as possible.
Medium	4 – 6.9	A vulnerability that could affect the desired outcome of executing the contract in a specific scenario.	Implementation of corrective actions in a certain period.
Low	2 – 3.9	A vulnerability that does not have a significant impact on possible scenarios for the use of the contract and is probably subjective.	Implementation of certain corrective actions or accepting the risk.
Informational	0 – 1.9	A vulnerability that have informational character but is not effecting any of the code.	An observation that does not determine a level of risk

4. Auditing Strategy and Techniques Applied

Throughout the review process, care was taken to evaluate the repository for security-related issues, code quality, and adherence to specification and best practices. To do so, reviewed line-by-line by our team of expert pentesters and smart contract developers, documenting any issues as there were discovered.

4.1 Methodology

The auditing process follows a routine series of steps:

1. Code review that includes the following:
   i. Review of the specifications, sources, and instructions provided to Chainsulting to make sure we understand the size, scope, and functionality of the smart contract.
   ii. Manual review of code, which is the process of reading source code line-by-line in an attempt to identify potential vulnerabilities.
   iii. Comparison to specification, which is the process of checking whether the code does what the specifications, sources, and instructions provided to Chainsulting describe.
2. Testing and automated analysis that includes the following:
   i. Test coverage analysis, which is the process of determining whether the test cases are actually covering the code and how much code is exercised when we run those test cases.
   ii. Symbolic execution, which is analysing a program to determine what inputs causes each part of a program to execute.
3. Best practices review, which is a review of the smart contracts to improve efficiency, effectiveness, clarify, maintainability, security, and control based on the established industry and academic practices, recommendations, and research.
4. Specific, itemized, actionable recommendations to help you take steps to secure your smart contracts.

4.2 Used Code from other Frameworks/Smart Contracts (direct imports)

Dependency / Import Path	Source
@openzeppelin/contracts/access/AccessControl.sol	https://github.com/OpenZeppelin/openzeppelin-contracts/tree/v3.4.0/contracts/access/AccessControl.sol
@openzeppelin/contracts/access/Ownable.sol	https://github.com/OpenZeppelin/openzeppelin-contracts/tree/v3.4.0/contracts/access/Ownable.sol
@openzeppelin/contracts/token/ERC20/IERC20.sol	https://github.com/OpenZeppelin/openzeppelin-contracts/tree/v3.4.0/contracts/token/ERC20/IERC20.sol
@openzeppelin/contracts/token/ERC20/SafeERC20.sol	https://github.com/OpenZeppelin/openzeppelin-contracts/tree/v3.4.0/contracts/token/ERC20/SafeERC20.sol
@openzeppelin/contracts/token/ERC721/ERC721.sol	https://github.com/OpenZeppelin/openzeppelin-contracts/tree/v3.4.0/contracts/ERC721/ERC721.sol
@openzeppelin/contracts/token/ERC721/IERC721Receiver.sol	https://github.com/OpenZeppelin/openzeppelin-contracts/tree/v3.4.0/contracts/ERC721/IERC721Receiver.sol
@openzeppelin/contracts/utils/Counters.sol	https://github.com/OpenZeppelin/openzeppelin-contracts/tree/v3.4.0/contracts/utils/Counters.sol
@openzeppelin/contracts/math/SafeMath.sol	https://github.com/OpenZeppelin/openzeppelin-contracts/tree/v3.4.0/contracts/math/SafeMath.sol
Whitelist.sol	https://github.com/HQ20/contracts/blob/6a4f166ca8ae0789955a33a0175edfa2dcb4b69f/contracts/access/Whitelist.sol

4.3 Tested Contract Files

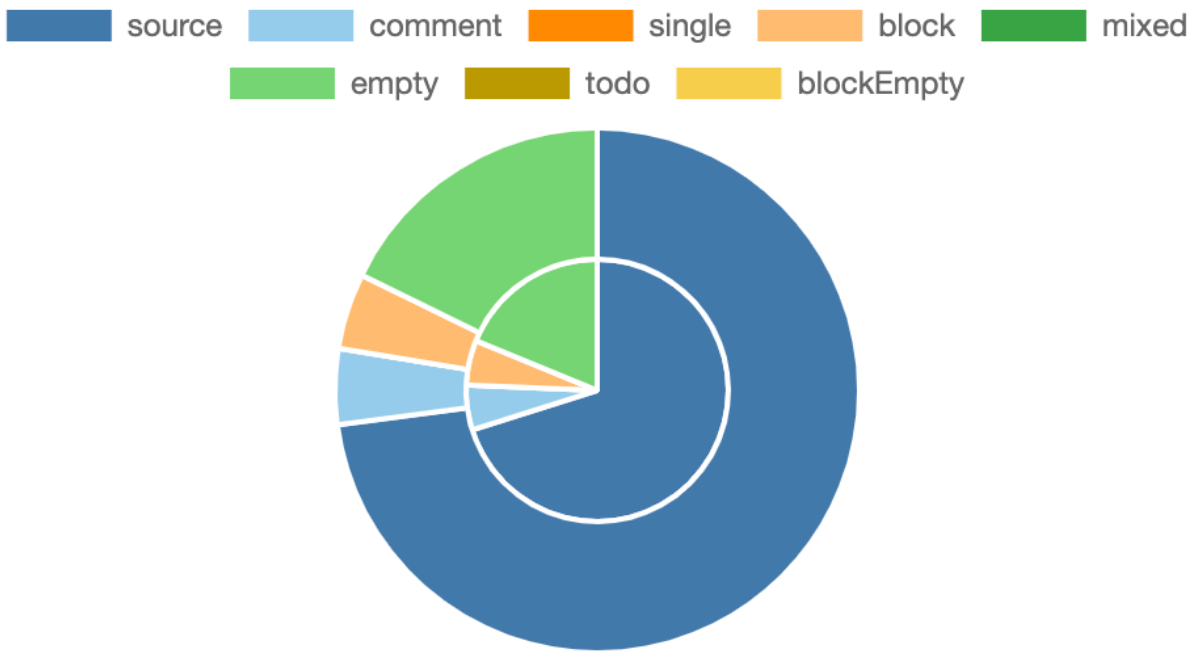
The following are the MD5 hashes of the reviewed files. A file with a different MD5 hash has been modified, intentionally or otherwise, after the security review. You are cautioned that a different MD5 hash could be (but is not necessarily) an indication of a changed condition or potential vulnerability that was not within the scope of the review

File	Fingerprint (MD5)
AliumCollectible.sol	0689a8a1214ba28e45e437ab9d19f4f1
AliumCollectionMintable.sol	8817fcc598447a182fec2a7a176b8983
IAliumCollectible.sol	e1d9f59f9e74a9dbb59b8b92a613a6f7
IERC20Optional.sol	41b6568ec414a00e2946f3a486c60f9a
NFTPrivateSeller.sol	8295bb0a9481e1f6eefad3d0d68f48e4
Whitelist.sol	f58ed7fdd5c62855f19f7d9d46708bd7

4.4 Metrics / CallGraph

Full version: http://chainsulting.de/wp-content/uploads/2021/03/aliumswap_solidity-metrics.html

4.5 Metrics / Source Lines



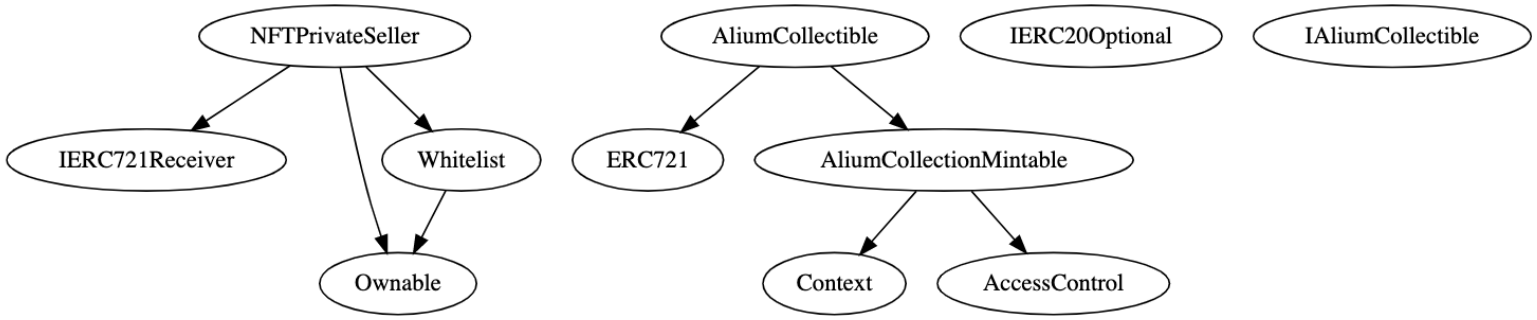
4.6 Metrics / Capabilities

Solidity Versions observed	Experimental Features	Can Receive Funds	Uses Assembly	Has Destroyable Contracts
`=0.6.2`			**** (0 asm blocks)	

Transfers ETH	Low-Level Calls	DelegateCall	Uses Hash Functions	ECRecover	New/Create/Create2
`yes`			`yes`		

Public	Payable
37	0

External	Internal	Private	Pure	View
15	40	0	0	9



4.7 Metrics / Source Unites in Scope

File	Logic Contracts	Interfaces	Lines	nLines	nSLOC	Comment Lines	Complex. Score	Capabilities
contracts/NFTPrivateSeller.sol	1		208	185	148		114	
contracts/AliumCollectible.sol	1		161	140	110		74	
contracts/IERC20Optional.sol		1	5	4	3		3	
contracts/Whitelist.sol	1		72	72	35	25	33	
contracts/IAliumCollectible.sol		1	42	4	3		23	
contracts/AliumCollectionMintable.sol	1		22	22	18		15	
Totals	**4**	**2**	**510**	**427**	**317**	**25**	**262**	

Legend: [-]

- **Lines**: total lines of the source unit
- **nLines**: normalized lines of the source unit (e.g. normalizes functions spanning multiple lines)
- **nSLOC**: normalized source lines of code (only source-code lines; no comments, no blank lines)
- **Comment Lines**: lines containing single or block comments
- **Complexity Score**: a custom complexity score derived from code statements that are known to introduce code complexity (branches, loops, calls, external interfaces, ...)

5. Scope of Work

The AliumSwap Team provided us with the files that needs to be tested. The scope of the audit is the AliumSwap NFT Collectible contract.

Following contracts with the direct imports has been tested:

- AliumCollectible.sol
- NFTPrivateSeller.sol

The team put forward the following assumptions regarding the security, usage of the contracts:

- A user can buy exact amount of nft cards from one pool at a time, i.e. if a user pays for 5 cards from pool 2 he gets 5 cards from pool 2.
- A user is charged for the exact amount of fee and in selected currency only, i.e. if a user buys 5 cards from pool 3 and has selected to pay in USDT he is charged for 5 cards x 15 000 USDT = 75 000 USDT.
- A user cannot buy more cards than are left in a pool, i.e. if there are 3 cards left in pool 1 a user can't buy 4 or more cards.
- The fee is sent only to the Founders address and can't be changed by anyone except the owner.
- Bought cards are sent to the buyer's address and can't be intercepted in the process.
- Owner rights are transferrable by the current owner.
- The owner can edit the Whitelist (can add and remove addresses that are allowed to make purchases).
- Only the owner is allowed to edit the Whitelist.
- Overall smart contract security needs to be checked

The main goal of this audit was to verify these claims. The auditors can provide additional feedback on the code upon the client's request.

5.1 Manual and Automated Vulnerability Test

CRITICAL ISSUES
During the audit, Chainsulting‘s experts found **no Critical issues** in the code of the smart contract.

HIGH ISSUES
During the audit, Chainsulting’s experts found **no High issues** in the code of the smart contract.

MEDIUM ISSUES
During the audit, Chainsulting’s experts found **no Medium issues** in the code of the smart contract.

LOW ISSUES

5.1.1 Same require checks in every function
Severity: LOW
Status: FIXED
File(s) affected: AliumCollectible.sol

Attack / Description	Code Snippet	Result/Recommendation
Same require checks in every function.	Line: 63 – 66, 80-83, 93-96	It is recommended to use modifiers. For admin roll checks, there is already „onlyAdmin“ modifier (used in line 145), which should be used.

INFORMATIONAL ISSUES

5.1.2 Missing natspec documentation
Severity: INFORMATIONAL
Status: FIXED
File(s) affected: NFTPrivateSeller.sol, AliumCollectible.sol

Attack / Description	Code Snippet	Result/Recommendation
Solidity contracts can use a special form of comments to provide rich documentation for functions, return variables and more. This special form is named the Ethereum Natural Language Specification Format (NatSpec).	NA	It is recommended to include natspec documentation and follow the doxygen style including @author, @title, @notice, @dev, @param, @return and make it easier to review and understand your smart contract. The team addressed the issue while auditing and added more documentation parts. https://github.com/1inch-exchange/mooniswap-v2/pull/13/commits

5.2. SWC Attacks & Special Checks

ID	Title	Relationships	Test Result
SWC-131	Presence of unused variables	CWE-1164: Irrelevant Code	✅
SWC-130	Right-To-Left-Override control character (U+202E)	CWE-451: User Interface (UI) Misrepresentation of Critical Information	✅
SWC-129	Typographical Error	CWE-480: Use of Incorrect Operator	✅
SWC-128	DoS With Block Gas Limit	CWE-400: Uncontrolled Resource Consumption	✅
SWC-127	Arbitrary Jump with Function Type Variable	CWE-695: Use of Low-Level Functionality	✅
SWC-125	Incorrect Inheritance Order	CWE-696: Incorrect Behavior Order	✅
SWC-124	Write to Arbitrary Storage Location	CWE-123: Write-what-where Condition	✅
SWC-123	Requirement Violation	CWE-573: Improper Following of Specification by Caller	✅

ID	Title	Relationships	Test Result
SWC-122	Lack of Proper Signature Verification	CWE-345: Insufficient Verification of Data Authenticity	✅
SWC-121	Missing Protection against Signature Replay Attacks	CWE-347: Improper Verification of Cryptographic Signature	✅
SWC-120	Weak Sources of Randomness from Chain Attributes	CWE-330: Use of Insufficiently Random Values	✅
SWC-119	Shadowing State Variables	CWE-710: Improper Adherence to Coding Standards	✅
SWC-118	Incorrect Constructor Name	CWE-665: Improper Initialization	✅
SWC-117	Signature Malleability	CWE-347: Improper Verification of Cryptographic Signature	✅
SWC-116	Timestamp Dependence	CWE-829: Inclusion of Functionality from Untrusted Control Sphere	✅
SWC-115	Authorization through tx.origin	CWE-477: Use of Obsolete Function	✅
SWC-114	Transaction Order Dependence	CWE-362: Concurrent Execution using Shared Resource with Improper Synchronization ('Race Condition')	✅

ID	Title	Relationships	Test Result
SWC-113	DoS with Failed Call	CWE-703: Improper Check or Handling of Exceptional Conditions	✅
SWC-112	Delegatecall to Untrusted Callee	CWE-829: Inclusion of Functionality from Untrusted Control Sphere	✅
SWC-111	Use of Deprecated Solidity Functions	CWE-477: Use of Obsolete Function	✅
SWC-110	Assert Violation	CWE-670: Always-Incorrect Control Flow Implementation	✅
SWC-109	Uninitialized Storage Pointer	CWE-824: Access of Uninitialized Pointer	✅
SWC-108	State Variable Default Visibility	CWE-710: Improper Adherence to Coding Standards	✅
SWC-107	Reentrancy	CWE-841: Improper Enforcement of Behavioral Workflow	✅
SWC-106	Unprotected SELFDESTRUCT Instruction	CWE-284: Improper Access Control	✅
SWC-105	Unprotected Ether Withdrawal	CWE-284: Improper Access Control	✅
SWC-104	Unchecked Call Return Value	CWE-252: Unchecked Return Value	✅

ID	Title	Relationships	Test Result
SWC-103	Floating Pragma	CWE-664: Improper Control of a Resource Through its Lifetime	✅
SWC-102	Outdated Compiler Version	CWE-937: Using Components with Known Vulnerabilities	✅
SWC-101	Integer Overflow and Underflow	CWE-682: Incorrect Calculation	✅
SWC-100	Function Default Visibility	CWE-710: Improper Adherence to Coding Standards	✅

7. Test Deployment

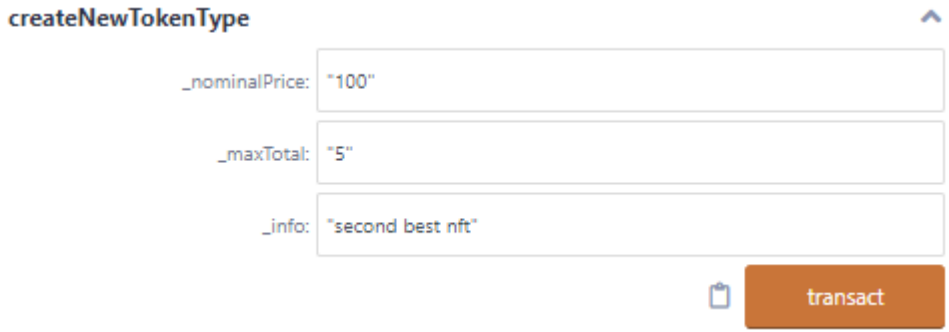
7.1 Deployment AliumCollectible Contract

Tx: https://kovan.etherscan.io/tx/0xb8fa5be59e0244da6f06065c159766e89cc07ca2e04b9378505ca17dd7bb5394
Contract: https://kovan.etherscan.io/address/0x02eda4046af868744c7f6240ee1e810f7fac173c

7.2 Deployment DAI Contract

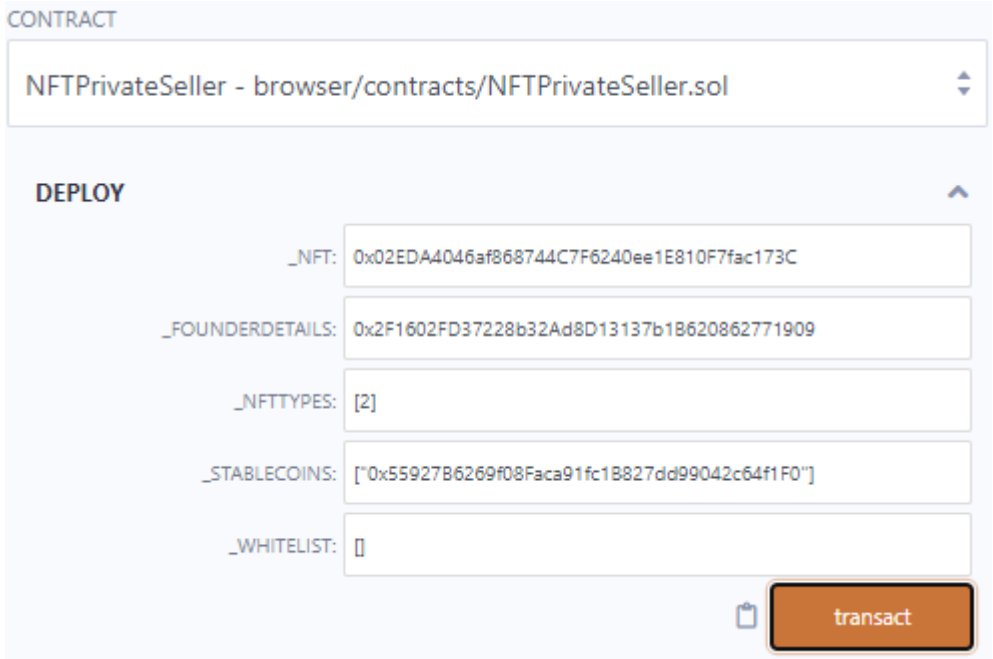
Tx: https://kovan.etherscan.io/tx/0x27d7cd468ad5ddcf52407e332c6d6e6cbe98327e0609678a62781f3ebbd911cb
Contract: https://kovan.etherscan.io/address/0x55927b6269f08faca91fc1b827dd99042c64f1f0

7.3 Create new token type



Tx: https://kovan.etherscan.io/tx/0xdd24ca9c5b6f2ebf176388112733c56bf0954ae7334b3c4309fa0f36779f7b48

7.4 Deployment NFTPrivateSeller Contract



Tx: https://kovan.etherscan.io/tx/0xd705db130e90c0c27d4d981ab8835789f6f96e202b1ea0bc7c0d8c04cc66b0e0
Contract: https://kovan.etherscan.io/address/0xc6f29eb74d39631bb056baff50860cc8c26681ac

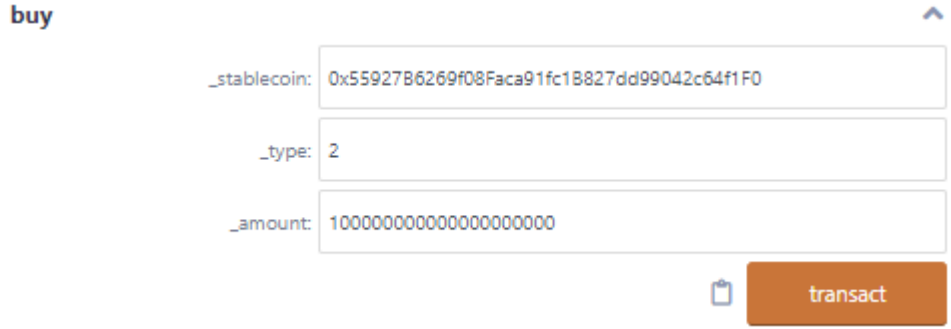
7.5 Add buyer to whitelist (only by owner)

The owner and only the owner can edit the whitelist of NFTPrivateSeller.

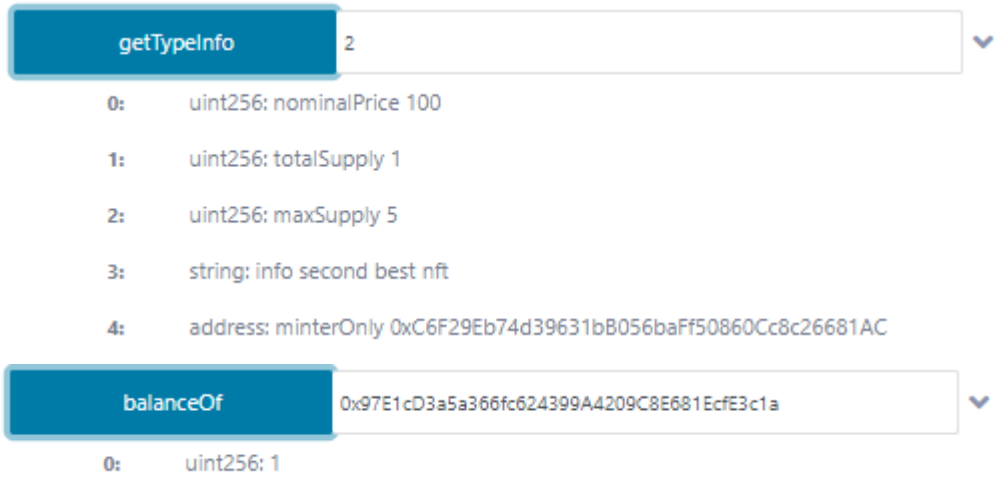
Tx: https://kovan.etherscan.io/tx/0xe728aec499461c8519c0fc46087797a3cccc4cd34d6f6b9c8c781fe6cf80a9b0

7.6 Buy one token with whitelisted address

Only whitelisted addresses are able to buy tokens. The total supply increases correctly by buying. The purchased item is mapped correctly to the buyers address and is send to the correct address. The purchased value is sent directly to the founder.

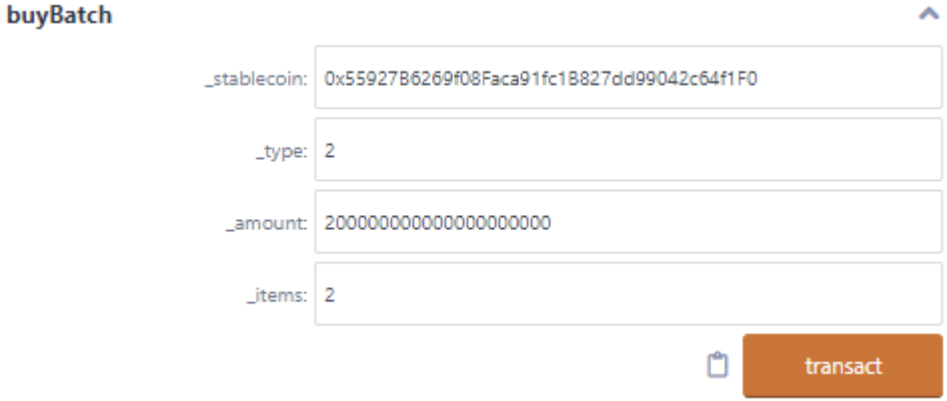


Tx: https://kovan.etherscan.io/tx/0x012593216d605708b27aeeb8b0d57f0584bc49fdd9f9ac390c2e66c8d5950ff0



7.7 Buy Batch

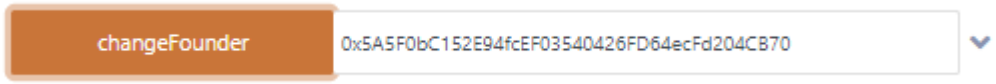
Buy batch is working as expected. The price is purchased to the founder and the correct amount of tokens is send to the buyer.



Tx: https://kovan.etherscan.io/tx/0x90e9d277925dcce23b8a43ee38be41b737469a89703d8f7fbd2e5a1f3f2a4911

7.8 Change founder by owner

Only the owner is able to change the founder of NFTPrivateSeller.

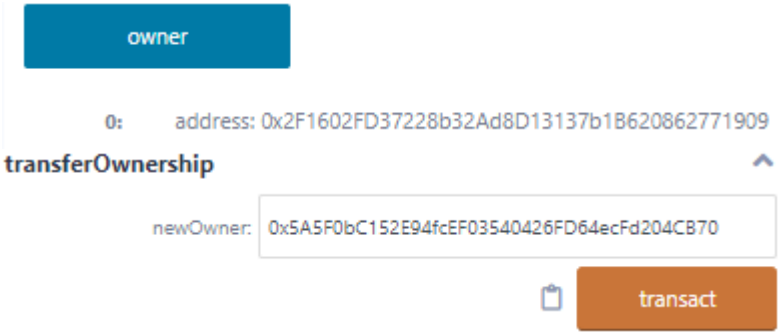


Tx: https://kovan.etherscan.io/tx/0xfc696ac21815e215942ab551bfa55561d7e7d957d719456f2a7da8905b4e577b

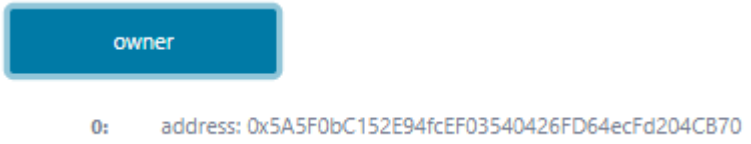
After buying again, the purchased value is sent to the new founder:
Tx: https://kovan.etherscan.io/tx/0x969c96a8ac4ec099ef6c868ccb3ad12be36ab031569a384fd259827f9cff15b2

7.9 Transfer ownership of NFTPrivateSeller

The ownership is transferable by the current owner.



Tx: https://kovan.etherscan.io/tx/0x74efa42d1bfb813f28bc4acc623357e4bc2f01c341c3ce325d5b4d8d47c74d7d



7.10 Unit Test

```
NFTPrivateSeller
  Management
    √ should change founder (721ms)
    √ should repair tokens (1525ms)
  Sell
    √ should buy 1 nft token type 1 (2124ms)
    √ should buy batch 2 nft token type 2 (2592ms)
    √ should fail on batch buy 2'nd nft token type 1 (collection type limit 1 item) (1111ms)
    √ should fail on buy 2'nd nft token type 1 (collection type limit 1 item) (2117ms)


6 passing (11s)
```

8. Verify claims

8.1 A user can buy exact amount of nft cards from one pool at a time, i.e. if a user pays for 5 cards from pool 2 he gets 5 cards from pool 2
Status: tested and verified ✅

8.2 A user is charged for the exact amount of fee and in selected currency only, i.e. if a user buys 5 cards from pool 3 and has selected to pay in USDT he is charged for 5 cards x 15 000 USDT = 75 000 USDT.
Status: tested and verified ✅

8.3 A user cannot buy more cards than are left in a pool, i.e. if there are 3 cards left in pool 1 a user can't buy 4 or more cards.
Status: tested and verified ✅

8.4 The fee is sent only to the Founders address and can't be changed by anyone except the owner.
Status: tested and verified ✅

8.5 Bought cards are sent to the buyer's address and can't be intercepted in the process.
Status: tested and verified ✅

8.6 Owner rights are transferrable by the current owner.
Status: tested and verified ✅

8.7 The owner can edit the Whitelist (can add and remove addresses that are allowed to make purchases).
Status: tested and verified ✅

8.8 Only the owner is allowed to edit the Whitelist.
Status: tested and verified ✅

9. Executive Summary

Two (2) independent Chainsulting experts performed an unbiased and isolated audit of the smart contract codebase. The overall code quality of the project is very good, and the simplicity greatly benefits the overall security. It correctly implemented widely-used and reviewed contracts from OpenZeppelin and for safe mathematical operations. It is recommended to include natspec documentation and follow the doxygen style including @author, @title, @notice, @dev, @param, @return and make it easier to review and understand your smart contract.

The main goal of the audit was to verify the claims regarding the security of the smart contract and the functions. During the audit, no critical issues were found after the manual and automated security testing.

10. Deployed Smart Contract

VERIFIED

Smart Contract is deployed here:
Alium NFT
https://bscscan.com/address/0x2991cc4aB9286416b7925916aE6bD2Dc5AF7bAcb#code

NFT private seller
https://bscscan.com/address/0xcde039E55eaFf293e40085a1BccEBdaDb445d626#code