CONTRACT WOLF

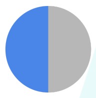
Security Assessment

Myth of Dagda

Verified on 11/16/2023

SUMMARY

Project	CHAIN	METHODOLOGY
Myth of Dagda	Binance Smart Chain	Manual & Automatic Analysis
FILES	DELIVERY	TYPE
Single	11/16/2023	Standard Audit



Total Findings	Critical	Major	Medium	Minor	Informational
0	0	0	0	1	1

🟥 0 Critical	An exposure that can affect the contract functions in several events that can risk and disrupt the contract
🟧 0 Major	An exposure that can affect the outcome when using the contract that can serve as an opening in manipulating the contract in an unwanted manner
🟧 0 Medium	An opening that could affect the outcome in executing the contract in a specific situation
⬜ 1 Minor	An opening but doesn't have an impact on the functionality of the contract
🟦 1 Informational	An opening that consists information but will not risk or affect the contract

STATUS ✔️**AUDIT PASSED**

TABLE OF CONTENTS | Myth of Dagda

DISCLAIMER | Myth of Dagda

ContractWolf audits and reports should not be considered as a form of project's "Advertisement" and does not cover any interaction and assessment from "Project Contract" to "External Contracts" such as PancakeSwap, UniSwap, SushiSwap or similar.

ContractWolf does not provide any warranty on its released report and should not be used as a decision to invest into audited projects.

ContractWolf provides a transparent report to all its "Clients" and to its "Clients Participants" and will not claim any guarantee of bug-free code within its **SMART CONTRACT**.

ContractWolf's presence is to analyze, audit and assess the Client's Smart Contract to find any underlying risk and to eliminate any logic and flow errors within its code.

Each company or project should be liable to its security flaws and functionalities.

SCOPE OF WORK | Myth of Dagda

Myth of Dagda team has agreed and provided us with the files that need to be tested (*Github, BSCscan, Etherscan, Local files etc*). The scope of audit is the main contract.

The goal of this engagement is to identify if there is a possibility of security flaws in the implementation of smart contract and its systems.

ContractWolf will be focusing on contract issues and functionalities along with the project claims from smart contract to their website, whitepaper, repository which has been provided by **Myth of Dagda**.

AUDITING APPROACH | Myth of Dagda

Every line of code along with its functionalities will undergo manual review to check for security issues, quality of logic and contract scope of inheritance. The manual review will be done by our team that will document any issues that they discovered.

METHODOLOGY

The auditing process follows a routine series of steps :

1. Code review that includes the following :
   - Review of the specifications, sources and instructions provided to ContractWolf to make sure we understand the size, scope and functionality of the smart contract.
   - Manual review of code. Our team will have a process of reading the code line-by-line with the intention of identifying potential vulnerabilities, underlying and hidden security flaws.

2. Testing and automated analysis that includes :
   - Testing the smart contract function with common test cases and scenarios to ensure that it returns the expected results.

3. Best practices and ethical review. The team will review the contract with the aim to improve efficiency, effectiveness, clarifications, maintainability, security and control within the smart contract.

4. Recommendations to help the project take steps to eliminate or minimize threats and secure the smart contract.

TOKEN DETAILS | Myth of Dagda

A figure of immense power. The Good God. Leader of the Tuatha Dé Danann. Mastery over all things magical and mystical.

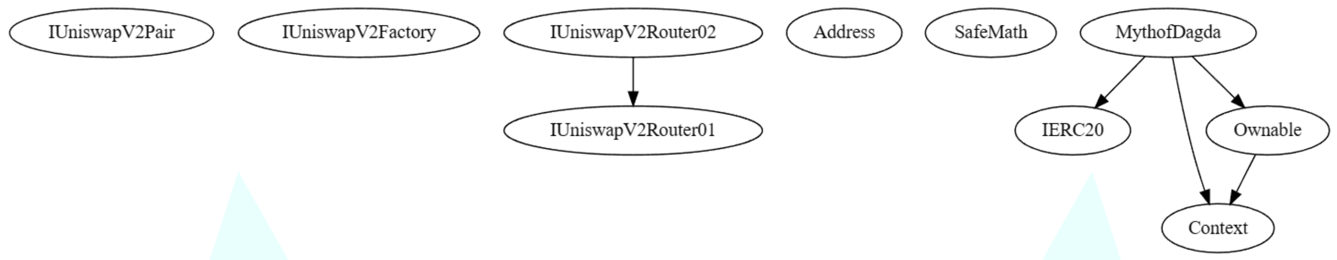
Token Name	Symbol	Decimal	Total Supply	Chain
Myth of Dagda	DAGDA	18	100,000,000	Binance Smart Chain

SOURCE

Source	*0x7455DE16f316A262830858F44CA8970b118Cd2De*

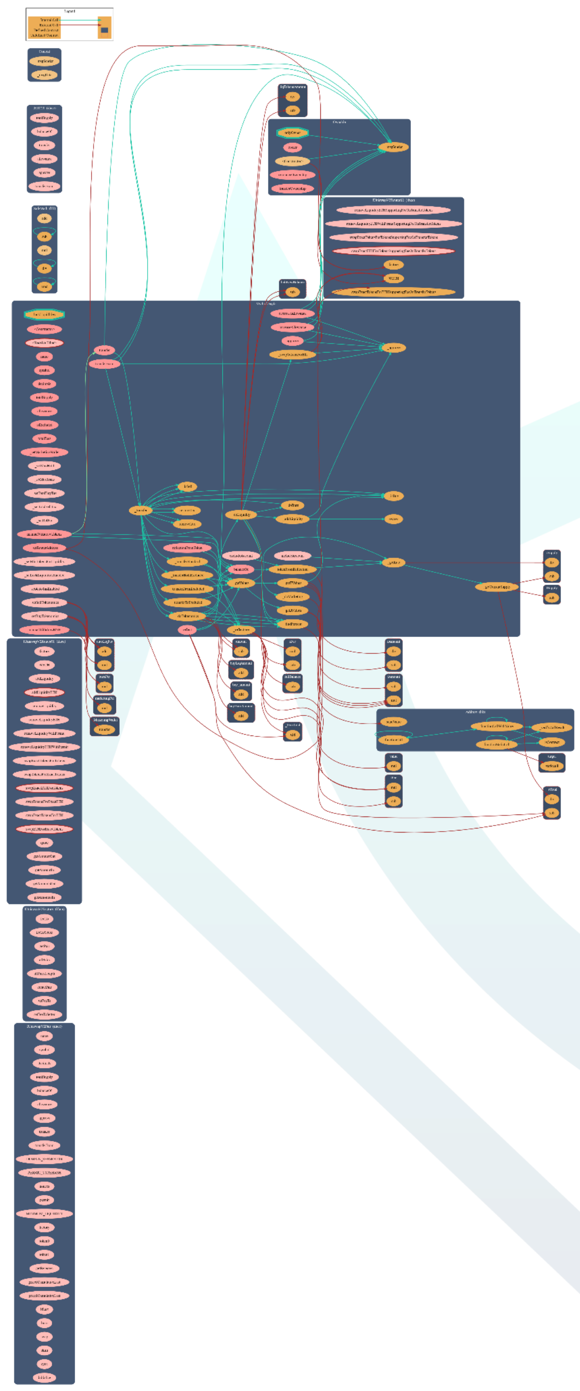
INHERITANCE GRAPH | Myth of Dagda

Inheritance Graph of Contract Functions

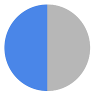


CALL GRAPH | Myth of Dagda

Call Graph of Contract Functions



FINDINGS | Myth of Dagda



0	0	0	0	1	1
Total Findings	Critical	Major	Medium	Minor	Informational

This report has been prepared to state the issues and vulnerabilities for Myth of Dagda through this audit. The goal of this report findings is to identify specifically and fix any underlying issues and errors

ID	Title	File & Line #	Severity	Status
SWC-102	Outdated Compiler Version	MythofDagda.sol	Minor	● Pending
SWC-103	Floating Pragma is set	MythofDagda.sol	Informational	● Pending

SWC ATTACKS | Myth of Dagda

Smart Contract Weakness Classification and Test Cases

ID	Description	Status
SWC-100	Function Default Visibility	● Passed
SWC-101	Integer Overflow and Underflow	● Passed
SWC-102	Outdated Compiler Version	● Not Passed
SWC-103	Floating Pragma	● Not Passed
SWC-104	Unchecked Call Return Value	● Passed
SWC-105	Unprotected Ether Withdrawal	● Passed
SWC-106	Unprotected SELF DESTRUCT Instruction	● Passed
SWC-107	Reentrancy	● Passed
SWC-108	State Variable Default Visibility	● Passed
SWC-109	Uninitialized Storage Pointer	● Passed
SWC-110	Assert Violation	● Passed
SWC-111	Use of Deprecated Solidity Functions	● Passed
SWC-112	Delegatecall to Untrusted Callee	● Passed
SWC-113	DoS with Failed Call	● Passed
SWC-114	Transaction Order Dependence	● Passed
SWC-115	Authorization through tx.origin	● Passed
SWC-116	Block values as a proxy for time	● Passed
SWC-117	Signature Malleability	● Passed
SWC-118	Incorrect Constructor Name	● Passed
SWC-119	Shadowing State Variables	● Passed
SWC-120	Weak Sources of Randomness from Chain Attributes	● Passed
SWC-121	Missing Protection against Signature Replay Attacks	● Passed
SWC-122	Lack of Proper Signature Verification	● Passed

ID	Description	Status
SWC-123	Requirement Violation	● Passed
SWC-124	Write to Arbitrary Storage Location	● Passed
SWC-125	Incorrect Inheritance Order	● Passed
SWC-126	Insufficient Gas Griefing	● Passed
SWC-127	Arbitrary Jump with Function Type Variable	● Passed
SWC-128	DoS With Block Gas Limit	● Passed
SWC-129	Typographical Error	● Passed
SWC-130	Right-To-Left-Override control character(U+202E)	● Passed
SWC-131	Presence of unused variables	● Passed
SWC-132	Unexpected Ether balance	● Passed
SWC-133	Hash Collisions With Multiple Variable Arguments	● Passed
SWC-134	Message call with hardcoded gas amount	● Passed
SWC-135	Code With No Effects	● Passed
SWC-136	Unencrypted Private Data On-Chain	● Passed

CW ASSESSMENT | Myth of Dagda

ContractWolf Vulnerability and Security Tests

ID	Name	Description	Status
CW-001	Multiple Version	Presence of multiple compiler version across all contracts	✓
CW-002	Incorrect Access Control	Additional checks for critical logic and flow	✓
CW-003	Payable Contract	A function to withdraw ether should exist otherwise the ether will be trapped	✓
CW-004	Custom Modifier	major recheck for custom modifier logic	✓
CW-005	Divide Before Multiply	Performing multiplication before division is generally better to avoid loss of precision	✓
CW-006	Multiple Calls	Functions with multiple internal calls	✓
CW-007	Deprecated Keywords	Use of deprecated functions/operators such as block.blockhash() for blockhash(), msg.gas for gasleft(), throw for revert(), sha3() for keccak256(), callcode() for delegatecall(), suicide() for selfdestruct(), constant for view or var for actual type name should be avoided to prevent unintended errors with newer compiler versions	✓
CW-008	Unused Contract	Presence of an unused, unimported or uncalled contract	✓
CW-009	Assembly Usage	Use of EVM assembly is error-prone and should be avoided or double-checked for correctness	✓
CW-010	Similar Variable Names	Variables with similar names could be confused for each other and therefore should be avoided	✓
CW-011	Commented Code	Removal of commented/unused code lines	✓
CW-012	SafeMath Override	SafeMath is no longer needed starting with Solidity v0.8+. The compiler now has built-in overflow checking.	✓

FIXES & RECOMMENDATION

SWC-102 | Outdated Compiler Version

Code

```
pragma solidity ^0.8.17;
```

Using an outdated compiler version can be problematic especially if there are publicly disclosed bugs and issues that affect the current compiler version.

```
pragma solidity 0.8.17;
```

SWC-103 | A Floating Pragma is Set

Code

```
pragma solidity 0.6.12;
```

The compiler version should be a fixed one to avoid undiscovered compiler bugs. Fixed version sample below

```
pragma solidity 0.8.21;
```

AUDIT COMMENTS | Myth of Dagda

Smart Contract audit comment for a non-technical perspective

🟢 Owner can renounce and transfer ownership
🟢 Owner can exclude/include addresses from reflections
🟢 Owner can change marketing wallet address
🟢 Owner can update max wallet and max transaction amount
🟢 Owner can update total buy and sell tax up to 25% each
🟢 Owner can set the percentages for auto-liquidity, burning, and marketing for both buying and selling transactions
🟢 Owner can update reflections percentage fee with an indefinite amount
🟢 Owner can update minimum tokens for liquidity
🟢 Owner can toggle locking of liquidities
🟢 Owner can toggle automatic taking of tax
🟢 Owner can update router and pair address
🟢 Owner can withdraw tokens and BNB from contract
🟢 Owner cannot block users
🟢 Owner cannot mint after initial deployment
🟢 Owner cannot burn
🟢 Owner cannot pause contract

CONTRACTWOLF

Blockchain Security - Smart Contract Audits