



Smart Contract Audits | KYC



PALLADIUM

Security Assessment

Day of Defeat Governance

December 15, 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 Governance 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	1	0
● Major	2	2	0
● Medium	0	0	0
● Minor	2	2	0
● Informational	0	0	0
Total	5	5	0

Project Overview

Contract Summary

Parameter	Result
Address	
Name	Day of Defeat
Token Tracker	Day of Defeat (DODNFT)
Decimals	
Supply	
Platform	BNB Chain
compiler	v0.8.0^
Contract Name	DODGovernor
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	0x9eB5E406810E006B159A2B3E4970349fE0aEd529	No

Solidity Code Provided

SolID	File Sha-1	FileName
DODGovernor	3e7fa35c30ba499b409eee694fe1701e1c96b749	DODGovernor.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 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	DODGovernor.sol	L: 0 C: 0
SWC-101	Pass	Integer Overflow and Underflow.	DODGovernor.sol	L: 0 C: 0
SWC-102	Pass	Outdated Compiler Version file.	DODGovernor.sol	L: 0 C: 0
SWC-103	Low	A floating pragma is set.	DODGovernor.sol	L: 2 C: 3
SWC-104	Pass	Unchecked Call Return Value.	DODGovernor.sol	L: 0 C: 0
SWC-105	Pass	Unprotected Ether Withdrawal.	DODGovernor.sol	L: 0 C: 0
SWC-106	Pass	Unprotected SELFDESTRUCT Instruction	DODGovernor.sol	L: 0 C: 0
SWC-107	Pass	Read of persistent state following external call.	DODGovernor.sol	L: 0 C: 0
SWC-108	Pass	State variable visibility is not set..	DODGovernor.sol	L: 0 C: 0
SWC-109	Pass	Uninitialized Storage Pointer.	DODGovernor.sol	L: 0 C: 0
SWC-110	Pass	Assert Violation.	DODGovernor.sol	L: 0 C: 0
SWC-111	Pass	Use of Deprecated Solidity Functions.	DODGovernor.sol	L: 0 C: 0
SWC-112	Pass	Delegate Call to Untrusted Callee.	DODGovernor.sol	L: 0 C: 0

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

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

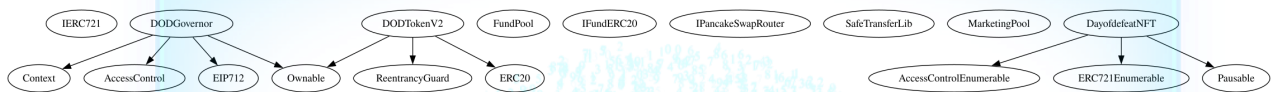
Pragma statements can be allowed to float when a contract is intended for consumption by other developers, as in the case with contracts in a library or EthPM package. Otherwise, the developer would need to manually update the pragma in order to compile locally.

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
---------------	------------	------------



DODNFT-02 | Function Visibility Optimization.

Category	Severity	Location	Status
Gas Optimization	● Minor	DODGovernor.sol:	 Pending

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
getActions		public
hashProposal		public
state		public
propose		public
execute		public
cancel		public
getUserVotes		public
castVote		public
castVoteWithReason		public
castVoteBySig		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.

References:

external vs public best practices.



DODNFT-03 | Lack of Input Validation.

Category	Severity	Location	Status
Volatile Code	 Minor	DODGovernor.sol: 492,5, 504,5, 517,5	 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

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

DODNFT-11 | If Statements.

Category	Severity	Location	Status
voteType == VoteTyp e.For	 Critical	DODGovernor.sol: 303,13, 305,13, 307,13	 Pending

Description

Three IF statements are all validating the same.



Remediation

Please review what this function is to accomplish. Perhaps two For's are supposed to be Against and Abstain.

Project Action

Pending Customer Response

DODNFT-12 | Centralization Risks In The ADMIN_ROLE Role(s)

Category	Severity	Location	Status
Centralization / Privilege	 Major	DODGovernor.sol: 145,9, 146,9, 147,9, 148,9	 Pending

Description

In the contract DODGovernor, the role ADMIN_ROLE has authority over the functions that lead to centralization risks.

Any compromise to the ADMIN_ROLE 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.

Furthermore, as the two functions, setVotePass() & voteForTakeOutToken() on DODTokenV2 contract, are called by a DOD Governor contract, this centralization risk can potentially lead to calling those two functions simply by ADMIN_ROLE account(s).

Project Action

Considering that this is a DAO governance contract, giving a power to add/remove/edit other roles by the ADMIN_ROLE is NOT recommended. Recommend the admin rights of the other roles to be given to the DAO, and implementing a function for DAO to vote on add/remove/edit of roles. Also clearly coding in how the two functions, setVotePass() & voteForTakeOutToken() on DODTokenV2 contract, gets called by other roles other than ADMIN_ROLE is strongly recommended.

DODNFT-15 | setGovernor.

Category	Severity	Location	Status
Centralization Risk	 Major	DODGovernor.sol: DODTokenV2 - 313,5	 Pending

Description

On DODTokenV2 contract, the Governor address gets set by the owner of DODTokenV2 contract, leaving a potential centralization risk of DODGovernor address getting switched by the owner at any time.

Remediation

Governor address should only be able to be set once by the deployer of DODTokenV2 contract and the right to change the address should be assigned to the DODGovernor contract.

Project Action

It is strongly recommended to set the Governor address in the constructor. And the governor address should be validated whether it's a smart contract or a regular wallet. The setGovernor function should have the modifier onlyGovernor. And DODGovernor contract should have a function to vote on changing the Governor address on DODTokenV2 by its DAO votes, clearly coded in on how the setGovernor function on DODTokenV2 contract gets called.

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

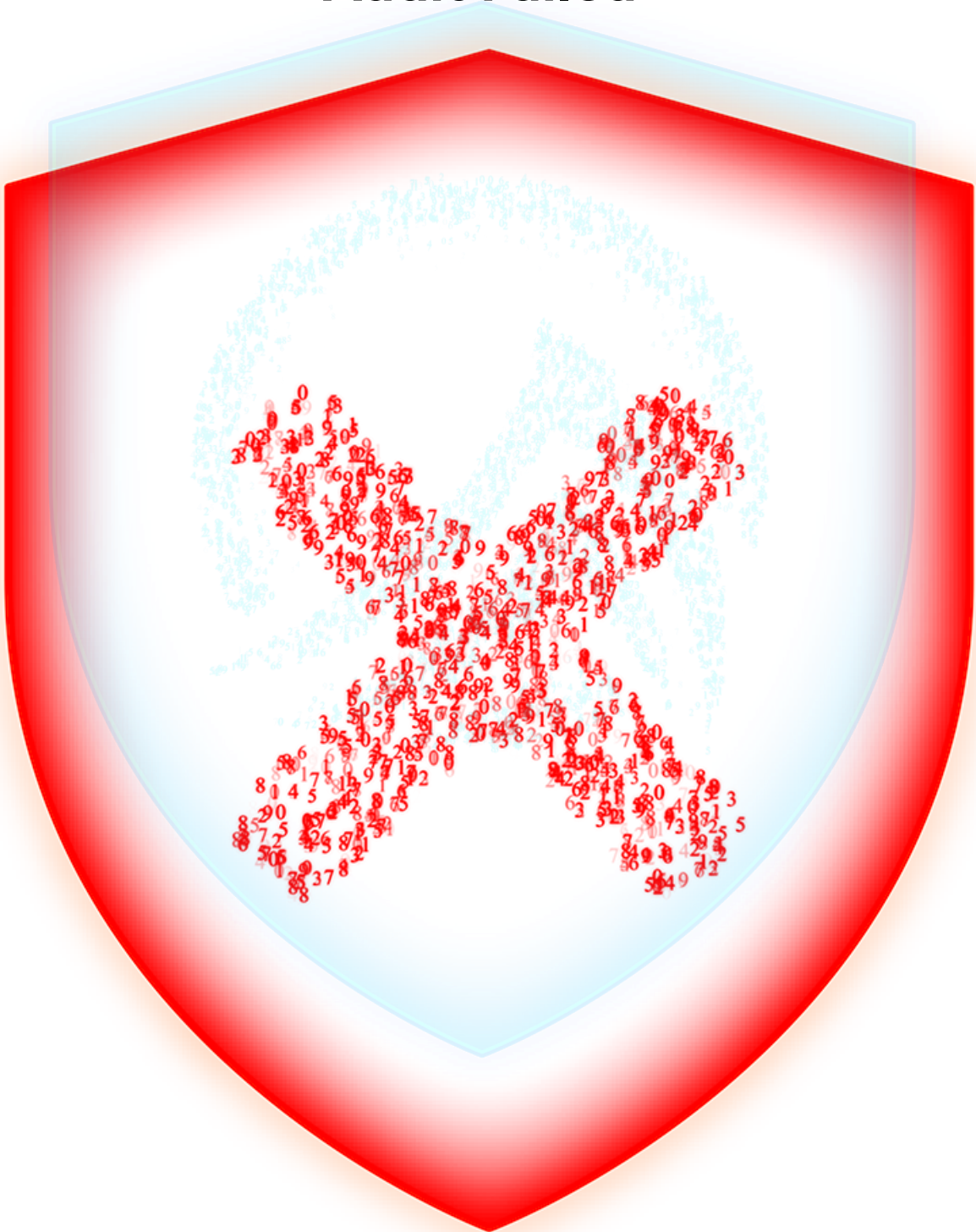
Review	Score
Overall Score	79/100
Auditor Score	77/100
Review by Section	Score
Manual Scan Score	15/18
SWC Scan Score	36 /37
Advance Check Score	28 /45

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 a critical and/or major finding(s) in the Palladium tier assessments, an automatic failure is given. Read our notes and final assessment below.



Assessment Results

Auditor Score = 77
Audit Failed



Important Notes from the Auditor:

- Considering that this is a DAO Governance contract, any potential centralization risks were carefully reviewed.
- The way ADMIN_ROLE is set as an admin role for the other roles declared left a concern of a centralization risk, defeating the purpose of the DAO governance.
- Also, how the Governor address gets set in the DODTokenV2 contract left another concern of a centralization risk, defeating the purpose of the DAO governance.
- The logic of how the two functions that are currently set to be called by the Governor contract isn't clearly coded on the contract. Which, combined with the centralization risk of ADMIN_ROLE, it left another concern of a centralization risk.
- Overall, the coding quality is commendable, following the recommended solidity coding principles to save gas and properly emitting events to be transparent.

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.

