ScanSandBox™

Security Assessment

AMM Coin

Mar. 05 2022

ScanSanBox™

Hi Everyone :

First of all, I would like to give my heartfelt thanks to all the people who have ever helped me in this paper. My sincere and hearty thanks and appreciations to our team, they have done a good job with their expertise, which is impressive. In the entire block chain industry, security has always been the focus of attention, it's nice to see more and more people attaches great importance to the security of smart contracts, ScanSandBox has a strong experienced technical team, they support our security audit services. For a long time, ScanSandBox's customers around the world to provide the best security services, has won widespread praise, We maintain a professional efficient and responsible attitude and will continue to provide the best security services.

Finally, congratulations to the AMM Coin team, this contract has passed the ScanSandBox security audit.

Thanks and wish you all the best.

Calvin Paul

Mar 05 2022

+(1) 2535339548

Service@ScanSandBox.com

15117 Main St. Suite 205 Mill Creek WA

Table of Contents

Summary

This report has been prepared for AMM Coin to discover issues and vulnerabilities in the source code of the AMM Coin project as well as any contract dependencies that were not part of an officially recognized library. A comprehensive examination has been performed, utilizing Static Analysis and Manual Review techniques.

The auditing process pays special attention to the following considerations:

- Testing the smart contracts against both common and uncommon attack vectors. Assessing the codebase to ensure compliance with current best practices and industry standards.
- Ensuring contract logic meets the specifications and intentions of the client.
- Cross referencing contract structure and implementation against similar smart contracts produced by industry leaders.
- Thorough line-by-line manual review of the entire codebase by industry experts.

The security assessment resulted in findings that ranged from critical to informational. We recommend addressing these findings to ensure a high level of security standards and industry practices. We suggest recommendations that could better serve the project from the security perspective:

- Enhance general coding practices for better structures of source codes;
- Add enough unit tests to cover the possible use cases given they are currently missing in the repository;
- Provide more comments per each function for readability, especially contracts are verified in public;
- Provide more transparency on privileged activities once the protocol is live.

Overview

Project Summary

Project Name	AMM Coin
Description	AMM Coin is a leveraged yield farming protocol based on Binance Smart Chain (BSC) released by AMM Coin Finance Lab.
Platform	BSC
Language	Solidity
Codebase	https://bscscan.com/address/0xa2ce2b0d961005ae2ada4d886643edc3b9a11704#code
Commit	

Audit Summary

Delivery Date	Feb 28, 2022
Audit Methodology	Static Analysis, Manual Review

Vulnerability Summary

Category	Total	Pending	Partially Resolved	Resolved	Acknowledged	Declined
Security	6	0	0	6	0	0
Economy	4	0	0	4	0	0
ERC standard	1	0	1	0	0	0
Lint	0	0	0	0	0	0
Miscellaneous	5	0	0	0	5	0
Discussion	0	0	0	0	0	0

Audit Scope

ID	File	SHA256 Checksum
01	AMM.sol	01634180f7dde9435847375d58850b6ca6938fa1a00fe7a3667c993add048711

Centralization Roles

The smart contract introduces several authorizations. After deployment, the owner would call the function renounceOwnership to renounce the ownership to improve the trustworthiness of this protocol.

Governance Check :

[AMM.sol] NOT FOUND

Findings

SEC-01 Could potentially lead to re-entrancy vulnerability

Category	Severity	Location	Status
Re-entrancy	Security	AMM.sol : 133	Resolved

Description

Any interaction from a contract (A) with another contract (B) and any transfer of Ether hands over control to that contract (B). This makes it possible for B to call back into A before this interaction is completed.

Potential violation of Checks-Effects-Interaction pattern in Address._function Call With Value (address, bytes, uint256, string): Could potentially lead to re-entrancy vulnerability.

Recommendation

We recommend Take control of your contract execution process. Alleviation.

SEC-02 Inline assembly

Category	Severity	Location	Status
Inline assembly	Security	Amm.sol :104	Resolved

Description

The Contract uses inline assembly, this is only advised in rare cases. Additionally static analysis modules do not parse inline Assembly, this can lead to wrong analysis results.
Inline assembly is a way to access the Ethereum Virtual Machine at a low level. This bypasses several important safety features and checks of Solidity.

Recommendation

We recommend only use it for tasks that need it, and only if you are confident with using it.

Alleviation

The AMM Coin team heeded our advice and ignore this warning.

SEC-03 Inline assembly

Category	Severity	Location	Status
Inline assembly	Security	AMM.sol :143	Resolved

Description

The Contract uses inline assembly, this is only advised in rare cases. Additionally static analysis modules do not parse inline Assembly, this can lead to wrong analysis results.

Inline assembly is a way to access the Ethereum Virtual Machine at a low level. This bypasses several important safety features and checks of Solidity.

Recommendation

We recommend only use it for tasks that need it, and only if you are confident with using it.

Alleviation

The AMM Coin team heeded our advice and ignore this warning.

SEC-04 Old compiler version declaration and version not locked

Category	Severity	Location	Status
Language Specific	Security	AMM.sol : 1	Resolved

Description

solc frequently releases new compiler versions. Using an old version prevents access to new Solidity security checks.

The contract uses some different versions, such as pragma solidity ^0.8.0; these are not locked. This is not recommended. Pragmas should be locked to specific compiler versions and flags that they have been tested the most with. Locking the pragma helps ensure that contracts do not accidentally get deployed using, for example, the latest compiler, which may have higher risks of undiscovered bugs.

Recommendation

Avoid a floating pragma version (i.e. pragma solidity ^0.8.0;) instead specify pragma version without using the caret symbol, i.e., pragma solidity 0.8.11;.
We recommend using the latest version of Solidity for testing.

Alleviation

The AMM Coin team heeded our advice and changed the latest version.

SEC-05 Low level calls

Category	Severity	Location	Status
Low Level	Security	AMM.sol : 112	Resolved

Description

A contract can decide how much of its remaining gas should be sent with the inner message call and how much it wants to retain. If an out-of-gas exception happens in the inner call (or any other exception), this will be signaled by an error value put onto the stack. In this case, only the gas sent together with the call is used up. In Solidity, the calling contract causes a manual exception by default in such situations, so that exceptions “bubble up” the call stack.

Use of "call": should be avoided whenever AMM.sol sible. It can lead to unexpected behavior if return value is not handled properly. Please use Direct Calls via specifying the called contract's interface.

Recommendation

We recommend that acknowledged this finding.

Alleviation

The AMM Coin team has acknowledged this finding.

SEC-06 Low level calls

Category	Severity	Location	Status
Low Level	Security	AMM.sol : 137	Resolved

Description

A contract can decide how much of its remaining gas should be sent with the inner message call and how much it wants to retain. If an out-of-gas exception happens in the inner call (or any other exception), this will be signaled by an error value put onto the stack. In this case, only the gas sent together with the call is used up. In Solidity, the calling contract causes a manual exception by default in such situations, so that exceptions “bubble up” the call stack.

Use of "call": should be avoided whenever AMM.sol sible. It can lead to unexpected behavior if return value is not handled properly. Please use Direct Calls via specifying the called contract's interface.

Recommendation

We recommend that acknowledged this finding.

Alleviation

The AMM Coin team has acknowledged this finding.

ECO-01 Gas requirement is infinite

Category	Severity	Location	Status
Gas Consume	Economy	AMM.sol : 183	Resolved

Description

Gas requirement of function AMM.name is infinite:
If the gas requirement of a function is higher than the block gas limit, it cannot be executed. Please avoid loops in your functions or actions that modify large areas of storage (this includes clearing or copying arrays in storage).

Loops that do not have a fixed number of iterations, for example, loops that depend on storage values, have to be used carefully: Due to the block gas limit, transactions can only consume a certain amount of gas. Either explicitly or just due to normal operation, the number of iterations in a loop can grow beyond the block gas limit which can cause the complete contract to be stalled at a certain point. This may not apply to view functions that are only executed to read data from the blockchain. Still, such functions may be called by other contracts as part of on-chain operations and stall those. Please be explicit about such cases in the documentation of your contracts.

Recommendation

We recommend that function name return String only.

Alleviation

The AMM Coin team was clearly warned.

ECO-02 Gas requirement is infinite

Category	Severity	Location	Status
Gas Consume	Economy	AMM.sol : 187	Resolved

Description

Gas requirement of function AMM.symbol is infinite:
If the gas requirement of a function is higher than the block gas limit, it cannot be executed. Please avoid loops in your functions or actions that modify large areas of storage (this includes clearing or copying arrays in storage).

Loops that do not have a fixed number of iterations, for example, loops that depend on storage values, have to be used carefully: Due to the block gas limit, transactions can only consume a certain amount of gas. Either explicitly or just due to normal operation, the number of iterations in a loop can grow beyond the block gas limit which can cause the complete contract to be stalled at a certain point. This may not apply to view functions that are only executed to read data from the blockchain. Still, such functions may be called by other contracts as part of on-chain operations and stall those. Please be explicit about such cases in the documentation of your contracts.

Recommendation

We recommend that function symbol return String only.

Alleviation

The AMM Coin team was clearly warned.

ECO-03 Gas requirement is infinite

Category	Severity	Location	Status
Gas Consume	Economy	AMM.sol : 208	Partially Resolved

Description

Gas requirement of function AMM. allowance is infinite:
If the gas requirement of a function is higher than the block gas limit, it cannot be executed. Please avoid loops in your functions or actions that modify large areas of storage (this includes clearing or copying arrays in storage).

Loops that do not have a fixed number of iterations, for example, loops that depend on storage values, have to be used carefully: Due to the block gas limit, transactions can only consume a certain amount of gas. Either explicitly or just due to normal operation, the number of iterations in a loop can grow beyond the block gas limit which can cause the complete contract to be stalled at a certain point. This may not apply to view functions that are only executed to read data from the blockchain. Still, such functions may be called by other contracts as part of on-chain operations and stall those. Please be explicit about such cases in the documentation of your contracts.

Recommendation

We recommend that function allowance reduce gas.

Alleviation

The AMM Coin team was clearly warned.

ECO-04 Gas requirement is infinite

Category	Severity	Location	Status
Gas Consume	Economy	AMM.sol : 217	Resolved

Description

Gas requirement of function AMM. transferFrom is infinite:
If the gas requirement of a function is higher than the block gas limit, it cannot be executed. Please avoid loops in your functions or actions that modify large areas of storage (this includes clearing or copying arrays in storage).

Loops that do not have a fixed number of iterations, for example, loops that depend on storage values, have to be used carefully: Due to the block gas limit, transactions can only consume a certain amount of gas. Either explicitly or just due to normal operation, the number of iterations in a loop can grow beyond the block gas limit which can cause the complete contract to be stalled at a certain point. This may not apply to view functions that are only executed to read data from the blockchain. Still, such functions may be called by other contracts as part of on-chain operations and stall those. Please be explicit about such cases in the documentation of your contracts.

Recommendation

We recommend that function transferFrom reduce gas.

Alleviation

The AMM Coin team was clearly warned.

ERC-01 Gas requirement is infinite

Category	Severity	Location	Status
ERC20 Stardand	ERC	AMM.so : 157	Partially Resolved

Description

ERC20 contract's "decimals" function should have "uint8" as return type

Loops that do not have a fixed number of iterations, for example, loops that depend on storage values, have to be used carefully: Due to the block gas limit, transactions can only consume a certain amount of gas. Either explicitly or just due to normal operation, the number of iterations in a loop can grow beyond the block gas limit which can cause the complete contract to be stalled at a certain point. This may not apply to view functions that are only executed to read data from the blockchain. Still, such functions may be called by other contracts as part of on-chain operations and stall those. Please be explicit about such cases in the documentation of your contracts.

Recommendation

We recommend that function decimals return uint8.

Alleviation

The AMM Coin team was clearly warned.

MISC-01 Constant View/Pure functions

Category	Severity	Location	Status
Pure function	Miscellaneous	AMM.sol : 101	Acknowledge

Description

Address.isContract(address) : Is constant but potentially should not be.

Recommendation

We recommend modify like this

```
function isContract(address account) internal view returns (bool) {
    uint256 size;
    // solhint-disable-next-line no-inline-assembly
    assembly { size := extcodesize(account) }
    return size > 0;
}
```

Alleviation

The AMM Coin team was clearly warned.

MISC-02 Similar variable names

Category	Severity	Location	Status
Variable	Miscellaneous	AMM.sol : 271	Acknowledged

Description

ERC20._mint(address,uint256) : Variables have very similar names "account" and "amount".

The same result exists in :

- Line 273
- Line 274
- Line 275
- Line 276
- Line 277
- Line 281
- Line 283
- Line 285
- Line 286
- Line 288
- Line 291
- Line 292

Recommendation

We recommend modify like this

function _mint(address sender, uint amount) internal virtual {

Alleviation

The AMM Coin team was clearly warned.

MISC-03 No return

Category	Severity	Location	Status
No return	Miscellaneous	AMM.sol : 155	Acknowledged

Description

IERC20.name(): Defines a return type but never explicitly returns a value.

The same result exists in :

- Line 156
- Line 157
- Line 158
- Line 159
- Line 160
- Line 161
- Line 162
- Line 163

Recommendation

We recommend modify like this

function balanceOf(address account) external view returns (uint amount)

Alleviation

The AMM Coin team was clearly warned.

MISC-04 Guard conditions

Category	Severity	Location	Status
Guard	Miscellaneous	AMM.sol : 72	Acknowledge

Description

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.

The same result exists in :

- Line 83
- Line 94
- Line 109
- Line 113
- Line 129
- Line 134
- Line 225
- Line 240
- Line 253
- Line 254
- Line 259
- Line 271
- Line 281
- Line 286
- Line 300
- Line 301

Recommendation

We recommend modify like this

require(b > 0, errorMessage);

Alleviation

The AMM Coin team was clearly warned.

MISC-05 Data truncated

Category	Severity	Location	Status
Truncation	Miscellaneous	AMM.sol : 27	Acknowledge

Description

Division of integer values yields an integer value again. That means e.g. 10 / 100 = 0 instead of 0.1 since the result is an integer again. This does not hold for division of (only) literal values since those yield rational constants.

The same result exists in :

- Line 35
- Line 59
- Line 84

Recommendation

We recommend modify like this

```
require(b > 0 && c > a, errorMessage);
if (c / a != b) return (false, 0);
```

Alleviation

The AMM Coin team was clearly warned.

Appendix

Finding Categories

Centralization / Privilege

Centralization / Privilege findings refer to either feature logic or implementation of components that act against the nature of decentralization, such as explicit ownership or specialized access roles in combination with a mechanism to relocate funds.

Gas Optimization

Gas Optimization findings do not affect the functionality of the code but generate different, more optimal EVM opcodes resulting in a reduction on the total gas cost of a transaction.

Mathematical Operations

Mathematical Operation findings relate to mishandling of math formulas, such as overflows, incorrect operations etc.

Logical Issue

Logical Issue findings detail a fault in the logic of the linked code, such as an incorrect notion on how block.timestamp works.

Control Flow

Control Flow findings concern the access control imposed on functions, such as owner-only functions being invoke-able by anyone under certain circumstances.

Language Specific

Language Specific findings are issues that would only arise within Solidity, i.e. incorrect usage of private or delete.

Coding Style

Coding Style findings usually do not affect the generated byte-code but rather comment on how to make the codebase more legible and, as a result, easily maintainable.

Checksum Calculation Method

The "Checksum" field in the "Audit Scope" section is calculated as the SHA-256 (Secure Hash Algorithm 2 with digest size of 256 bits) digest of the content of each file hosted in the listed source repository under the specified commit.

The result is hexadecimal encoded and is the same as the output of the Linux "sha256sum" command against the target file.

Disclaimer

This report is subject to the terms and conditions (including without limitation, description of services, confidentiality, disclaimer and limitation of liability) set forth in the Services Agreement, or the scope of services, and terms and conditions provided to you ("Customer" or the "Company") in connection with the Agreement. This report provided in connection with the Services set forth in the Agreement shall be used by the Company only to the extent permitted under the terms and conditions set forth in the Agreement.

This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes, nor may copies be delivered to any other person other than the Company, without ScanSandBox's prior written consent in each instance. This report is not, nor should be considered, an "endorsement" or "disapproval" of any particular project or team. This report is not, nor should be considered, an indication of the economics or value of any "product" or "asset" created by any team or project that contracts ScanSandBox to perform a security assessment. This report does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors, business, business model or legal compliance.

This report should not be used in any way to make decisions around investment or involvement with any particular project. This report in no way provides investment advice, nor should be leveraged as investment advice of any sort. This report represents an extensive assessing process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.

Blockchain technology and cryptographic assets present a high level of ongoing risk. ScanSandBox's position is that each company and individual are responsible for their own due diligence and continuous security. ScanSandBox's goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies, and in no way claims any guarantee of security or functionality of the technology we agree to analyze.

The assessment services provided by ScanSandBox is subject to dependencies and under continuing development. You agree that your access and/or use, including but not limited to any services, reports, and materials, will be at your sole risk on an as-is, where-is, and as-available basis. Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty. The assessment reports could include false positives, false negatives, and other unpredictable results. The services may access, and depend upon, multiple layers of third-parties.

ALL SERVICES, THE LABELS, THE ASSESSMENT REPORT, WORK PRODUCT, OR OTHER MATERIALS, OR ANY PRODUCTS OR RESULTS OF THE USE THEREOF ARE PROVIDED “AS IS” AND “AS VAILABLE” AND WITH ALL FAULTS AND DEFECTS WITHOUT WARRANTY OF ANY KIND. TO THE MAXIMUM EXTENT PERMITTED UNDER APPLICABLE LAW, SCANSANDBOX HEREBY DISCLAIMS ALL WARRANTIES, WHETHER EXPRESS, IMPLIED, STATUTORY, OR OTHERWISE WITH RESPECT TO THE SERVICES, ASSESSMENT REPORT, OR OTHER MATERIALS. WITHOUT LIMITING THE FOREGOING, SCANSANDBOX SPECIFICALLY DISCLAIMS ALL IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT, AND ALL WARRANTIES ARISING FROM COURSE OF DEALING, USAGE, OR TRADE PRACTICE. WITHOUT LIMITING THE FOREGOING, SCANSANDBOX MAKES NO WARRANTY OF ANY KIND THAT THE SERVICES, THE LABELS, THE ASSESSMENT EPORT, WORK PRODUCT, OR OTHER MATERIALS, OR ANY PRODUCTS OR RESULTS OF THE USE THEREOF, WILL MEET CUSTOMER’S OR ANY OTHER PERSON’S REQUIREMENTS, ACHIEVE ANY INTENDED RESULT, BE COMPATIBLE OR WORK WITH ANY SOFTWARE, SYSTEM, OR OTHER SERVICES, OR BE SECURE, ACCURATE, COMPLETE, FREE OF HARMFUL CODE, OR ERROR-FREE. WITHOUT LIMITATION TO THE FOREGOING, SCANSANDBOX PROVIDES NO WARRANTY OR UNDERTAKING, AND MAKES NO REPRESENTATION OF ANY KIND THAT THE SERVICE WILL MEET CUSTOMER’S REQUIREMENTS, ACHIEVE ANY INTENDED RESULTS, BE COMPATIBLE OR WORK WITH ANY OTHER SOFTWARE, APPLICATIONS, SYSTEMS OR SERVICES, OPERATE WITHOUT INTERRUPTION, MEET ANY PERFORMANCE OR RELIABILITY STANDARDS OR BE ERROR FREE OR THAT ANY ERRORS OR DEFECTS CAN OR WILL BE CORRECTED.

WITHOUT LIMITING THE FOREGOING, NEITHER SCANSANDBOX NOR ANY OF SCANSANDBOX’S AGENTS MAKES ANY REPRESENTATION OR WARRANTY OF ANY KIND, EXPRESS OR IMPLIED AS TO THE ACCURACY, RELIABILITY, OR CURRENCY OF ANY INFORMATION OR CONTENT PROVIDED THROUGH THE SERVICE. SCANSANDBOX WILL ASSUME NO LIABILITY OR RESPONSIBILITY FOR (I) ANY ERRORS, MISTAKES, OR INACCURACIES OF CONTENT AND MATERIALS OR FOR ANY LOSS OR DAMAGE OF ANY KIND INCURRED AS A RESULT OF THE USE OF ANY CONTENT, OR (II) ANY PERSONAL INJURY OR PROPERTY DAMAGE, OF ANY NATURE WHATSOEVER, RESULTING FROM CUSTOMER’S ACCESS TO OR USE OF THE SERVICES, ASSESSMENT REPORT, OR OTHER MATERIALS.

ALL THIRD-PARTY MATERIALS ARE PROVIDED “AS IS” AND ANY REPRESENTATION OR WARRANTY OF OR CONCERNING ANY THIRD-PARTY MATERIALS IS STRICTLY BETWEEN CUSTOMER AND THE THIRD-PARTY OWNER OR DISTRIBUTOR OF THE THIRD-PARTY MATERIALS.

THE SERVICES, ASSESSMENT REPORT, AND ANY OTHER MATERIALS HEREUNDER ARE SOLELY PROVIDED TO CUSTOMER AND MAY NOT BE RELIED ON BY ANY OTHER PERSON OR FOR ANY PURPOSE NOT SPECIFICALLY IDENTIFIED IN THIS AGREEMENT, NOR MAY COPIES BE DELIVERED TO, ANY OTHER PERSON WITHOUT SCANSANDBOX’S PRIOR WRITTEN CONSENT IN EACH INSTANCE.

NO THIRD PARTY OR ANYONE ACTING ON BEHALF OF ANY THEREOF, SHALL BE A THIRD PARTY OR OTHER BENEFICIARY OF SUCH SERVICES, ASSESSMENT REPORT, AND ANY ACCOMPANYING MATERIALS AND NO SUCH THIRD PARTY SHALL HAVE ANY RIGHTS OF CONTRIBUTION AGAINST SCANSANDBOX WITH RESPECT TO SUCH SERVICES, ASSESSMENT REPORT, AND ANY ACCOMPANYING MATERIALS.

THE REPRESENTATIONS AND WARRANTIES OF SCANSANDBOX CONTAINED IN THIS AGREEMENT ARE SOLELY FOR THE BENEFIT OF CUSTOMER. ACCORDINGLY, NO THIRD PARTY OR ANYONE ACTING ON BEHALF OF ANY THEREOF, SHALL BE A THIRD PARTY OR OTHER BENEFICIARY OF SUCH REPRESENTATIONS AND WARRANTIES AND NO SUCH THIRD PARTY SHALL HAVE ANY RIGHTS OF CONTRIBUTION AGAINST SCANSANDBOX WITH RESPECT TO SUCH REPRESENTATIONS OR WARRANTIES OR ANY MATTER SUBJECT TO OR RESULTING IN INDEMNIFICATION UNDER THIS AGREEMENT OR OTHERWISE.

FOR AVOIDANCE OF DOUBT, THE SERVICES, INCLUDING ANY ASSOCIATED ASSESSMENT REPORTS OR MATERIALS, SHALL NOT BE CONSIDERED OR RELIED UPON AS ANY FORM OF FINANCIAL, TAX, LEGAL, REGULATORY, OR OTHER ADVICE.

About

Founded in 2021 by leading academics in the field of Computer Science from Columbia University, ScanSandBox is a leading blockchain security company that serves to verify the security and correctness of smart contracts and blockchain-based protocols. Through the utilization of our world-class technical expertise, alongside our proprietary, innovative tech, we're able to support the success of our clients with best-in-class security, all whilst realizing our overarching vision; provable trust for all throughout all facets of blockchain.

ScanSanBox™