AegisX

Smart Contract Audits | KYC

PALLADIUM

Security Assessment

Day of Defeat FundPool

January 14, 2023

Table of Contents

Assessment Summary

This report has been prepared for Day of Defeat FundPool on the BNB Chain network. AegisX provides both client-centered and user-centered examination of the smart contracts and their current status when applicable. This report represents the security assessment made to find issues and vulnerabilities on the source code along with the current liquidity and token holder statistics of the protocol.

A comprehensive examination has been performed, utilizing Cross Referencing, Static Analysis, In-House Security Tools, and line-by-line Manual Review.

The auditing process pays special attention to the following considerations:

- Testing the smart contracts against both common and uncommon attack vectors.
- Inspecting liquidity and holders statistics to inform the current status to both users and client when applicable.
- Assessing the codebase to ensure compliance with current best practices and industry standards.
- Verifying contract functions that allow trusted and/or untrusted actors to mint, lock, pause, and transfer assets.
- 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.

Technical Findings Summary

Classification of Risk

Severity	Description
Critical	Risks are those that impact the safe functioning of a platform and must be addressed before launch. Users should not invest in any project with outstanding critical risks.
Major	Risks can include centralization issues and logical errors. Under specific circumstances, these major risks can lead to loss of funds and/or control of the project.
Medium	Risks may not pose a direct risk to users' funds, but they can affect the overall functioning of a platform
Minor	Risks can be any of the above but on a smaller scale. They generally do not compromise the overall integrity of the Project, but they may be less efficient than other solutions.
Informational	Errors are often recommended to improve the code's style or certain operations to fall within industry best practices. They usually do not affect the overall functioning of the code.

Findings

Severity	Found	Pending	Resolved
Critical	1	0	1
Major	0	0	0
Medium	1	0	1
Minor	4	0	4
Informational	3	1	2
Total	9	1	8

Project Overview

Contract Summary

Parameter	Result
Address	
Name	Day of Defeat
Token Tracker	
Decimals	
Supply	
Platform	BNB Chain
compiler	v0.8.0^
Contract Name	FundPool
Optimization	
LicenseType	MIT
Language	Solidity
Codebase	Solidity file provided by the project team.
Payment Tx	

Main Contract Assessed Contract Name

Name	Contract	Live
Day of Defeat		No

TestNet Contract Assessed Contract Name

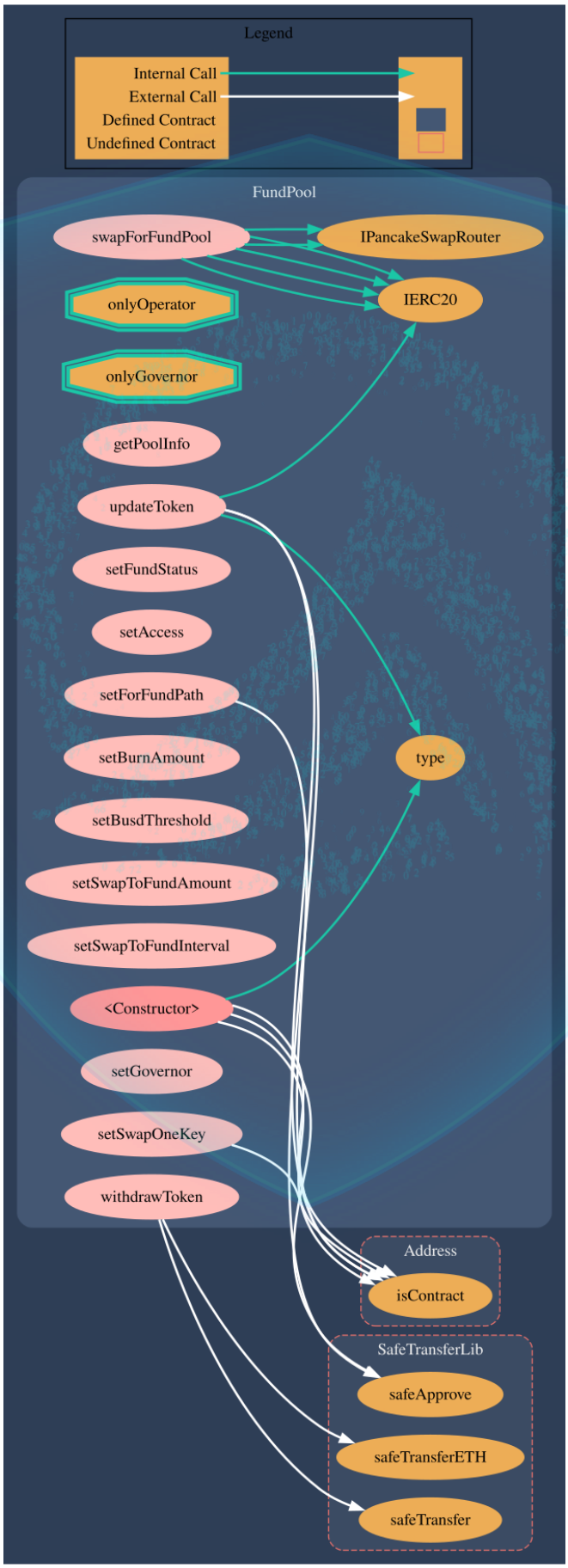
Name	Contract	Live
Day of Defeat	0x4d6bd230B08A2f938d2867FE0036c26F3BEE90ce	No

Solidity Code Provided

SolID	File Sha-1	FileName
FundPool	1f0025bfb111556c5b7aa20269a94fcd886ab631	FundPool.sol

Call Graph

The contract for Day of Defeat has the following call graph structure.



KYC Information

The Project Owners of Day of Defeat are not KYC'd. .

The owner wallet has the power to call the functions displayed on the priviliged functions chart below, if the owner wallet is compromised this privileges could be exploited.

We recommend the team to renounce ownership at the right timing if possible, or gradually migrate to a timelock with governing functionalities in respect of transparency and safety considerations.

KYC Information Notes:

Auditor Notes:

Project Owner Notes:

Smart Contract Vulnerability Checks

ID	Severity	Name	File	location
SWC-100	Pass	Function Default Visibility	FundPool.sol	L: 0 C: 0
SWC-101	Pass	Integer Overflow and Underflow.	FundPool.sol	L: 0 C: 0
SWC-102	Pass	Outdated Compiler Version file.	FundPool.sol	L: 0 C: 0
SWC-103	Low	A floating pragma is set.	FundPool.sol	IPancakeSwapRouter.sol, L: 2 C: 0
SWC-104	Pass	Unchecked Call Return Value.	FundPool.sol	L: 0 C: 0
SWC-105	Pass	Unprotected Ether Withdrawal.	FundPool.sol	L: 0 C: 0
SWC-106	Pass	Unprotected SELFDESTRUCT Instruction	FundPool.sol	L: 0 C: 0
SWC-107	Pass	Read of persistent state following external call.	FundPool.sol	L: 0 C: 0
SWC-108	Pass	State variable visibility is not set..	FundPool.sol	L: 0 C: 0
SWC-109	Pass	Uninitialized Storage Pointer.	FundPool.sol	L: 0 C: 0
SWC-110	Pass	Assert Violation.	FundPool.sol	L: 0 C: 0
SWC-111	Pass	Use of Deprecated Solidity Functions.	FundPool.sol	L: 0 C: 0

ID	Severity	Name	File	location
SWC-112	Pass	Delegate Call to Untrusted Callee.	FundPool.sol	L: 0 C: 0
SWC-113	Pass	Multiple calls are executed in the same transaction.	FundPool.sol	L: 0 C: 0
SWC-114	Pass	Transaction Order Dependence.	FundPool.sol	L: 0 C: 0
SWC-115	Pass	Authorization through tx.origin.	FundPool.sol	L: 0 C: 0
SWC-116	Pass	A control flow decision is made based on The block.timestamp environment variable.	FundPool.sol	L: 0 C: 0
SWC-117	Pass	Signature Malleability.	FundPool.sol	L: 0 C: 0
SWC-118	Pass	Incorrect Constructor Name.	FundPool.sol	L: 0 C: 0
SWC-119	Pass	Shadowing State Variables.	FundPool.sol	L: 0 C: 0
SWC-120	Pass	Potential use of block.number as source of randonmness.	FundPool.sol	L: 0 C: 0
SWC-121	Pass	Missing Protection against Signature Replay Attacks.	FundPool.sol	L: 0 C: 0
SWC-122	Pass	Lack of Proper Signature Verification.	FundPool.sol	L: 0 C: 0
SWC-123	Pass	Requirement Violation.	FundPool.sol	L: 0 C: 0
SWC-124	Pass	Write to Arbitrary Storage Location.	FundPool.sol	L: 0 C: 0
SWC-125	Pass	Incorrect Inheritance Order.	FundPool.sol	L: 0 C: 0
SWC-126	Pass	Insufficient Gas Griefing.	FundPool.sol	L: 0 C: 0

ID	Severity	Name	File	location
SWC-127	Pass	Arbitrary Jump with Function Type Variable.	FundPool.sol	L: 0 C: 0
SWC-128	Pass	DoS With Block Gas Limit.	FundPool.sol	L: 0 C: 0
SWC-129	Pass	Typographical Error.	FundPool.sol	L: 0 C: 0
SWC-130	Pass	Right-To-Left-Override control character (U+202E).	FundPool.sol	L: 0 C: 0
SWC-131	Pass	Presence of unused variables.	FundPool.sol	L: 0 C: 0
SWC-132	Pass	Unexpected Ether balance.	FundPool.sol	L: 0 C: 0
SWC-133	Pass	Hash Collisions with Multiple Variable Length Arguments.	FundPool.sol	L: 0 C: 0
SWC-134	Pass	Message call with hardcoded gas amount.	FundPool.sol	L: 0 C: 0
SWC-135	Pass	Code With No Effects (Irrelevant/Dead Code).	FundPool.sol	L: 0 C: 0
SWC-136	Pass	Unencrypted Private Data On-Chain.	FundPool.sol	L: 0 C: 0

We scan the contract for additional security issues using MYTHX and industry-standard security scanning tools.

Smart Contract Vulnerability Details

SWC-103 - Floating Pragma.

CWE-664: Improper Control of a Resource Through its Lifetime.

References:

Description:

Contracts should be deployed with the same compiler version and flags that they have been tested with thoroughly. Locking the pragma helps to ensure that contracts do not accidentally get deployed using, for example, an outdated compiler version that might introduce bugs that affect the contract system negatively.

Remediation:

Lock the pragma version and also consider known bugs (https://github.com/ethereum/solidity/releases) for the compiler version that is chosen.

Pragma statements can be allowed to float when a contract is intended for consumption by other developers, as in the case with contracts in a library or EthPM package. Otherwise, the developer would need to manually update the pragma in order to compile locally.

References:

Ethereum Smart Contract Best Practices - Lock pragmas to specific compiler version.

Inheritance

The contract for Day of Defeat has the following inheritance structure.

Privileged Functions (onlyOwner)

Function Name	Parameters	Visibility
setFundStatus	bool _enable	External
setAccess	address account, bool _access	External
setForFundPath	address[] calldata _path	External
setBurnAmount	uint256 _burnAmount	External
setBusdThreshold	uint256 _busdThreshold	External
setSwapToFundAmount	uint256 _swapAmount	External
setSwapToFundInterval	uint256 _swapToFundInterval	External

Function Name	Parameters	Visibility
setSwapOneKey	address[] calldata _path, bool _fundStatus, uint256 _burnAmount, uint256 _swapAmount, uint256 _swapToFundInterval	External

DOD-01 | Potential Sandwich Attacks.

Category	Severity	Location	Status
Security	Informational	FundPool.sol: 146,25	Pending

Description

A sandwich attack might happen when an attacker observes a transaction swapping tokens or adding liquidity without setting restrictions on slippage or minimum output amount. The attacker can manipulate the exchange rate by frontrunning (before the transaction being attacked) a transaction to purchase one of the assets and make profits by back running (after the transaction being attacked) a transaction to sell the asset. The following functions are called without setting restrictions on slippage or minimum output amount, so transactions triggering these functions are vulnerable to sandwich attacks, especially when the input amount is large:

Function Name	Slippage
swapExactTokensForTokensSupportingFeeOnTransferTokens	50%

Remediation

We recommend setting reasonable minimum output amounts, instead of 0, based on token prices when calling the aforementioned functions.

Project Action

function swapForFundPool; An Oracle Implementation recommended from the previous review was followed up by the dev and an oracle has been implemented, commendably. However, with a 50% slippage, this still may result a sandwich attack.

Referrences:

What Are Sandwich Attacks in DeFi — and How Can You Avoid Them?.

DOD-03 | Lack of Input Validation.

Category	Severity	Location	Status
Volatile Code	Minor	FundPool.sol: 138,5, 143,5, 158,5, 163,5	Resolved

Description

The given input is missing the check for the non-zero address and/or check for the value that is already set.

Remediation

We advise the client to add the check for the passed-in values to prevent unexpected errors as below:

```
...
 require(receiver != address(0), "Receiver is the zero address");
 require(currentValue != NewValue, "Already set to the same value");
...
```

Project Action

Since the initial review, input validations have been implemented on many functions by the dev. However, there still are some functions that can utilize input validations. Ie. Validating the value being set isn't already set to the same. It's the best practice to utilize require to ensure the data is valid and not waste gas.

All functions have input validations.

DOD-05 | Missing Event Emission.

Category	Severity	Location	Status
Volatile Code	Minor	FundPool.sol:	Resolved

Description

Detected missing events for critical arithmetic parameters. There are functions that have no event emitted, so it is difficult to track off-chain changes.The linked code does not create an event for the transfer.

Remediation

Emit an event for critical parameter changes. It is recommended emitting events for the sensitive functions that are controlled by centralization roles.

Project Action

Previous: All of the functions; the developer should consider adding an emit or log file to the functions so they are recorded into the blockchain.

FOLLOW-UP: Event emissions have been implemented in most of the functions.

DOD-07 | State Variables could be Declared Constant.

Category	Severity	Location	Status
Coding Style	Minor	FundPool.sol: 14,5, 15,5, 16,5	Resolved

Description

Constant state variables should be declared constant to save gas.

```
BUSD
BNB
ROUTER
```

Remediation

Add the constant attribute to state variables that never changes.

https://docs.soliditylang.org/en/latest/contracts.html#constant-state-variables

Project Action

Previous: Declaring these addresses as a constant variable recommended to save gas.

FOLLOW-UP: Have been declared constant.

DOD-11 | Sell Tx Fail.

Category	Severity	Location	Status
Transfer Fail	Medium	FundPool.sol: 124,5, DODTokenV2 - 144,5	Resolved

Description

The testing revealed that there are instances of sell transactions failing when the fundPool already has met the threshold to complete a swap of tokens within the function of swapForFundPool, but the very same sell transaction that would trigger the swap causes a discrepancy with the number of tokens being transferred in as a tax and the drop of DOD token price, leading to threshold no longer being met. To remedy this, a manual trigger of swapForFundPool or selling of a larger number of tokens that makes the threshold to be met despite the price drop was necessary.

Remediation

Limiting the threshold for DOD tokens to BUSD swap to simply the number of tokens without the price is recommended. May let the token contract to receive whatever amount of BUSD swapped as the current V1 token contract is doing, or may change the recipient to FundPool and a logic to transfer 100 BUSD to the token contract when it meets the threshold can coexist. This may lead to less than extra 100 BUSD tokens remaining in the FundPool contract after the Reward pool unlocks, but these can be withdrawn with the function withdrawToken with the DAO's approval.

Project Action

7th: Simple threshold of a fixed amount of tokens trigger has been implemented. Successful sell transactions with the contract triggered swap in the function verified.

DOD-12 | Centralization Risks In The onlyOperator Role(s)

Category	Severity	Location	Status
Centralization / Privilege	Minor	FundPool.sol: 56,6	Resolved

Description

In the contract FundPool, the role onlyOperator has authority over the functions that lead to centralization risks.

Any compromise to the onlyOperator account(s) may allow the hacker to take advantage of this authority.

Remediation

The risk describes the current project design and potentially makes iterations to improve in the security operation and level of decentralization, which in most cases cannot be resolved entirely at the present stage.

We advise the client to carefully manage the privileged account's private key to avoid any potential risks of being hacked.

In general, we strongly recommend centralized privileges or roles in the protocol be improved via a decentralized mechanism or smart-contract-based accounts with enhanced security practices, e.g., multisignature wallets.

Project Action

Centralization risk resides with onlyOperator at fundStatus, access, swapPath, burnAmount, swapToFundAmount, and swapToFundInterval. Multi-sig safe contract for onlyOperator role along with the declaration on the contract recommended.

It's now clearly stated on the contract in comments that the address assigned to the onlyOperator role will be passed onto a multi-sig safe contract once the contract is deployed.

DOD-13 | Extra Gas Cost For User

Category	Severity	Location	Status
Logical Issue	Informational	FundPool.sol: 160,17, 286,9, 128,5	Resolved

Description

The user may trigger a tax distribution during the transfer process, which will cost a lot of gas and it is unfair to let a single user bear it.

Remediation

We advise the client to make the owner responsible for the gas costs of the tax distribution.

Project Action

7th: Recommendations - (1) Moving lines 155 & 156 to inside line 161 if statement recommended. (2) function to change the dodToken CA by onlyGovernor, and removal of Line 158 recommended. It's already done in the constructor. Line 158 would only have its usage and effectiveness worthy of its gas cost if there is a function to change the dodToken CA and it lies in that function.

8th: (1) Implemented. getAmountsOut is now only called if dodBalance >= swapToFundAmount. (2) Implemented. safeApprove(dodToken, ROUTER, type(uint256).max) now resides in the function updateToken. (3) Furthermore, the boolean executed and return values have been removed, further improving the gas cost.

DOD-14 | Unnecessary Use Of SafeMath

Category	Severity	Location	Status
Logical Issue	Informational	FundPool.sol:	Resolved

Description

The SafeMath library is used unnecessarily. With Solidity compiler versions 0.8.0 or newer, arithmetic operations will automatically revert in case of integer overflow or underflow.

An implementation of SafeMath library is found. SafeMath library is used for uint256 type in FundPool contract.

Remediation

We advise removing the usage of SafeMath library and using the built-in arithmetic operations provided by the Solidity programming language

Project Action

Compiler version was updated and Safemath was eliminated.

DOD-15 | Divide Before Multiply.

Category	Severity	Location	Status
Mathematical Operations	Critical	FundPool.sol: 707,13, 826,9	Resolved

Description

Starting from line 707 to 826, it was found that divisions are being done before multiplication. Performing integer division before multiplication truncates the low bits, losing the precision of calculation.

Remediation

It is strongly advised to apply multiplication before division to avoid loss of precision that can result in a significant loss in assets

Project Action

All of the arithmetic equations have been updated to perform multiplication before division.

Social Media Checks

Social Media	URL	Result
Website	https://www.dayofdefeat.app/	Pass
Telegram	https://t.me/DayOfDefeatBSC	Pass
Twitter	https://twitter.com/dayofdefeatBSC	Pass
OtherSocial	https://titanservice.cn/dayofdefeatCN	Pass

We recommend to have 3 or more social media sources including a completed working websites.

Social Media Information Notes:

Auditor Notes: undefined

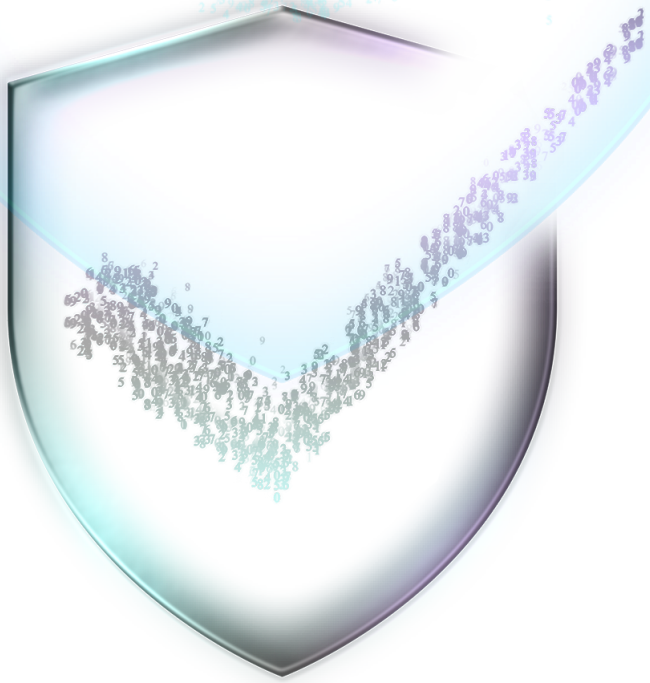
Project Owner Notes:

Assessment Results

Score Results

Review	Score
Overall Score	96/100
Auditor Score	94/100
Review by Section	**Score**
Manual Scan Score	16/18
SWC Scan Score	36/37
Advance Check Score	44/45

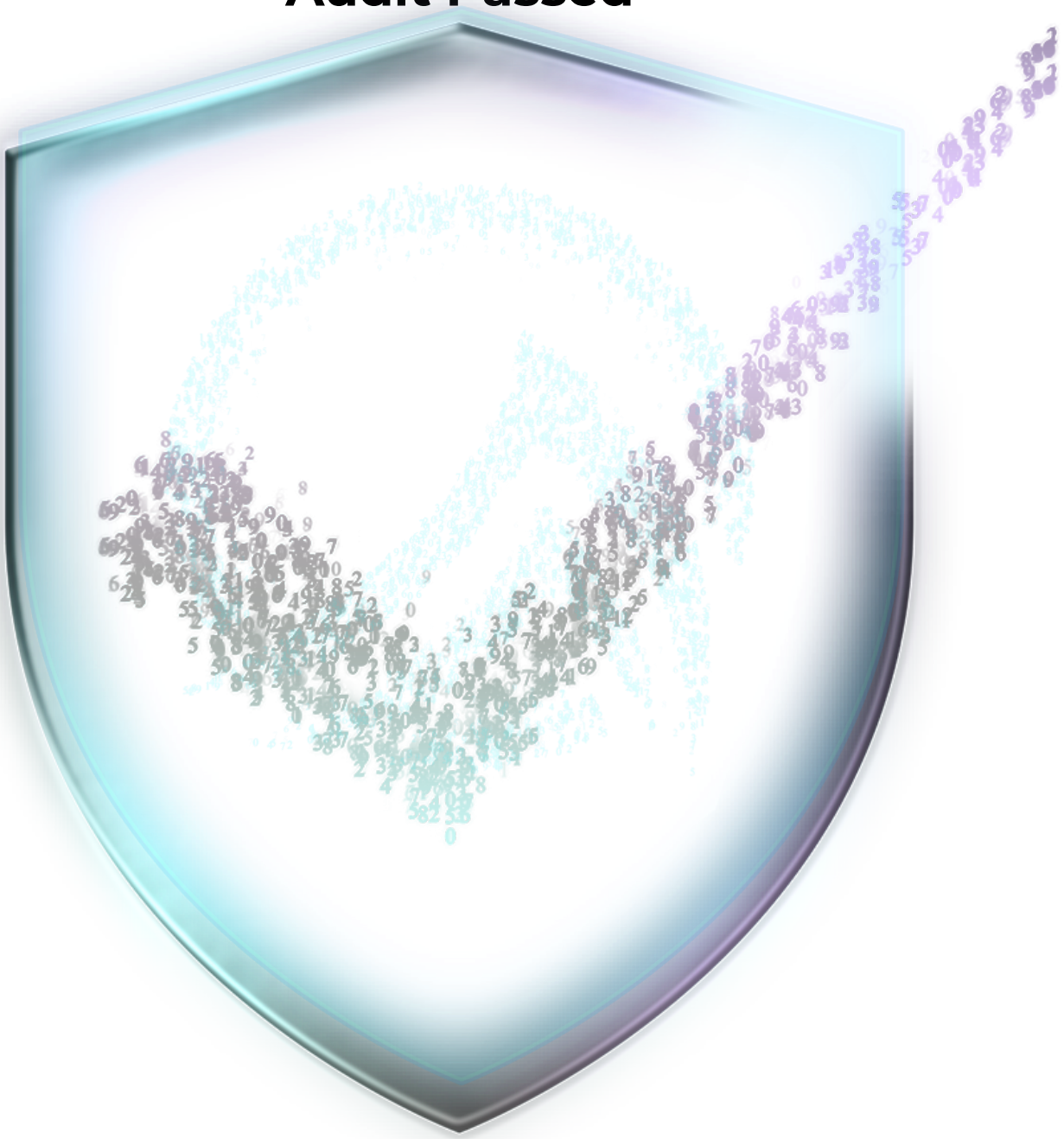
The maximum score is 100, however to attain that value the project must pass the reviews and provide all the data needed for the assessment. Minimum score to pass is 80 points. If a project fails to attain 80 and/or has unresolved critical and/or major and/or medium finding(s) in the Palladium tier assessments, an automatic failure is given. Read our notes and final assessment below.



PASSED

Assessment Results

Auditor Score = 94
Audit Passed



Important Notes from the Auditor:

- 9th(NEW): Concern (1) has been addressed by updating the FundPool contract to burn the DOD tokens to the same DEAD address and updating the function _burn to transfer to the same DEAD address.
- 8th: Concern #1) Although in the DOD Token contract, the burn address DEAD has been declared and the function _transfer updates _totalSupply by subtracting the DOD tokens being burned to DEAD address, the FundPool does not burn to the same DEAD address, hence the function swap and the logic inside the DOD Token contract to calculate the exchange ratio when all 3 conditions aren't met will not properly function.
-
- 9th(NEW): Gas optimization - The number of characters in the function swapForFundPool has been reduced.
- 8th: Gas optimization recommendations - (1) Implemented. getAmountsOut is now only called if dodBalance >= swapToFundAmount. (2) Implemented. safeApprove(dodToken, ROUTER, type(uint256).max) now resides in the function updateToken. (3)

Furthermore, the boolean executed and return bool have been removed, further improving the gas cost.

- 7th: Gas optimization recommendations - (1) Moving lines 155 & 156 to inside line 161 if statement recommended. (2) Removal of Line 158 recommended. It's already done in the constructor. Line 158 would only have its usage and effectiveness worthy of its gas cost if there is a function to change the dodToken CA and it lies in that function.
-
- 8th: It's now clearly stated on the contract in comments that the address assigned to the onlyOperator role will be passed onto a multi-sig safe contract once the contract is deployed.
- 7th: The declaration of multi-sig safe use on the contract is still not present.
- 6th: Centralization risk resides with onlyOperator at fundStatus, access, swapPath, burnAmount, swapToFundAmount, and swapToFundInterval. Use of multi-sig safe contract for onlyOperator role along with the declaration on the contract itself recommended.
- 5th: An address input to assign the operator role for FundPool & MarketingPool have been implemented (its

been clearly stated that a multi-sig safe will be utilized for the administrative role addresses). However, the issue with initiating automatic swaps could not be verified due to the critical errors with the involved variables.

- 4th: A function to stop the FundPool & MarketingPool swaps have been implemented as well as a function to replace the FundPool & MarketingPool by the DODGovernor. However, the issue with initiating automatic swaps could not be verified due to the critical errors with the involved variables.
- 3rd: FundPool & MarketingPool smart contracts serve their purpose of independently handling taxed funds for rewards/burn and marketing.
- FIRSTLY, onlyOperator role in these two contracts which gets assigned to the contract deployer can potentially prevent functionalities of the whole project by limiting the token's tax mechanism with functions such as setAccess, set...Path, etc. The need for an administrative role for these settings is understandable, however, the centralization risk and potential harm can follow in case the Operator wallet gets compromised. Please carefully review if these functions with onlyOperator modifier are absolutely necessary, and if so, please use extra caution on who's given this privilege and do consider using a multi-

sig contract for this Operator role. Lastly, do consider functions that can replace the FundPool contract and MarketingPool contract in case any of these contracts/ Operators get compromised.

- Also SECONDLY, the tests revealed that both Fundpool & MarketingPool failed to initiate swaps in the functions swapForFundPool() & swapForMarketingPool() even when the conditions were met. Please review the boolean variables fundStatus & marketingStatus in respective contracts, 'to' address parameter on IPancakeSwapRouter swap functions, and who becomes the initiator of the swaps to pay the necessary gas fees.

- 7th: Simple threshold of a fixed amount of tokens trigger has been implemented. Successful sell transactions with the contract triggered swap in the function verified.

- 6th: Successful automatic swaps were verified. However there is a standing issue of instances of sell transactions failing when the fundPool already has met the threshold to complete a swap of tokens within the function of swapForFundPool, but the very same sell transaction that would trigger the swap causes a discrepancy with the number of tokens being transferred in as a tax and the

drop of DOD token price, leading to threshold no longer being met.

- 2nd: Updated to the latest compiler version.
- 1st: Use of the most up-to-date compiler version is recommended to avoid known bugs and chances of exploits.

- 2nd: All necessary files have been provided.
- 1st: A complete audit cannot be done as key information behind the custom interface, IDao is missing.

- 2nd: All arithmetic equations have been updated to do multiplication before division.
- 1st: Division before multiplication will result in a loss of precision in arithmetic calculations, which can lead to a significant loss in assets.

Appendix

Finding Categories

Centralization / Privilege

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

Gas Optimization

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

Logical Issue

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

Control Flow

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

Volatile Code

Volatile Code findings refer to segments of code that behave unexpectedly on certain edge cases that mayresult in a vulnerability.

Coding Style

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

Inconsistency

Inconsistency findings refer to functions that should seemingly behave similarly yet contain different code,such as a constructor assignment imposing different require statements on the input variables than a setterfunction.

Coding Best Practices

ERC 20 Conding Standards are a set of rules that each developer should follow to ensure the code meet a set of creterias and is readable by all the developers.

Disclaimer

AegisX has conducted an independent security assessment to verify the integrity of and highlight any vulnerabilities or errors, intentional or unintentional, that may be present in the reviewed code for the scope of this assessment. This report does not constitute agreement, acceptance, or advocation for the Project, and users relying on this report should not consider this as having any merit for financial advice in any shape, form, or nature. The contracts audited do not account for any economic developments that the Project in question may pursue, and the veracity of the findings thus presented in this report relate solely to the proficiency, competence, aptitude, and discretion of our independent auditors, who make no guarantees nor assurance that the contracts are entirely free of exploits, bugs, vulnerabilities or deprecation of technologies.

All information provided in this report does not constitute financial or investment advice, nor should it be used to signal that any persons reading this report should invest their funds without sufficient individual due diligence, regardless of the findings presented. Information is provided 'as is, and AegisX is under no covenant to audited completeness, accuracy, or solidity of the contracts. In no event will AegisX or its partners, employees, agents, or parties related to the provision of this audit report be liable to any parties for, or lack thereof, decisions or actions with regards to the information provided in this audit report.

The assessment services provided by AegisX are subject to dependencies and are under continuing development. You agree that your access 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 with high levels of technical risk and uncertainty. The assessment reports could include false positives, negatives, and unpredictable results. The services may access, and depend upon, multiple layers of third parties.