



Smart Contract Audits | KYC



PALLADIUM

Security Assessment

Day of Defeat Token

December 19, 2022

Table of Contents

1 Assessment Summary

2 Technical Findings Summary

3 Project Overview

3.1 Token Summary

3.2 Risk Analysis Summary

3.3 Main Contract Assessed

4 Smart Contract Risk Checks

4.1 Mint Check

4.2 Fees Check

4.3 Blacklist Check

4.4 MaxTx Check

4.5 Pause Trade Check

5 Contract Ownership

6 Liquidity Ownership

7 KYC Check

8 Smart Contract Vulnerability Checks

8.1 Smart Contract Vulnerability Details

8.2 Smart Contract Inheritance Details

8.3 Smart Contract Privileged Functions

9 Assessment Results and Notes(Important)

10 Social Media Checks(Informational)

11 Technical Findings Details

12 Disclaimer

Assessment Summary

This report has been prepared for Day of Defeat Token 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	2	0	2
● Major	5	2	3
● Medium	0	0	0
● Minor	5	1	4
● Informational	4	3	1
Total	14	7	7

Project Overview

Contract Summary

Parameter	Result
Address	
Name	Day of Defeat
Token Tracker	Day of Defeat (DOD)
Decimals	18
Supply	100,000,000,000,000
Platform	BNB Chain
compiler	v0.8.7^
Contract Name	DODTokenV2
Optimization	
LicenseType	MIT
Language	Solidity
Codebase	Solidity file provided by the project team.
Payment Tx	

Project Overview

Risk Analysis Summary

Parameter	Result
Buy Tax	19%
Sale Tax	19%
Is honeypot?	Clean
Can edit tax?	Yes
Is anti whale?	No
Is blacklisted?	No
Is whitelisted?	Yes
Holders	
Security Score	94
Auditor Score	74
Confidence Level	Medium

The following quick summary it's added to the project overview; however, there are more details about the audit and its results. Please read every detail.

Main Contract Assessed Contract Name

Name	Contract	Live
Day of Defeat		No

TestNet Contract Assessed Contract Name

Name	Contract	Live
Day of Defeat	0xD313193640Ff46e2a85c2416B2ebf382eeF7CB84	No

Solidity Code Provided

SolID	File Sha-1	FileName
DODTokenV2	21d1c6841cfe5af61824a5bf7be81eb51dd69357	DODTokenV2.sol

Mint Check

The project owners of Day of Defeat do not have a mint function in the contract, owner cannot mint tokens after initial deploy.

The Project has a Total Supply of 100,000,000,000,000 and cannot mint any more than the Max Supply.

Mint Notes:

Auditor Notes: A mint Function does exist but it could be only called once when the contract was being created.

Project Owner Notes:



Fees Check

The project owners of Day of Defeat have the ability to set fees over 25%

We Recommend the team to create a new contract with fees restrictions to avoid any problems, as alternative the team can use multi signature wallet to ensure the project is safe from a potential fee increase.

Tax Fee Notes:

Auditor Notes: The contract does have a tax of 19%, but can set up to a maximum of 30% and is controlled by a DAO.

Project Owner Notes:



Blacklist Check

The project owners of Day of Defeat do not have a blacklist function their contract.

The Project allow owners to transfer their tokens without any restrictions.

Token owner cannot blacklist the contract: Malicious or compromised owners can trap contracts relying on tokens with a blacklist.

Blacklist Notes:

Auditor Notes: The contract does not have a blacklist function.

Project Owner Notes:



MaxTx Check

The Project Owners of Day of Defeat can set max tx amount.

The ability to set MaxTx can be used as a bad actor, this can limit the ability of investors to sell their tokens at any given time if is set too low.

We recommend the project to set MaxTx to Total Supply or similar to avoid swap or transfer from failures.

MaxTX Notes:

Auditor Notes: The contract always transfers 99.99% of the full value currently, whether it is a buy/sell/transfer.

Project Owner Notes:



Pause Trade Check

The Project Owners of Day of Defeat don't have the ability to stop or pause trading.

The Team has done a great job to avoid stop trading, and investors has the ability to trade at any given time without any problems

Pause Trade Notes:

Auditor Notes: Does not have a pause trade function.

Project Owner Notes:



Contract Ownership

The contract Day of Defeat is not live yet.



Liquidity Ownership

The token does not have liquidity at the moment of the audit, block

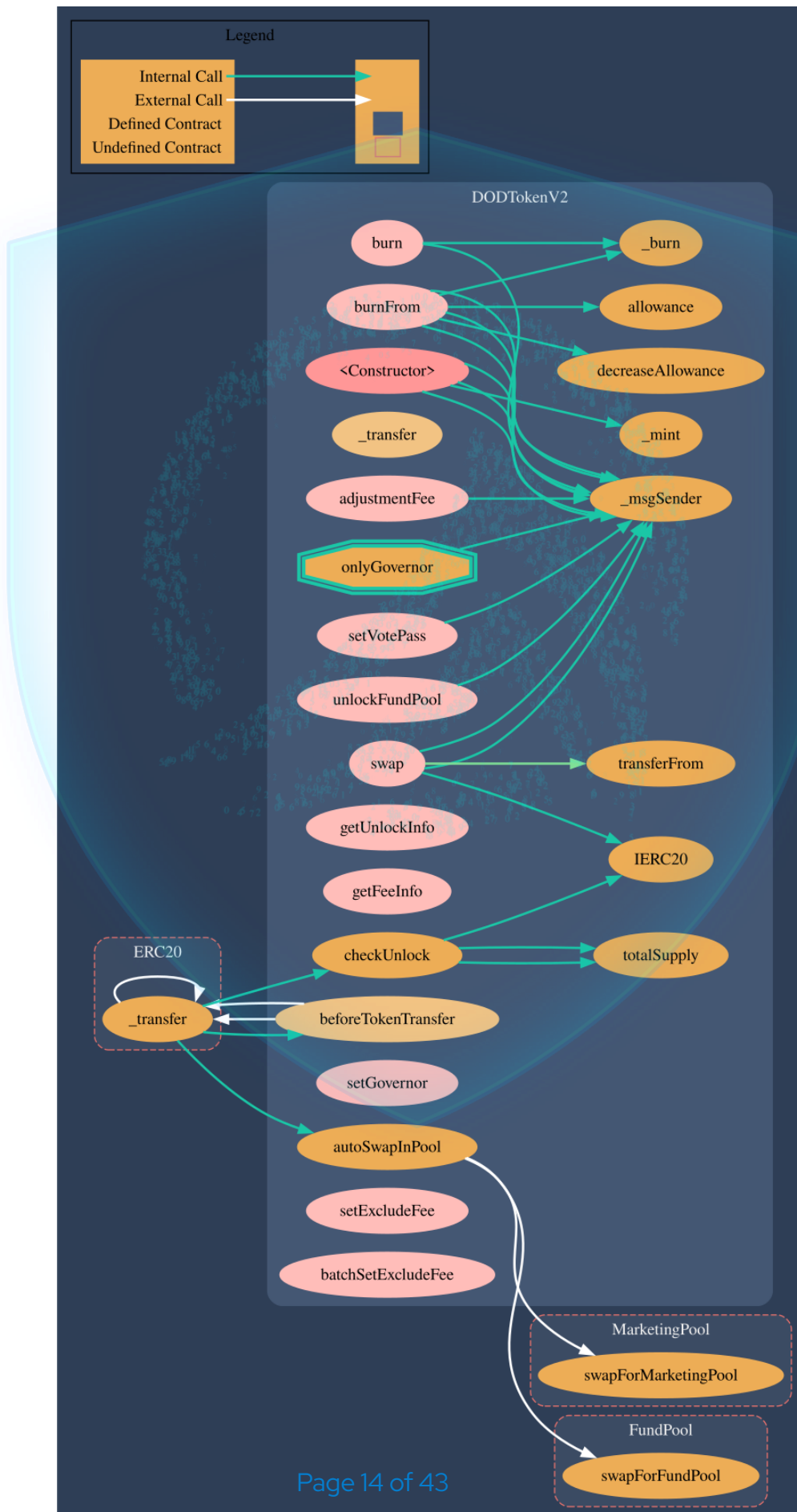
If liquidity is unlocked, then the token developers can do what is infamously known as 'rugpull'. Once investors start buying token from the exchange, the liquidity pool will accumulate more and more coins of established value (e.g., ETH or BNB or Tether). This is because investors are basically sending these tokens of value to the exchange, to get the new token. Developers can withdraw this liquidity from the exchange, cash in all the value and run off with it. Liquidity is locked by renouncing the ownership of liquidity pool (LP) tokens for a fixed time period, by sending them to a time-lock smart contract. Without ownership of LP tokens, developers cannot get liquidity pool funds back. This provides confidence to the investors that the token developers will not run away with the liquidity money. It is now a standard practice that all token developers follow, and this is what really differentiates a scam coin from a real one.

[Read More](#)



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 privileged 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	DODTokenV2.sol	L: 0 C: 0
SWC-101	Pass	Integer Overflow and Underflow.	DODTokenV2.sol	L: 0 C: 0
SWC-102	Pass	Outdated Compiler Version file.	DODTokenV2.sol	L: 0 C: 0
SWC-103	Low	A floating pragma is set.	DODTokenV2.sol	L: 2 C: 2
SWC-104	Pass	Unchecked Call Return Value.	DODTokenV2.sol	L: 0 C: 0
SWC-105	Pass	Unprotected Ether Withdrawal.	DODTokenV2.sol	L: 0 C: 0
SWC-106	Pass	Unprotected SELFDESTRUCT Instruction	DODTokenV2.sol	L: 0 C: 0
SWC-107	Pass	Read of persistent state following external call.	DODTokenV2.sol	L: 0 C: 0
SWC-108	Pass	State variable visibility is not set..	DODTokenV2.sol	L: 0 C: 0
SWC-109	Pass	Uninitialized Storage Pointer.	DODTokenV2.sol	L: 0 C: 0
SWC-110	Pass	Assert Violation.	DODTokenV2.sol	L: 0 C: 0
SWC-111	Pass	Use of Deprecated Solidity Functions.	DODTokenV2.sol	L: 0 C: 0
SWC-112	Pass	Delegate Call to Untrusted Callee.	DODTokenV2.sol	L: 0 C: 0

ID	Severity	Name	File	location
SWC-113	Pass	Multiple calls are executed in the same transaction.	DODTokenV2.sol	L: 0 C: 0
SWC-114	Pass	Transaction Order Dependence.	DODTokenV2.sol	L: 0 C: 0
SWC-115	Pass	Authorization through tx.origin.	DODTokenV2.sol	L: 0 C: 0
SWC-116	Pass	A control flow decision is made based on The block.timestamp environment variable.	DODTokenV2.sol	L: 0 C: 0
SWC-117	Pass	Signature Malleability.	DODTokenV2.sol	L: 0 C: 0
SWC-118	Pass	Incorrect Constructor Name.	DODTokenV2.sol	L: 0 C: 0
SWC-119	Pass	Shadowing State Variables.	DODTokenV2.sol	L: 0 C: 0
SWC-120	Pass	Potential use of block.number as source of randomness.	DODTokenV2.sol	L: 0 C: 0
SWC-121	Pass	Missing Protection against Signature Replay Attacks.	DODTokenV2.sol	L: 0 C: 0
SWC-122	Pass	Lack of Proper Signature Verification.	DODTokenV2.sol	L: 0 C: 0
SWC-123	Pass	Requirement Violation.	DODTokenV2.sol	L: 0 C: 0
SWC-124	Pass	Write to Arbitrary Storage Location.	DODTokenV2.sol	L: 0 C: 0
SWC-125	Pass	Incorrect Inheritance Order.	DODTokenV2.sol	L: 0 C: 0
SWC-126	Pass	Insufficient Gas Griefing.	DODTokenV2.sol	L: 0 C: 0

ID	Severity	Name	File	location
SWC-127	Pass	Arbitrary Jump with Function Type Variable.	DODTokenV2.sol	L: 0 C: 0
SWC-128	Pass	DoS With Block Gas Limit.	DODTokenV2.sol	L: 0 C: 0
SWC-129	Pass	Typographical Error.	DODTokenV2.sol	L: 0 C: 0
SWC-130	Pass	Right-To-Left-Override control character (U+202E).	DODTokenV2.sol	L: 0 C: 0
SWC-131	Pass	Presence of unused variables.	DODTokenV2.sol	L: 0 C: 0
SWC-132	Pass	Unexpected Ether balance.	DODTokenV2.sol	L: 0 C: 0
SWC-133	Pass	Hash Collisions with Multiple Variable Length Arguments.	DODTokenV2.sol	L: 0 C: 0
SWC-134	Pass	Message call with hardcoded gas amount.	DODTokenV2.sol	L: 0 C: 0
SWC-135	Pass	Code With No Effects (Irrelevant/Dead Code).	DODTokenV2.sol	L: 0 C: 0
SWC-136	Pass	Unencrypted Private Data On-Chain.	DODTokenV2.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.

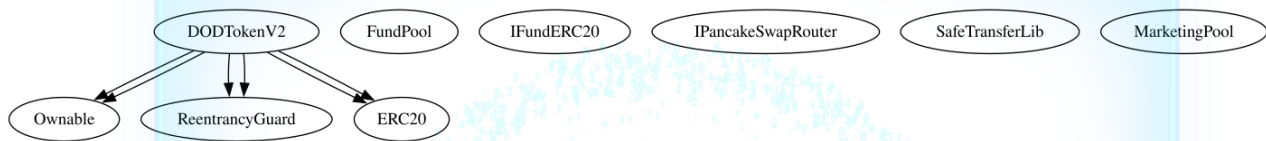
References:

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

Inheritance

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

The Project has a Total Supply of
100,000,000,000,000





Privileged Functions (onlyOwner)

Function Name	Parameters	Visibility
unlockFundPool		External
setExcludeFee		External
batchSetExcludeFee		External



DOD-01 | Potential Sandwich Attacks.

Category	Severity	Location	Status
Security	 Informational	DODTokenV2.sol: 103,21, 110,25, 119,25	 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:

- swapExactETHForTokensSupportingFeeOnTransferTokens()
- swapExactTokensForETHSupportingFeeOnTransferTokens()
- swapExactTokensForTokensSupportingFeeOnTransferTokens()
- addLiquidityETH()

Remediation

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


Project Action

The contract FundPool's function swapForFundPool and the contract MarketingPool's swapForMarketingPool; 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.

References:

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

DOD-02 | Function Visibility Optimization.

Category	Severity	Location	Status
Gas Optimization	● Minor	DODTokenV2.sol:	 Resolved

Description

The following functions are declared as public and are not invoked in any of the contracts contained within the projects scope:

Function Name	Parameters	Visibility
autoSwapInPool		public

The functions that are never called internally within the contract should have external visibility

Remediation

We advise that the function's visibility specifiers are set to external, and the array-based arguments change their data location from memory to calldata, optimizing the gas cost of the function.

Project Action



Review public functions and change those that can be changed to external. ie. burn function called by FundPool and MarketingPool may be set to _burn instead, and set burn as an external function.

All of them have been updated to external except for autoSwapInPool().

References:

external vs public best practices.

DOD-03 | Lack of Input Validation.

Category	Severity	Location	Status
Volatile Code	 Minor	DODTokenV2.sol: 327,5, FundPool: 128,5, 139,5, 144,5, 149,5, MarketingPool: 139,5, 151,5, 156,5	 Pending

Description

The given input is missing the check for the non-zero address.

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");  
...
```

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. It's the best practice to utilize require to ensure the data is valid and not waste gas.

DOD-04 | Centralized Risk In addLiquidity.

Category	Severity	Location	Status
Coding Style	 Major	DODTokenV2.sol: 583,5, 831,5	 Resolved

Description

`uniswapV2Router.addLiquidityETH{value: ethAmount}(address(this), tokenAmount, 0, 0, owner(), block.timestamp);`

The `addLiquidity` function calls the `uniswapV2Router.addLiquidityETH` function with the to address specified as `owner()` for acquiring the generated LP tokens from the DOD-WBNB pool.

As a result, over time the `_owner` address will accumulate a significant portion of LP tokens. If the `_owner` is an EOA (Externally Owned Account), mishandling of its private key can have devastating consequences to the project as a whole.

Remediation

We advise the to address of the `uniswapV2Router.addLiquidityETH` function call to be replaced by the contract itself, i.e. `address(this)`, and to restrict the management of the LP tokens within the scope of the contract's business logic. This will also protect the LP tokens from being stolen if the `_owner` account is compromised. In general, we strongly recommend centralized privileges or roles in the protocol to be improved via a decentralized mechanism or via smart-contract based accounts with enhanced security practices, f.e. Multisignature wallets.



1. Indicatively, here are some feasible solutions that would also mitigate the potential risk:
2. Time-lock with reasonable latency, i.e. 48 hours, for awareness on privileged operations;
3. Assignment of privileged roles to multi-signature wallets to prevent single point of failure due to the private key;

Introduction of a DAO / governance / voting module to increase transparency and user involvement

Project Action

Add liquidity function no longer exists.

DOD-05 | Missing Event Emission.

Category	Severity	Location	Status
Volatile Code	 Minor	DODTokenV2.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.

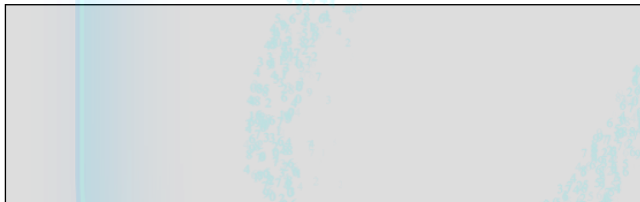
FOLLOW-UP: Event emissions have been implemented in most of the functions.

DOD-06 | Conformance with Solidity Naming Conventions.

Category	Severity	Location	Status
Coding Style	 Minor	DODTokenV2.sol: 77,5, 184,5	 Resolved

Description

Solidity defines a naming convention that should be followed. Rule exceptions: Allow constant variable name/symbol/decimals to be lowercase. Allow _ at the beginning of the mixed_case match for private variables and unused parameters.



Remediation



Follow the Solidity naming convention.

<https://docs.soliditylang.org/en/v0.4.25/style-guide.html#naming-convention>

Project Action

Names have been changed.

DOD-07 | State Variables could be Declared Constant.

Category	Severity	Location	Status
Coding Style	 Minor	DODTokenV2.sol: 64,5, 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-09 | Third Party Dependencies.

Category	Severity	Location	Status
Volatile Code	 Major	DODTokenV2.sol: 141,9	 Pending

Description

The contract is serving as the underlying entity to interact with third party FundPool & MarketingPool protocols.

The scope of the audit treats 3rd party entities as black boxes and assume their functional correctness.

However, in the real world, 3rd parties can be compromised and this may lead to lost or stolen assets.

In addition, upgrades of 3rd parties can possibly create severe impacts, such as increasing fees of 3rd parties, migrating to new LP pools, etc.

Remediation



We understand that the business logic of Day of Defeat requires interaction with FundPool & MarketingPool, etc.

We encourage the team to constantly monitor the statuses of 3rd parties to mitigate the side effects when unexpected activities are observed.

Project Action

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.

DOD-10 | Initial Token Distribution.

Category	Severity	Location	Status
Centralization / Privilege	 Major	DODTokenV2.sol: 632,5	 Resolved

Description

All of the Day of Defeat tokens are sent to the contract deployer when deploying the contract.

This could be a centralization risk as the deployer can distribute tokens without obtaining the consensus of the community.



Remediation

We recommend the team to be transparent regarding the initial token distribution process, and the team shall make enough efforts to restrict the access of the private key.

Project Action

A separate genesis wallet has been implemented where all of the tokens get sent to.

DOD-11 | Max TX.

Category	Severity	Location	Status
transferLimit // Limit per transfer	 Major	DODTokenV2.sol: 20,5,122,13	 Pending

Description

According to the paper provided, the intention is to set max tx to 99.9% to ensure that there is a tiny amount of dust left when a holder completely sells his/her token holdings. However, it is currently set to 9.99%, instead of 99.9%, and it gets applied to buys as well, not just sells or wallet to wallet transfers.



Remediation

If this feature must be implemented to make the number of holders appear to be more presentable, then implement a logic to ONLY apply if the holder is attempting to completely sell/transfer his/her token holdings.

Project Action

transferLimit has been updated to 9999 or 99.99%. However, it's still being applied to not only sells and transfers, but also buys as well. Even though it may only be a .01%, considering the project concept and how .01% worth of tokens of a trade can later worth hundreds and thousands of dollars, applying this transferLimit only to sells and wallet to wallet transfers is strongly recommended. Rather than applying a common divisor of 10000, recommend applying a divisor of 1,000,000,000 or greater and setting the transferLimit accordingly to add precision and leave a truly minuscule amount of dust.

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

Category	Severity	Location	Status
Centralization / Privilege	 Informational	DODTokenV2.sol: 327,5, 334,5	 Pending

Description

In the contract DODTokenV2, the role onlyOwner has authority over the functions that lead to centralization risks.

Any compromise to the onlyOwner 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

Main contract's centralization risk with onlyOwner has been reduced down to excluding accounts from fees.

DOD-13 | Extra Gas Cost For User

Category	Severity	Location	Status
Logical Issue	 Informational	DODTokenV2.sol: 141,9	 Pending

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

Swap and liquify no longer exists. Instead, autoSwapInPool function exists to replace the serve the purpose of the previous swap and liquify function with another limit of swap interval that lessens the frequency of making an individual users bear a burden of extra gas cost from the contract swapping.

However, please also be aware that calling on two different contract's functions at every transfer/trade will increase the gas fee burden on the investors.

DOD-14 | Unnecessary Use Of SafeMath

Category	Severity	Location	Status
Logical Issue	 Informational	DODTokenV2.sol: 5,1, 41,1	 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 DayofdefeatToken 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	DODTokenV2.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	80/100
Auditor Score	74/100
Review by Section	Score
Manual Scan Score	12/18
SWC Scan Score	36 /37
Advance Check Score	32 /45

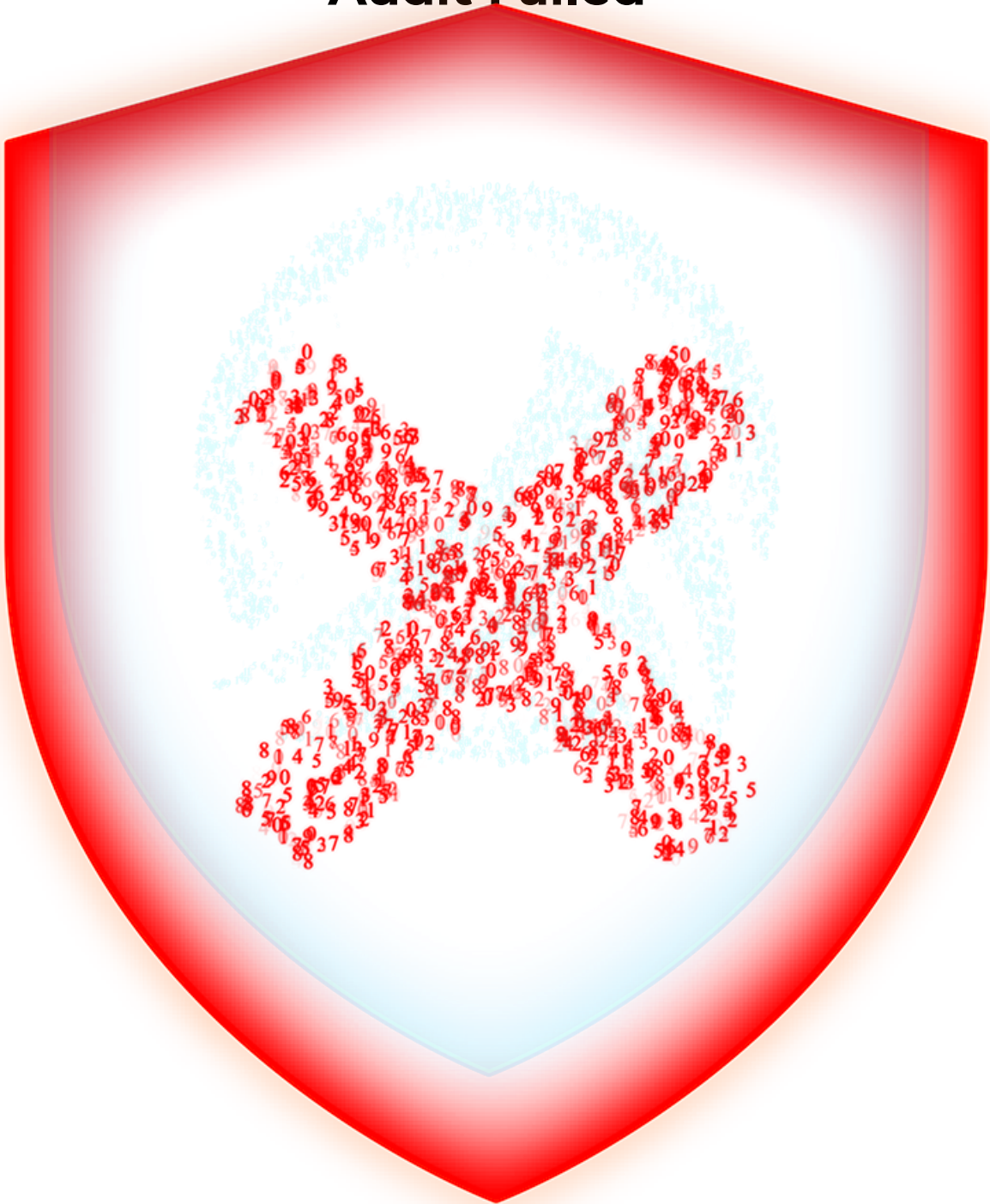
The maximum score is 100, however to attain that value the project must pass the reviews and provide all the data needed for the assessment. Minimum score to pass is 80 points. If a project fails to attain 80 and/or has unresolved critical and/or major and/or medium finding(s) in the Palladium tier assessments,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

Audit Fail



Assessment Results

Auditor Score = 74
Audit Failed



Important Notes from the Auditor:

- 3rd Review(NEW): 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.

- 2nd Review: Max Transfer is set to 9.99%. The provided paper shows the intention is to set the max tx to 99.9% to ensure some dust is left over.
- 3rd Review: "transferLimit" has been updated to 9999 or 99.99%. However, it's still being applied to not only sells and transfers, but also buys as well. Even though it may only be a .01%, considering the project concept and how .01% worth of tokens of a trade can later worth hundreds and thousands of dollars, applying this transferLimit only to sells and wallet to wallet transfers is strongly recommended. Rather than applying a common divisor of 10000, recommend applying a divisor of 1,000,000,000 or greater and setting the transferLimit accordingly to add precision and leave a truly minuscule amount of dust.
- 1st Review: Use of the most up-to-date compiler version is recommended to avoid known bugs and chances of exploits.

- 2nd Review: Updated to the latest compiler version.
- 1st Review: There is a fee of 19% and cannot be changed.
- 2nd Review: There is a tax of 19% which can be changed up to a maximum of 30% only by their DAO.
- 1st Review: The owner can ban a user with the function `setBlacklist`.
- 2nd Review: A blacklist function no longer exists.
- 1st Review: A complete audit cannot be done as key information behind the custom interface, IDao is missing.
- 2nd Review: All necessary files have been provided.
- 1st Review: Division before multiplication will result in a loss of precision in arithmetic calculations, which can lead to a significant loss in assets.
- 2nd Review: All arithmetic equations have been updated to do multiplication before division.

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.

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.

Volatile Code

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

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.

Inconsistency

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

Coding Best Practices

ERC 20 Coding Standards are a set of rules that each developer should follow to ensure the code meets a set of criteria 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 advocacy 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.

