



Smart Contract Audits | KYC



**PALLADIUM**

Security Assessment

**Day of Defeat Token**

January 12, 2023

# 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      | 1     | 0       | 1        |
| ● Major         | 4     | 0       | 4        |
| ● Medium        | 0     | 0       | 0        |
| ● Minor         | 4     | 0       | 4        |
| ● Informational | 4     | 3       | 1        |
| Total           | 13    | 3       | 10       |

# 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    | 79     |
| Confidence Level | High   |

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 | 0x4A10ADdC712744DD58f1A1103Df6943A7183868e | No   |

## Solidity Code Provided

| SolID      | File Sha-1                                   | FileName       |
|------------|--|----------------|
| DODTokenV2 | a6db539447ae63eb4a0dbe4dbe89eb6dd616f35<br>4 | 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 do not have the ability to set fees higher than 25%.**

**The team May have fees defined; however, they can't set those fees higher than 25% or may not be able to configure the same.**

**Tax Fee Notes:**

**Auditor Notes:** The contract does have a tax of 19%, but only can set up to a maximum of 25% and is controlled by its 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 cannot set max tx amount**

**The Team allows any investors to swap, transfer or sell their total amount if needed.**

**MaxTX Notes:**

**Auditor Notes:** TransferLimit only applies to sells to completely liquidate its holdings at 99.99999999% to leave a 0.00000001% dust.

**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](#)





1998, 1999, 2000, 2001, 2002, 2003, 2004, 2005, 2006, 2007, 2008, 2009, 2010, 2011, 2012, 2013, 2014, 2015, 2016, 2017, 2018, 2019, 2020, 2021, 2022, 2023, 2024, 2025, 2026, 2027, 2028, 2029, 2030, 2031, 2032, 2033, 2034, 2035, 2036, 2037, 2038, 2039, 2040, 2041, 2042, 2043, 2044, 2045, 2046, 2047, 2048, 2049, 2050, 2051, 2052, 2053, 2054, 2055, 2056, 2057, 2058, 2059, 2060, 2061, 2062, 2063, 2064, 2065, 2066, 2067, 2068, 2069, 2070, 2071, 2072, 2073, 2074, 2075, 2076, 2077, 2078, 2079, 2080, 2081, 2082, 2083, 2084, 2085, 2086, 2087, 2088, 2089, 2090, 2091, 2092, 2093, 2094, 2095, 2096, 2097, 2098, 2099, 2100, 2101, 2102, 2103, 2104, 2105, 2106, 2107, 2108, 2109, 2110, 2111, 2112, 2113, 2114, 2115, 2116, 2117, 2118, 2119, 2120, 2121, 2122, 2123, 2124, 2125, 2126, 2127, 2128, 2129, 2130, 2131, 2132, 2133, 2134, 2135, 2136, 2137, 2138, 2139, 2140, 2141, 2142, 2143, 2144, 2145, 2146, 2147, 2148, 2149, 2150, 2151, 2152, 2153, 2154, 2155, 2156, 2157, 2158, 2159, 2160, 2161, 2162, 2163, 2164, 2165, 2166, 2167, 2168, 2169, 2170, 2171, 2172, 2173, 2174, 2175, 2176, 2177, 2178, 2179, 2180, 2181, 2182, 2183, 2184, 2185, 2186, 2187, 2188, 2189, 2190, 2191, 2192, 2193, 2194, 2195, 2196, 2197, 2198, 2199, 2200, 2201, 2202, 2203, 2204, 2205, 2206, 2207, 2208, 2209, 2210, 2211, 2212, 2213, 2214, 2215, 2216, 2217, 2218, 2219, 2220, 2221, 2222, 2223, 2224, 2225, 2226, 2227, 2228, 2229, 2230, 2231, 2232, 2233, 2234, 2235, 2236, 2237, 2238, 2239, 2240, 2241, 2242, 2243, 2244, 2245, 2246, 2247, 2248, 2249, 2250, 2251, 2252, 2253, 2254, 2255, 2256, 2257, 2258, 2259, 2260, 2261, 2262, 2263, 2264, 2265, 2266, 2267, 2268, 2269, 2270, 2271, 2272, 2273, 2274, 2275, 2276, 2277, 2278, 2279, 2280, 2281, 2282, 2283, 2284, 2285, 2286, 2287, 2288, 2289, 2290, 2291, 2292, 2293, 2294, 2295, 2296, 2297, 2298, 2299, 2300, 2301, 2302, 2303, 2304, 2305, 2306, 2307, 2308, 2309, 2310, 2311, 2312, 2313, 2314, 2315, 2316, 2317, 2318, 2319, 2320, 2321, 2322, 2323, 2324, 2325, 2326, 2327, 2328, 2329, 2330, 2331, 2332, 2333, 2334, 2335, 2336, 2337, 2338, 2339, 2340, 2341, 2342, 2343, 2344, 2345, 2346, 2347, 2348, 2349, 2350, 2351, 2352, 2353, 2354, 2355, 2356, 2357, 2358, 2359, 2360, 2361, 2362, 2363, 2364, 2365, 2366, 2367, 2368, 2369, 2370, 2371, 2372, 2373, 2374, 2375, 2376, 2377, 2378, 2379, 2380, 2381, 2382, 2383, 2384, 2385, 2386, 2387, 2388, 2389, 2390, 2391, 2392, 2393, 2394, 2395, 2396, 2397, 2398, 2399, 2400, 2401, 2402, 2403, 2404, 2405, 2406, 2407, 2408, 2409, 2410, 2411, 2412, 2413, 2414, 2415, 2416, 2417, 2418, 2419, 2420, 2421, 2422, 2423, 2424, 2425, 2426, 2427, 2428, 2429, 2430, 2431, 2432, 2433, 2434, 2435, 2436, 2437, 2438, 2439, 2440, 2441, 2442, 2443, 2444, 2445, 2446, 2447, 2448, 2449, 2450, 2451, 2452, 2453, 2454, 2455, 2456, 2457, 2458, 2459, 2460, 2461, 2462, 2463, 2464, 2465, 2466, 2467, 2468, 2469, 2470, 2471, 2472, 2473, 2474, 2475, 2476, 2477, 2478, 2479, 2480, 2481, 2482, 2483, 2484, 2485, 2486, 2487, 2488, 2489, 2490, 2491, 2492, 2493, 2494, 2495, 2496, 2497, 2498, 2499, 2500, 2501, 2502, 2503, 2504, 2505, 2506, 2507, 2508, 2509, 2510, 2511, 2512, 2513, 2514, 2515, 2516, 2517, 2518, 2519, 2520, 2521, 2522, 2523, 2524, 2525, 2526, 2527, 2528, 2529, 2530, 2531, 2532, 2533, 2534, 2535, 2536, 2537, 2538, 2539, 2540, 2541, 2542, 2543, 2544, 2545, 2546, 2547, 2548, 2549, 2550, 2551, 2552, 2553, 2554, 2555, 2556, 2557, 2558, 2559, 2560, 2561, 2562, 2563, 2564, 2565, 2566, 2567, 2568, 2569, 2570, 2571, 2572, 2573, 2574, 2575, 2576, 2577, 2578, 2579, 2580, 2581, 2582, 2583, 2584, 2585, 2586, 2587, 2588, 2589, 2590, 2591, 2592, 2593, 2594, 2595, 2596, 2597, 2598, 2599, 2600, 2601, 2602, 2603, 2604, 2605, 2606, 2607, 2608, 2609, 2610, 2611, 2612, 2613, 2614, 2615, 2616, 2617, 2618, 2619, 2620, 2621, 2622, 2623, 2624, 2625, 2626, 2627, 2628, 2629, 2630, 2631, 2632, 2633, 2634, 2635, 2636, 2637, 2638, 2639, 2640, 2641, 2642, 2643, 2644, 2645, 2646, 2647, 2648, 2649, 2650, 2651, 2652, 2653, 2654, 2655, 2656, 2657, 2658, 2659, 2660, 2661, 2662, 2663, 2664, 2665, 2666, 2667, 2668, 2669, 2670, 2671, 2672, 2673, 2674, 2675, 2676, 2677, 2678, 2679, 26

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 | Context.sol, L: 2 C: 1 |
| 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              |

| ID      | Severity | Name   | File           | location  |
|---------|----------|--|----------------|-----------|
| SWC-112 | Pass     | Delegate Call to Untrusted Callee.   | DODTokenV2.sol | L: 0 C: 0 |
| 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:

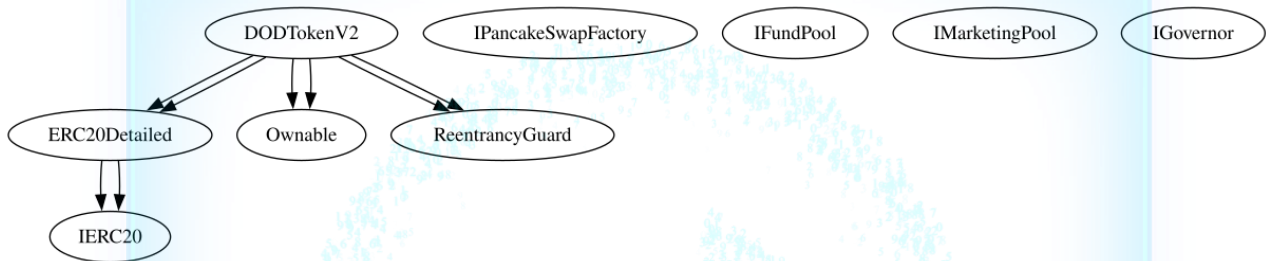
Ethereum Smart Contract Best Practices - Lock pragmas to specific compiler version.



# 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 |
|--------------------|--|------------|
| initializePool     | address  | External   |
| unlockFundPool     | _marketingPool,<br>address _fundPool                       | External   |
| includePairLimit   | address pair, bool<br>enable                               | External   |
| setSwapStatus      | bool enable  | External   |
| setTriggerInterval | uint256<br>_triggerInterval                                | External   |
| setExcludeFee      | address account,<br>bool enable                            | External   |
| batchSetExcludeFee | address[] calldata<br>accounts, bool[]<br>calldata enables | External   |

## DOD-02 | Function Visibility Optimization.

| Category         | Severity  | Location        | Status  |
|------------------|---|-----------------|---|
| Gas Optimization |  Informational | DODTokenV2.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 |
|-------------------|--|------------|
| decreaseAllowance | address spender, uint256 subtractedValue | 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.



7th: Please review new additional public functions and change those that can be changed to external. ie. functions that are calling totalSupply() may be replaced with \_totalSupply.

8th: Most of functions have been updated to external except decreaseAllowance.

### References:

external vs public best practices.

## DOD-03 | Lack of Input Validation.

| Category      | Severity  | Location                            | Status   |
|---------------|---|-------------------------------------|--|
| Volatile Code |  Minor | DODTokenV2.sol: 362,5, 370,5, 378,5 |  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. I.e. 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-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.

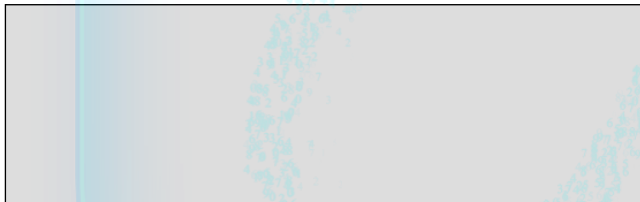


## DOD-06 | Conformance with Solidity Naming Conventions.

| 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: 79,5 |  Resolved |

### Description

Constant state variables should be declared constant to save gas.



BUSD

### 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 |  Resolved |

### 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

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.

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.

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.

6th: Successful automatic swaps were verified. However there is a standing issue of instances of sell transactions failing when either of the FundPool & MarketingPool already has met the threshold to complete a swap of tokens within the function of swapForFundPool & swapForMarketingPool, 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.

7th: Simple threshold of a fixed amount of tokens trigger has been implemented in both FundPool & MarketingPool contracts. Successful sell transactions with the contract triggered swaps in the contracts FundPool & MarketingPool verified.

## 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 |  Resolved |

### 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

3rd: 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 are 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.



4th: The codes have been updated for the transferLimit to only apply when it's only a sell. However, there are critical errors in the variables involved. Also, it's being applied to every sell, not just those sell attempts to completely empty its holdings.

5th: The codes have been updated for the transferLimit to only apply when it's only an attempt to completely liquidate its holdings. However, there are still critical errors in the variables involved (exponents) and could not be tested.

6th: All of the concerning variables have been fixed.



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

| Category                   | Severity  | Location              | Status  |
|----------------------------|---|-----------------------|---|
| Centralization / Privilege |  Informational | DODTokenV2.sol: 156,6 |  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.

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

## DOD-13 | Extra Gas Cost For User

| Category      | Severity  | Location                      | Status  |
|---------------|---|-------------------------------|---|
| Logical Issue |  Informational | DODTokenV2.sol: 361,9, 362,13 |  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

4th: 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.



5th: 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.

6th: The function autoSwapInPool has been updated to call only either one of swapForFundPool and swapForMarketingPool at a time to reduce the gas fee burden on the investors. However, testings revealed that even when a FundPool or MarketingPool swap isn't triggered, the gas limit is approximately 33% higher than the current V1 contract, and when a FundPool or MarketingPool swap is triggered, the gas limit is approximately 137% higher. RECOMMENDATION: If the concern on gas fee burden on the investors still do stand, do consider whether separate contracts of FundPool and MarketingPool are absolutely necessary. The purpose of FundPool and MarketingPool are ultimately to temporarily hold the DOD tokens until the threshold is met, then burn DOD or sell DOD to BUSD/BNB to send to the Token Contract or the marketing receiver. (1) Can this be done in the token contract to eliminate the extra gas fee coming from interacting with another smart contract? (2) Can the frequency of sending DOD tokens to the FundPool & MarketingPool, and the frequency of calling swapForFundPool & swapForMarketingPool be limited by setting thresholds?

7th: The function autoSwapInPool has been updated to call only every set triggerInterval (default 20 mins) AND only either one of swapForFundPool and swapForMarketingPool

at a time to reduce the gas fee burden on the investors. The gas limit has been improved for buys/sells, approximately 10% lower than the current V1 contract. When a FundPool or MarketingPool swap is triggered, the gas limit is approximately 50~130% higher depending on if a swap is occurring or if just transfers are occurring, compared to 137% higher in the previous contract. RECOMMENDATION: If the concern about the gas fee burden on the investors still does stand, do consider whether separate contracts of FundPool and MarketingPool are absolutely necessary. The purpose of FundPool and MarketingPool is ultimately to temporarily hold the DOD tokens until the threshold is met, then burn DOD or sell DOD to BUSD/BNB to send to the Token Contract or the marketing receiver. (1) Can this be done in the token contract to eliminate the extra gas fee coming from interacting with another smart contract? (2) Can the frequency of sending DOD tokens to the FundPool & MarketingPool, and the frequency of calling swapForFundPool & swapForMarketingPool be limited by setting thresholds?

## 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      | <a href="https://www.dayofdefeat.app/">https://www.dayofdefeat.app/</a>                   | Pass   |
| Telegram     | <a href="https://t.me/DayOfDefeatBSC">https://t.me/DayOfDefeatBSC</a>                     | Pass   |
| Twitter      | <a href="https://twitter.com/dayofdefeatBSC">https://twitter.com/dayofdefeatBSC</a>       | Pass   |
| OtherSocial  | <a href="https://titanservice.cn/dayofdefeatCN">https://titanservice.cn/dayofdefeatCN</a> | 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

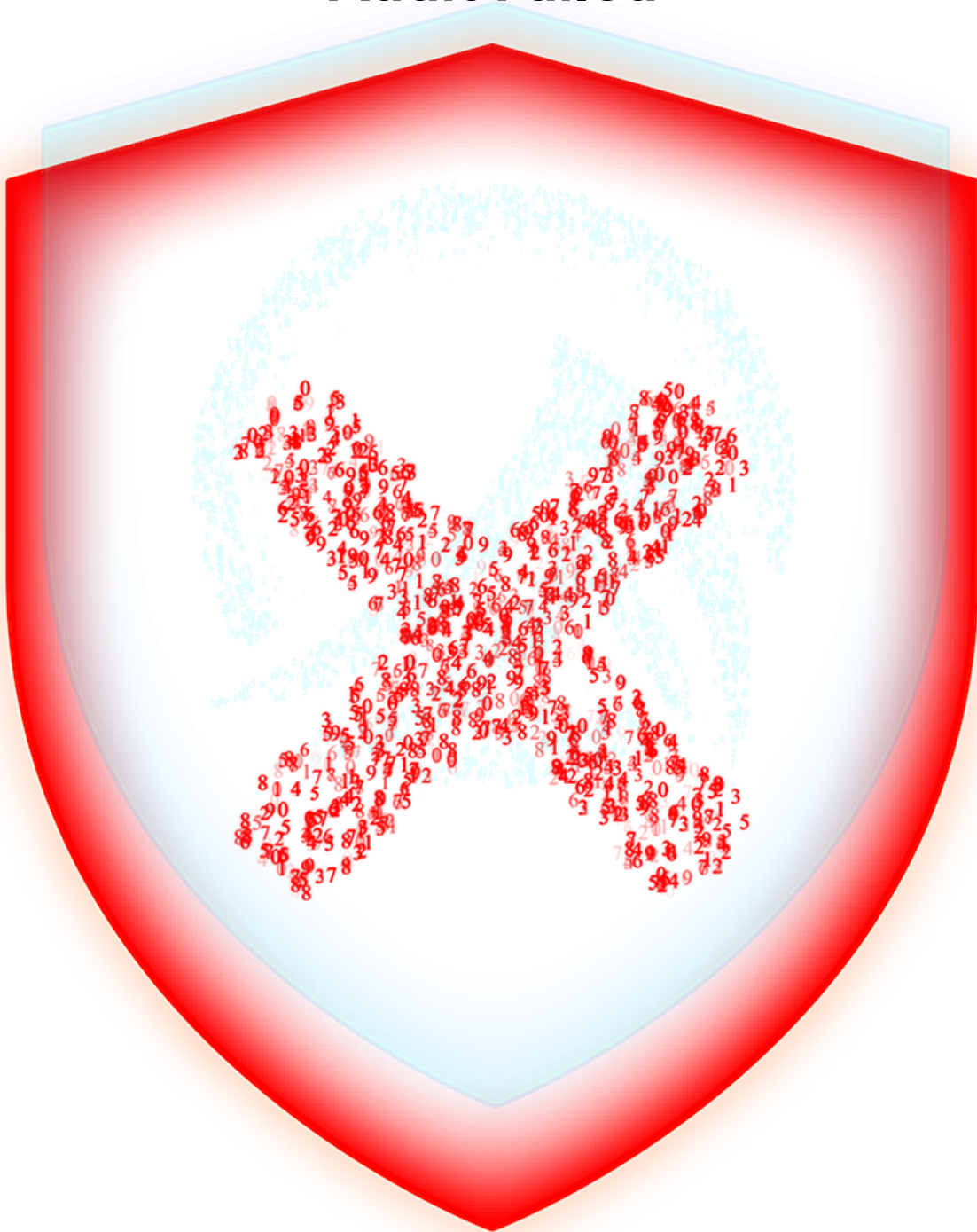
| Review              | Score  |
|---------------------|--------|
| Overall Score       | 92/100 |
| Auditor Score       | 79/100 |
| Review by Section   | Score  |
| Manual Scan Score   | 14/18  |
| SWC Scan Score      | 36/37  |
| Advance Check Score | 42/45  |

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 = 79**  
**Audit Failed**



## Important Notes from the Auditor:

- 8th(NEW): New Concern #1) Is it fair for the participants to keep the trading open even when the rewards pool has been unlocked and the exchange ratio has been fixed, and allow the new participants to enter and exchange the newly bought DOD to BUSD?
- 8th(NEW): New Concern #2) Although 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 to calculate the exchange ratio when all 3 conditions aren't met will not properly function.
- 8th(NEW): function `burnFrom` removed. Concerns #1 & #2 have been addressed by the swap ratio to be 10 DOD:1 BUSD only when all 3 conditions are met, and if not, the newly implemented logic will calculate the exchange ratio in the proportion of the remaining supply of DOD:BUSD in the rewards pool at the time.
- 7th: functions `burnFrom` & `swap` have NOT been updated yet. Concerns #1 & #2 still stand.

- 6th: The function burnFrom fails to execute because of line 219. Please review.
- 6th: Rewards pool unlock has been simulated and tested. The swap function failed due to the error, ERC20: insufficient allowance. Manual token approval with the spender being the token address was necessary for the swap to succeed. The swap did succeed in the proportion of 10 DOD:1 BUSD, with a couple of concerns:
  - 6th: Concern #1) The condition was met with 99,999,000 BUSD in the rewards pool with 4.3m DOD tokens burnt or 99,999,995,700,000 DOD token supply remaining. Which then allowed one wallet with over 100m DOD tokens to exchange every single BUSD in the pool, while there are multiple wallets with over 100m DOD tokens standing. While this scenario happening in the real world may be slim, the question remains "what if"?
  - 6th: Concern #2) The paper provided showed that when in the case the unlock condition does not meet by November 18, 2026, and the DAO votes to proceed to unlock the rewards pool, the exchange ratio is supposed to be set in the proportion of the remaining supply of DOD:BUSD in the rewards pool at the time. However, the codes that would calculate the proportion if the aforementioned scenario occurs were not found. Please review.



- 
- 8th(NEW): With new additional codes in the function `_transfer`, the gas limit is now
- 7th: The function `autoSwapInPool` has been updated to call only every set `triggerInterval` (default 20 mins) AND only either one of `swapForFundPool` and `swapForMarketingPool` at a time to reduce the gas fee burden on the investors. The gas limit has been improved for buys/sells, approximately 10% lower than the current V1 contract. When a `FundPool` or `MarketingPool` swap is triggered, the gas limit is approximately 50~130% higher depending on if a swap is occurring or if just transfers are occurring, compared to 137% higher in the previous contract. RECOMMENDATION: If the concern about the gas fee burden on the investors still does stand, do consider whether separate contracts of `FundPool` and `MarketingPool` are absolutely necessary. The purpose of `FundPool` and `MarketingPool` is ultimately to temporarily hold the DOD tokens until the threshold is met, then burn DOD or sell DOD to BUSD/BNB to send to the Token Contract or the marketing receiver. (1) Can this be done in the token contract to eliminate the extra gas fee coming from interacting with another smart contract? (2) Can the frequency of sending DOD tokens to the `FundPool` &

MarkeetingPool, and the frequency of calling swapForFundPool & swapForMarketingPool be limited by setting thresholds?

- 6th: The function autoSwapInPool has been updated to call only either one of swapForFundPool and swapForMarketingPool at a time to reduce the gas fee burden on the investors. However, testings revealed that even when a FundPool or MarketingPool swap isn't triggered, the gas limit is approximately 33% higher than the current V1 contract, and when a FundPool or MarketingPool swap is triggered, the gas limit is approximately 137% higher. RECOMMENDATION: If the concern on gas fee burden on the investors still do stand, do consider whether separate contracts of FundPool and MarketingPool are absolutely necessary. The purpose of FundPool and MarketingPool are ultimately to temporarily hold the DOD tokens until the threshold is met, then burn DOD or sell DOD to BUSD/BNB to send to the Token Contract or the marketing receiver. (1) Can this be done in the token contract to eliminate the extra gas fee coming from interacting with another smart contract? (2) Can the frequency of sending DOD tokens to the FundPool & MarkeetingPool, and the frequency of calling swapForFundPool & swapForMarketingPool be limited by



setting thresholds?

- 
- 7th: Simple threshold of a fixed amount of tokens trigger has been implemented in both FundPool & MarketingPool contracts. Successful sell transactions with the contract triggered swaps in the contracts FundPool & MarketingPool verified.
- 6th: Successful automatic swaps were verified. However there is a standing issue of instances of sell transactions failing when either of the FundPool & MarketingPool already has met the threshold to complete a swap of tokens within the function of swapForFundPool & swapForMarketingPool, 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.
- 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.

- 

- 6th: All of the concerning variables have been fixed.
- 5th: CHECK: genesisTotalSupply, BASE, marketingFee, transitionFee, maxFee, transferLimit, bonusPoolTrigger, totalRemainingTrigger, totalBurnedTrigger, dodToBusdMultiplier. Most of the declared variables with exponents have critical errors, and some are using exponents unnecessarily. Please review.
- 4th: Most of the declared variables with exponents have critical errors, and some are using exponents unnecessarily. Please review.

- 

- 5th: Follow-Up: The codes have been updated for the transferLimit to only apply when it's only an attempt to completely liquidate its holdings. However, there are still

critical errors in the variables involved (exponents) and the contract could not be tested.

- 4th: The codes have been updated for the transferLimit to only apply when it's only a sell. However, there are critical errors in the variables involved.
- 3rd: "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 sellings and wallet to wallet transfers are 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.
- 2nd: 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.
- 
- 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: There is a tax of 19% which can be changed up to a maximum of 30% only by their DAO.
- 1st: There is a fee of 19% and cannot be changed.
- 
- 2nd: A blacklist function no longer exists.
- 1st: The owner can ban a user with the function setBlacklist.
- 
- 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 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.

