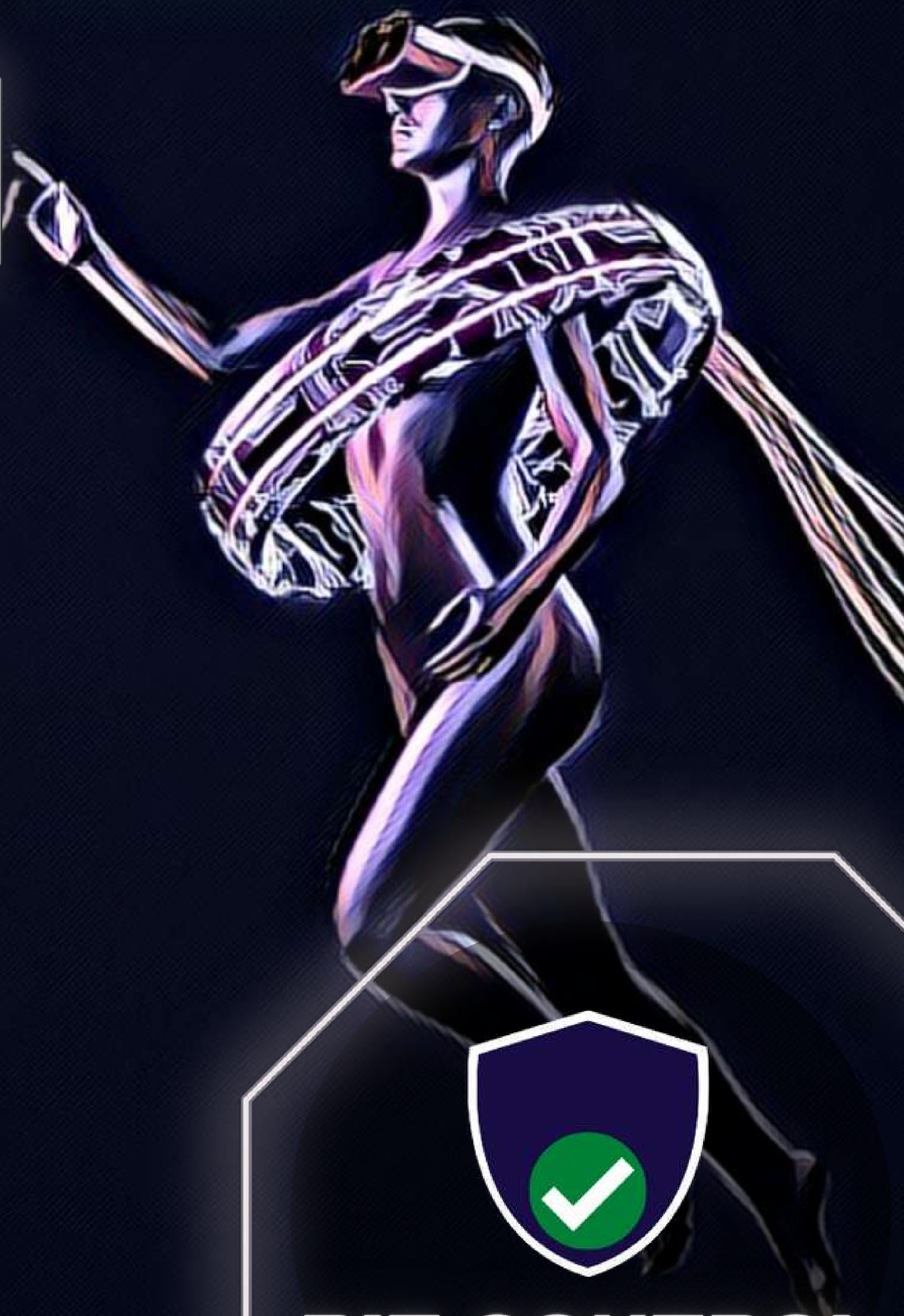
SMART CONTRACT AUDIT

DEFLATIONARY



BIT CONTROL

AUDIT & KYC SERVICES

@BIT_CONTROL

Summary

Auditing Firm	Bit Control
Architecture	Bit Control Auditing Standard
Smart Contract Audit Approved By	Nuno | Blockchain Specialist at Bit Control
Project Overview Approved By	Ricardo | Marketing Specialist at Bit Control
Platform	Solidity
Mandatory Audit Check	Static, Software & Manual Analysis
Consultation Request Date	January 12, 2022
Report Date	January 13, 2022

Audit Summary

Bit Control team has performed a line-by-line manual analysis and automated review of the smart contract. The smart contract was analyzed mainly for common smart contract vulnerabilities, exploits, and manipulation hacks. According to the smart contract audit:

- ★ Deflationary Token smart contract source code has **LOW RISK SEVERITY**.
- ★ Deflationary Token has **PASSED** the smart contract audit.

For the detailed understanding of risk severity, source code vulnerability, and functional test, kindly refer to the audit.

✅ Verify the authenticity of this report on Bit Control Website: https://www.bit-control.com/

Table Of Contents

Project Overview 3

Audit Scope & Methodology 4

- Bit Control Audit Standard 5
- Bit Control's Risk Classification 6

Smart Contract Risk Assessment 7

- Static Analysis 7
- Software Analysis 10
- SWC Attacks 11
- Risk Status 12

Report Summary 13

Legal Advisory 14

- Important Disclaimer 14
- About Bit Control 15

Project Overview

Bit Control was consulted by Deflationary Token to conduct the smart contract security audit of their solidity source code.

Project	Deflationary Token
Blockchain	Binance Smart Chain
Language	Solidity
Contracts	0x0091371eaef2da68c8dd77975358c91e8570640f
Website	https://deflationarytoken.com/

Public logo:

Solidity Source Code On Blockchain (Verified Contract Source Code)

https://bscscan.com/token/0x0091371eaef2da68c8dd77975358c91e8570640f#code

Contract Name: Deflationary Token

Compiler Version: v0.8.17

Optimization Enabled: Yes with 200 runs

SHA-1 Hash

Solidity source code is audited at hash

#xk282js9ksxiam3ca8c76a9bd11283d33371h786

Audit Scope & Methodology

The scope of this report is to audit the smart contract source code of Deflationary Token , Bit Control has scanned the contract and reviewed the project for common vulnerabilities, exploits, hacks, and back-doors. Below is the list of commonly known smart contract vulnerabilities, exploits, and hacks:

Smart Contract Vulnerabilities

- □ Re-entrancy
- □ Unhandled Exceptions
- □ Transaction Order Dependency
- □ Integer Overflow
- □ Unrestricted Action
- □ Incorrect Inheritance Order
- □ Typographical Errors
- □ Requirement Violation

Source Code Review

- □ Ownership Takeover
- □ Gas Limit and Loops
- □ Deployment Consistency
- □ Repository Consistency
- □ Data Consistency
- □ Token Supply Manipulation

Functional Assessment

- □ Access Control and Authorization
- □ Operations Trail and Event Generation
- □ Assets Manipulation
- □ Liquidity Access

Bit Control Audit Standard

The aim of Bit Control standard is to analyze the smart contract and identify the vulnerabilities and the hacks in the smart contract. Mentioned are the steps used by ECHELON-1 to assess the smart contract:

1. Solidity smart contract source code reviewal:

- Review of the specifications, sources, and instructions provided to Bit Control to make sure we understand the size, scope, and functionality of the smart contract.
- Manual review of code, which is the process of reading source code line-by-line to identify potential vulnerabilities.

2. Static, Manual, and Software analysis:

- Test coverage analysis, which is the process of determining whether the test cases are covering the code and how much code is exercised when we run those test cases.
- Symbolic execution, which is analyzing a program to determine what inputs cause each part of a program to execute.

3. Best practices review, which is a review of the smart contracts to improve efficiency, effectiveness, clarify, maintainability, security, and control based on the established industry and academic practices, recommendations, and research.

4. Specific, itemized, actionable recommendations to help you take steps to secure your smart contracts

Automated 3P frameworks used to assess the smart contract vulnerabilities

- Slither
- Consensys MythX
- Consensus Surya
- Open Zeppelin Code Analyzer
- Solidity Code Complier

Bit Control's Risk Classification

Smart contracts are generally designed to manipulate and hold funds denominated in ETH/BNB. This makes them very tempting attack targets, as a successful attack may allow the attacker to directly steal funds from the contract. Below are the typical risk levels of a smart contract: Vulnerable:

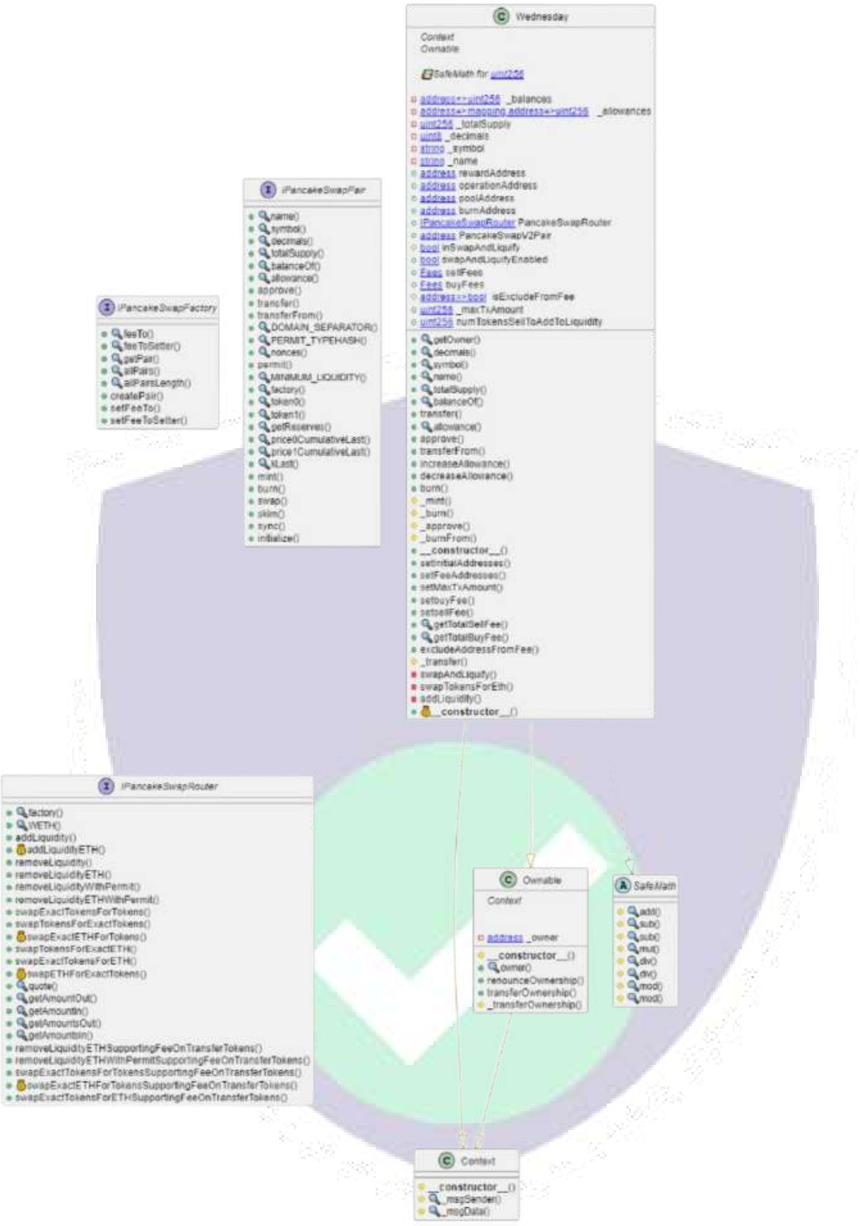
A contract is vulnerable if it has been flagged by a static analysis tool as such. As we will see later, this means that some contracts may be vulnerable because of a false-positive.

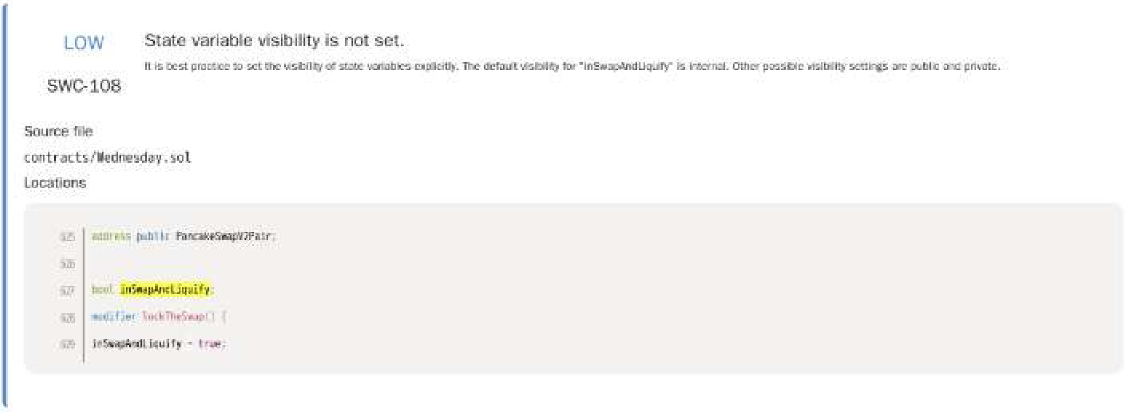
Exploitable: A contract is exploitable if it is vulnerable and the vulnerability could be exploited by an external attacker. For example, if the "vulnerability" flagged by a tool is in a function which requires to own the contract, it would be vulnerable but not exploitable.

Exploited: A contract is exploited if it received a transaction on the main network which triggered one of its vulnerabilities. Therefore, a contract can be vulnerable or even exploitable without having been exploited.

Risk severity	Meaning
! Critical	This level of vulnerability could be exploited easily, and can lead to asset loss, data loss, asset manipulation, or data manipulation. They should be fixed right away.
! High	This level vulnerabilities are hard to exploit but very important to fix, they carry an elevated risk of smart contract manipulation, which can lead to critical risk severity
! Medium	This level of vulnerabilities should be fixed, as they carry an inherent risk of future exploits, and hacks which may or may not impact the smart contract execution.
! Low	This level of vulnerability can be ignored. They are code style violations, and informational statements in the code. They may not affect the smart contract execution

Smart Contract Risk Assessment





LOW

SWC-108

State variable visibility is not set.

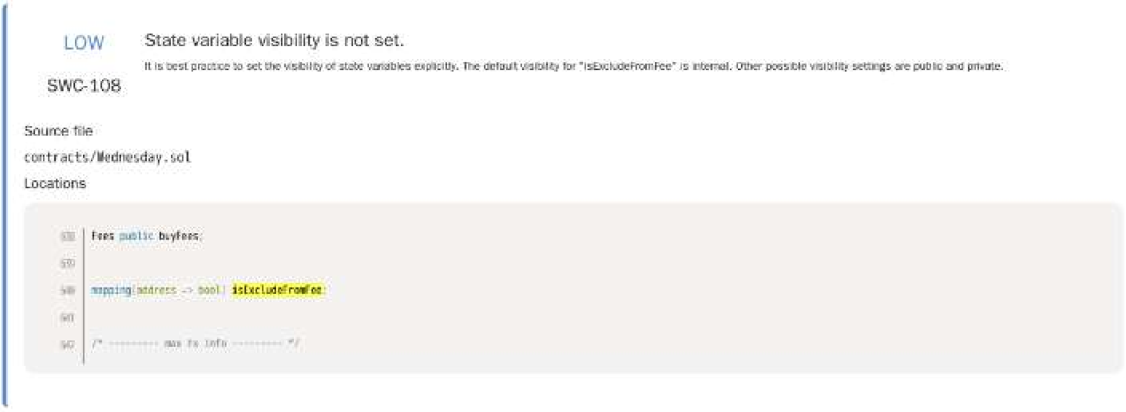
It is best practice to set the visibility of state variables explicitly. The default visibility for "inSwapAndLiquify" is internal. Other possible visibility settings are public and private.

Source file

contracts/Wednesday.sol

Locations

```
address public PancakeSwapV2Pair;

bool inSwapAndLiquify;
modifier lockTheSwap() {
inSwapAndLiquify = true;
```



LOW

SWC-108

State variable visibility is not set.

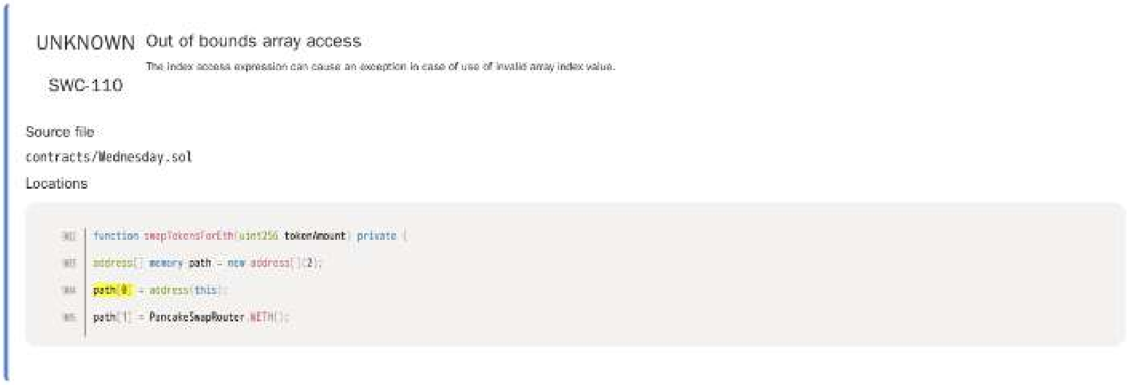
It is best practice to set the visibility of state variables explicitly. The default visibility for "isExcludeFromFee" is internal. Other possible visibility settings are public and private.

Source file

contracts/Wednesday.sol

Locations

```
Fees public buyFees;

mapping(address => bool) isExcludeFromFee;

/* ---------- max tx info ---------- */
```



UNKNOWN

SWC-110

Out of bounds array access

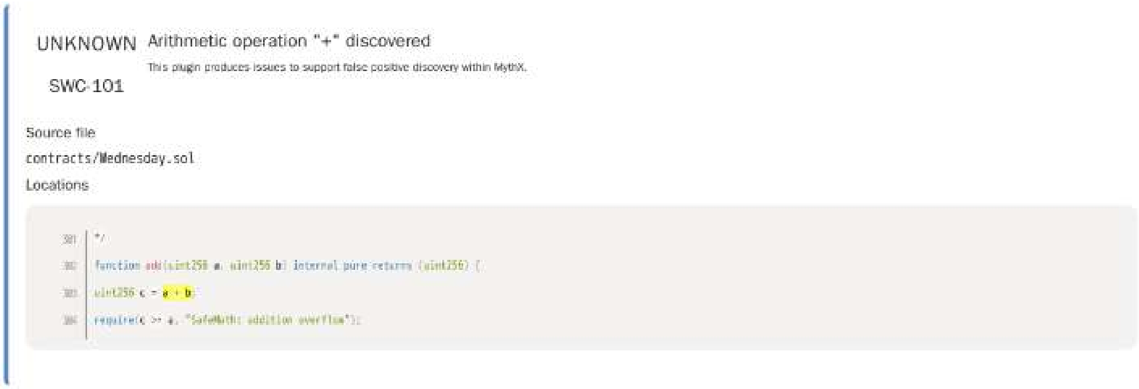
The index access expression can cause an exception in case of use of invalid array index value.

Source file

contracts/Wednesday.sol

Locations

```
function swapTokensForEth(uint256 tokenAmount) private {
address[] memory path = new address[](2);
path[0] = address(this);
path[1] = PancakeSwapRouter.WETH();
```



UNKNOWN Arithmetic operation "+" discovered

SWC-101

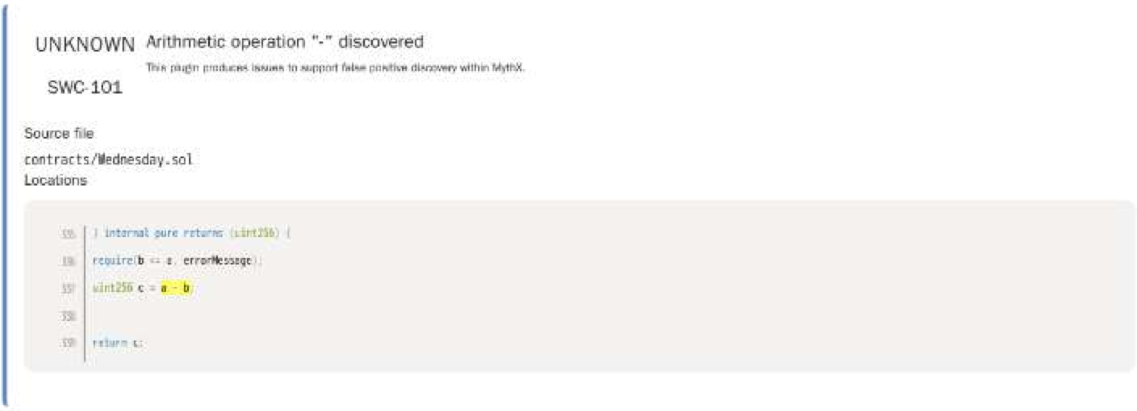
This plugin produces issues to support false positive discovery within MythX.

Source file

contracts/Wednesday.sol

Locations

```
*/
function add(uint256 a, uint256 b) internal pure returns (uint256) {
uint256 c = a + b;
require(c >= a, "SafeMath: addition overflow");
```



UNKNOWN Arithmetic operation "-" discovered

SWC-101

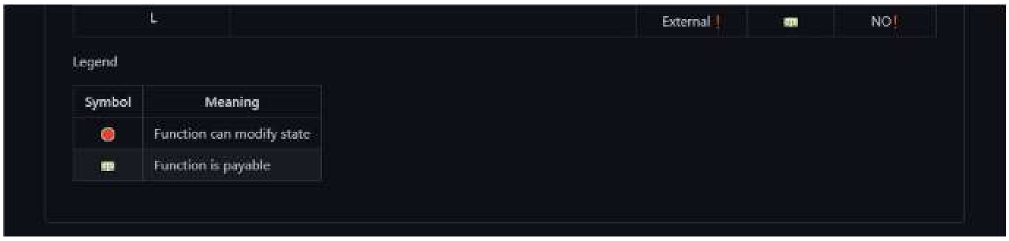
This plugin produces issues to support false positive discovery within MythX.

Source file

contracts/Wednesday.sol

Locations

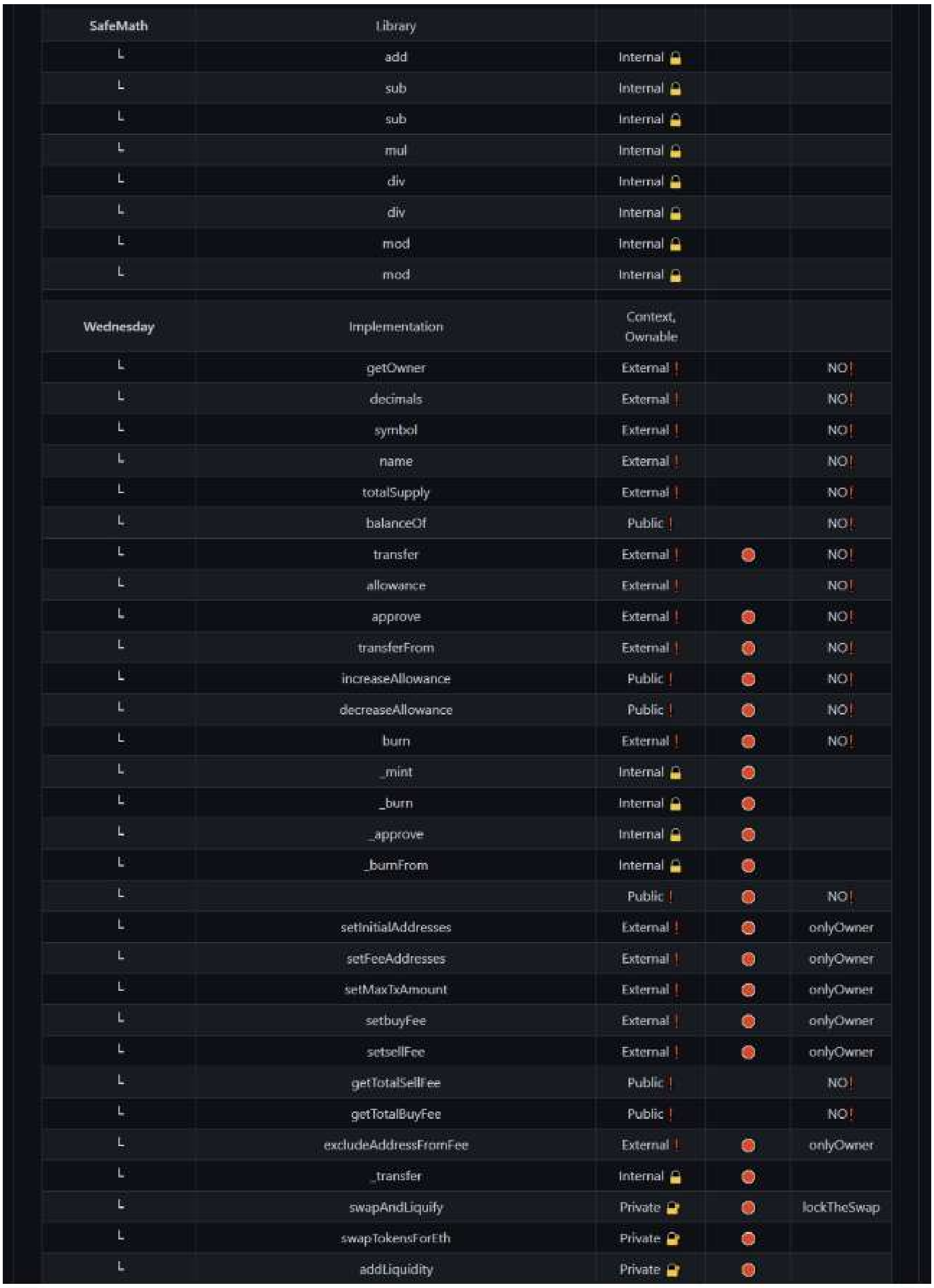
```
) internal pure returns (uint256) {
require(b <= a, errorMessage);
uint256 c = a - b;

return c;
```



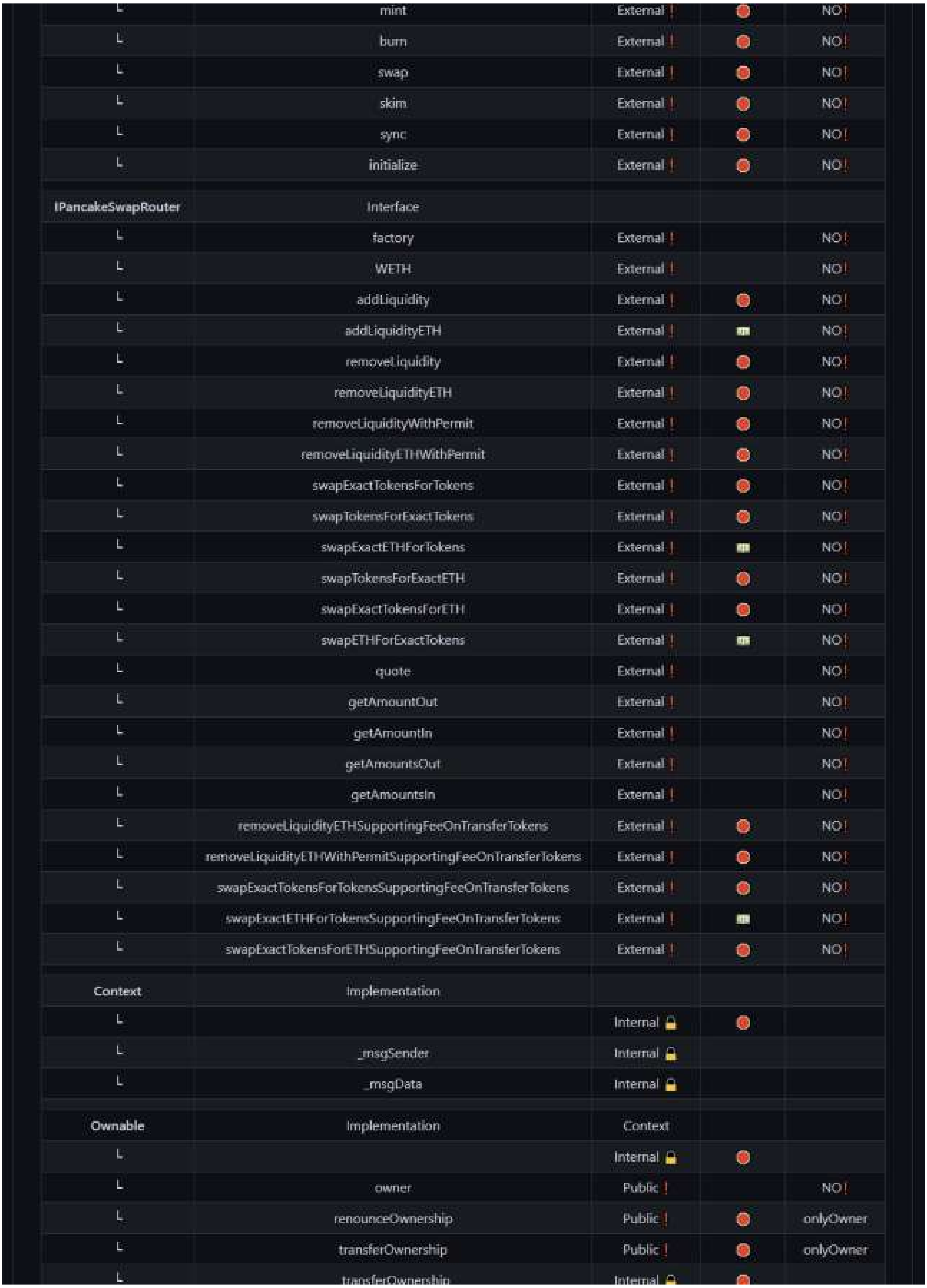
└		External !	💵	NO!

Legend

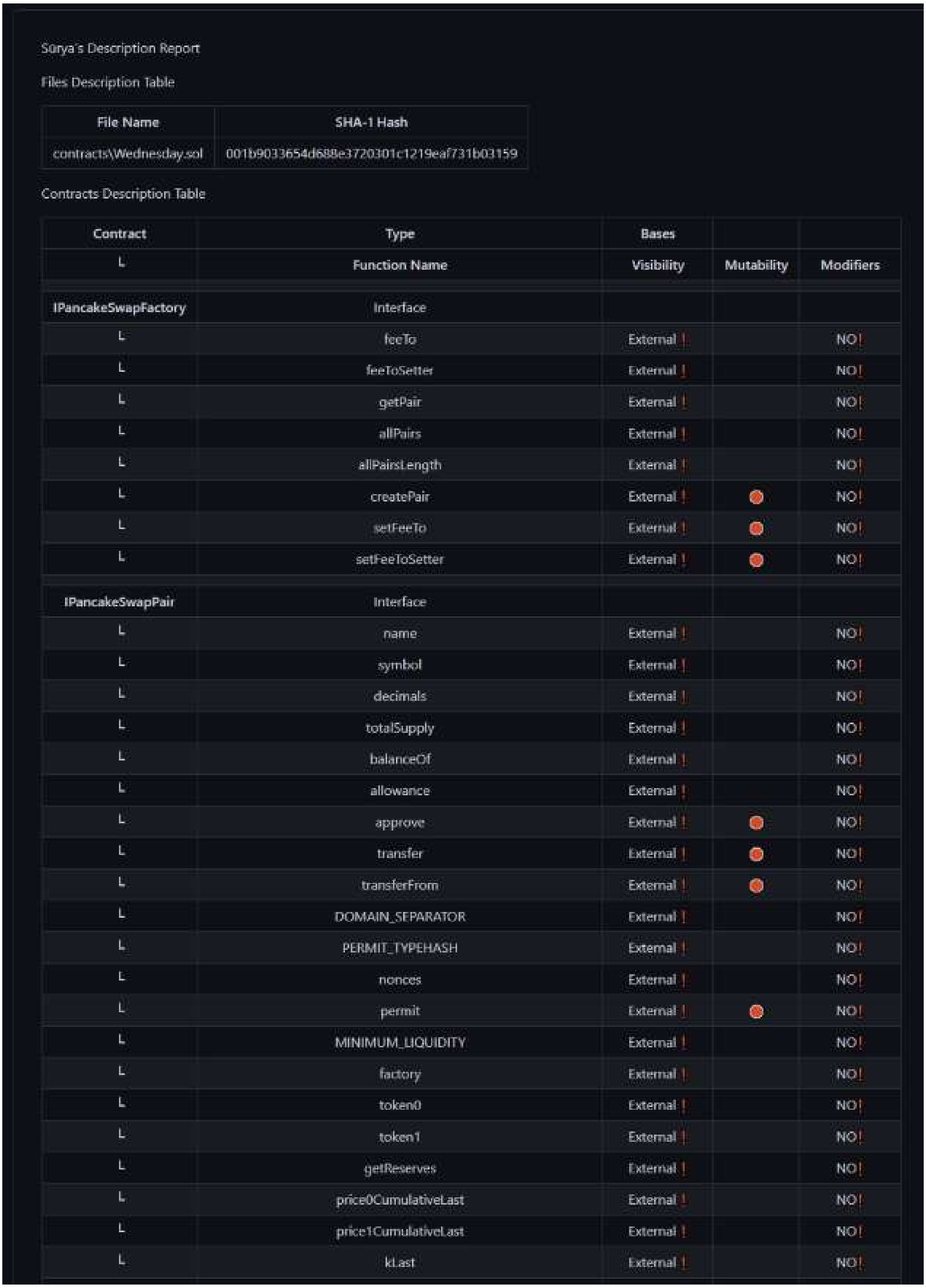
Symbol	Meaning
🔴	Function can modify state
💵	Function is payable



SafeMath	Library			
└	add	Internal 🔒		
└	sub	Internal 🔒		
└	sub	Internal 🔒		
└	mul	Internal 🔒		
└	div	Internal 🔒		
└	div	Internal 🔒		
└	mod	Internal 🔒		
└	mod	Internal 🔒		
Wednesday	Implementation	Context, Ownable		
└	getOwner	External !		NO!
└	decimals	External !		NO!
└	symbol	External !		NO!
└	name	External !		NO!
└	totalSupply	External !		NO!
└	balanceOf	Public !		NO!
└	transfer	External !	🔴	NO!
└	allowance	External !		NO!
└	approve	External !	🔴	NO!
└	transferFrom	External !	🔴	NO!
└	increaseAllowance	Public !	🔴	NO!
└	decreaseAllowance	Public !	🔴	NO!
└	burn	External !	🔴	NO!
└	_mint	Internal 🔒	🔴	
└	_burn	Internal 🔒	🔴	
└	_approve	Internal 🔒	🔴	
└	_burnFrom	Internal 🔒	🔴	
└		Public !	🔴	NO!
└	setInitialAddresses	External !	🔴	onlyOwner
└	setFeeAddresses	External !	🔴	onlyOwner
└	setMaxTxAmount	External !	🔴	onlyOwner
└	setbuyFee	External !	🔴	onlyOwner
└	setsellFee	External !	🔴	onlyOwner
└	getTotalSellFee	Public !		NO!
└	getTotalBuyFee	Public !		NO!
└	excludeAddressFromFee	External !	🔴	onlyOwner
└	_transfer	Internal 🔒	🔴	
└	swapAndLiquify	Private 🔐	🔴	lockTheSwap
└	swapTokensForEth	Private 🔐	🔴	
└	addLiquidity	Private 🔐	🔴	



└	mint	External ❗	🛑	NO❗
└	burn	External ❗	🛑	NO❗
└	swap	External ❗	🛑	NO❗
└	skim	External ❗	🛑	NO❗
└	sync	External ❗	🛑	NO❗
└	initialize	External ❗	🛑	NO❗
IPancakeSwapRouter	Interface			
└	factory	External ❗		NO❗
└	WETH	External ❗		NO❗
└	addLiquidity	External ❗	🛑	NO❗
└	addLiquidityETH	External ❗	💵	NO❗
└	removeLiquidity	External ❗	🛑	NO❗
└	removeLiquidityETH	External ❗	🛑	NO❗
└	removeLiquidityWithPermit	External ❗	🛑	NO❗
└	removeLiquidityETHWithPermit	External ❗	🛑	NO❗
└	swapExactTokensForTokens	External ❗	🛑	NO❗
└	swapTokensForExactTokens	External ❗	🛑	NO❗
└	swapExactETHForTokens	External ❗	💵	NO❗
└	swapTokensForExactETH	External ❗	🛑	NO❗
└	swapExactTokensForETH	External ❗	🛑	NO❗
└	swapETHForExactTokens	External ❗	💵	NO❗
└	quote	External ❗		NO❗
└	getAmountOut	External ❗		NO❗
└	getAmountIn	External ❗		NO❗
└	getAmountsOut	External ❗		NO❗
└	getAmountsIn	External ❗		NO❗
└	removeLiquidityETHSupportingFeeOnTransferTokens	External ❗	🛑	NO❗
└	removeLiquidityETHWithPermitSupportingFeeOnTransferTokens	External ❗	🛑	NO❗
└	swapExactTokensForTokensSupportingFeeOnTransferTokens	External ❗	🛑	NO❗
└	swapExactETHForTokensSupportingFeeOnTransferTokens	External ❗	💵	NO❗
└	swapExactTokensForETHSupportingFeeOnTransferTokens	External ❗	🛑	NO❗
Context	Implementation			
└		Internal 🔒	🛑	
└	_msgSender	Internal 🔒		
└	_msgData	Internal 🔒		
Ownable	Implementation	Context		
└		Internal 🔒	🛑	
└	owner	Public ❗		NO❗
└	renounceOwnership	Public ❗	🛑	onlyOwner
└	transferOwnership	Public ❗	🛑	onlyOwner
└	_transferOwnership	Internal 🔒	🛑	



Sūrya's Description Report

Files Description Table

File Name	SHA-1 Hash
contracts\Wednesday.sol	001b9033654d688e3720301c1219eaf731b03159

Contracts Description Table

Contract	Type	Bases		
└	**Function Name**	**Visibility**	**Mutability**	**Modifiers**
IPancakeSwapFactory	Interface			
└	feeTo	External ❗		NO❗
└	feeToSetter	External ❗		NO❗
└	getPair	External ❗		NO❗
└	allPairs	External ❗		NO❗
└	allPairsLength	External ❗		NO❗
└	createPair	External ❗	🔴	NO❗
└	setFeeTo	External ❗	🔴	NO❗
└	setFeeToSetter	External ❗	🔴	NO❗
IPancakeSwapPair	Interface			
└	name	External ❗		NO❗
└	symbol	External ❗		NO❗
└	decimals	External ❗		NO❗
└	totalSupply	External ❗		NO❗
└	balanceOf	External ❗		NO❗
└	allowance	External ❗		NO❗
└	approve	External ❗	🔴	NO❗
└	transfer	External ❗	🔴	NO❗
└	transferFrom	External ❗	🔴	NO❗
└	DOMAIN_SEPARATOR	External ❗		NO❗
└	PERMIT_TYPEHASH	External ❗		NO❗
└	nonces	External ❗		NO❗
└	permit	External ❗	🔴	NO❗
└	MINIMUM_LIQUIDITY	External ❗		NO❗
└	factory	External ❗		NO❗
└	token0	External ❗		NO❗
└	token1	External ❗		NO❗
└	getReserves	External ❗		NO❗
└	price0CumulativeLast	External ❗		NO❗
└	price1CumulativeLast	External ❗		NO❗
└	kLast	External ❗		NO❗

Risk Status

Risk severity	Meaning
! Critical	None critical severity issues identified
! High	None high severity issues identified
! Medium	None medium severity issues identified
! Low	None low severity issues identified
Verified	7 functions and instances verified and checked
Safety Score	95 out of 100

Report Summary

Bit Control team has performed a line-by-line manual analysis and automated review of the smart contract. The smart contract was analyzed mainly for common smart contract vulnerabilities, exploits, and manipulation hacks.

Deflationary Token smart contract source code has **LOW RISK SEVERITY.**
Deflationary Token has **PASSED** the smart contract audit.

Note for stakeholders:

Be aware that active smart contract owner privileges constitute an elevated impact on smart contract’s safety and security.
Make sure that the project team’s KYC/identity is verified by an independent firm, e.g., Bit Control.
Always check if the contract’s liquidity is locked. A longer liquidity lock plays an important role in the project's longevity. It is recommended to have multiple liquidity providers.
Examine the unlocked token supply in the owner, developer, or team’s private wallets. Understand the project’s tokenomics, and make sure the tokens outside of the LP Pair are vested or locked for a longer period of time.
Ensure that the project’s official website is hosted on a trusted platform, and is using an active SSL certificate. The website’s domain should be registered for a longer period of time.

Legal Advisory

Important Disclaimer

Bit Control provides contract auditing and project verification services for blockchain projects. The purpose of the audit is to analyze the on-chain smart contract source code, and to provide a basic overview of the project. This report should not be transmitted, disclosed, referred to, or relied upon by any person for any purposes without Bit Control prior written consent.

Bit Control provides the easy-to-understand assessment of the project, and the smart contract (otherwise known as the source code). The audit makes no statements or warranties on the security of the code. It also cannot be considered as an adequate assessment regarding the utility and safety of the code, bug-free status, or any other statements of the contract. While we have used all the data at our disposal to provide the transparent analysis, it is important to note that you should not rely on this report only — we recommend proceeding with several independent audits and a public bug bounty program to ensure the security of smart contracts. Be aware that smart contracts deployed on a blockchain aren't resistant from external vulnerability, or a hack. Be aware that active smart contract owner privileges constitute an elevated impact to smart contract's safety and security. Therefore, Bit Control does not guarantee the explicit security of the audited smart contract.

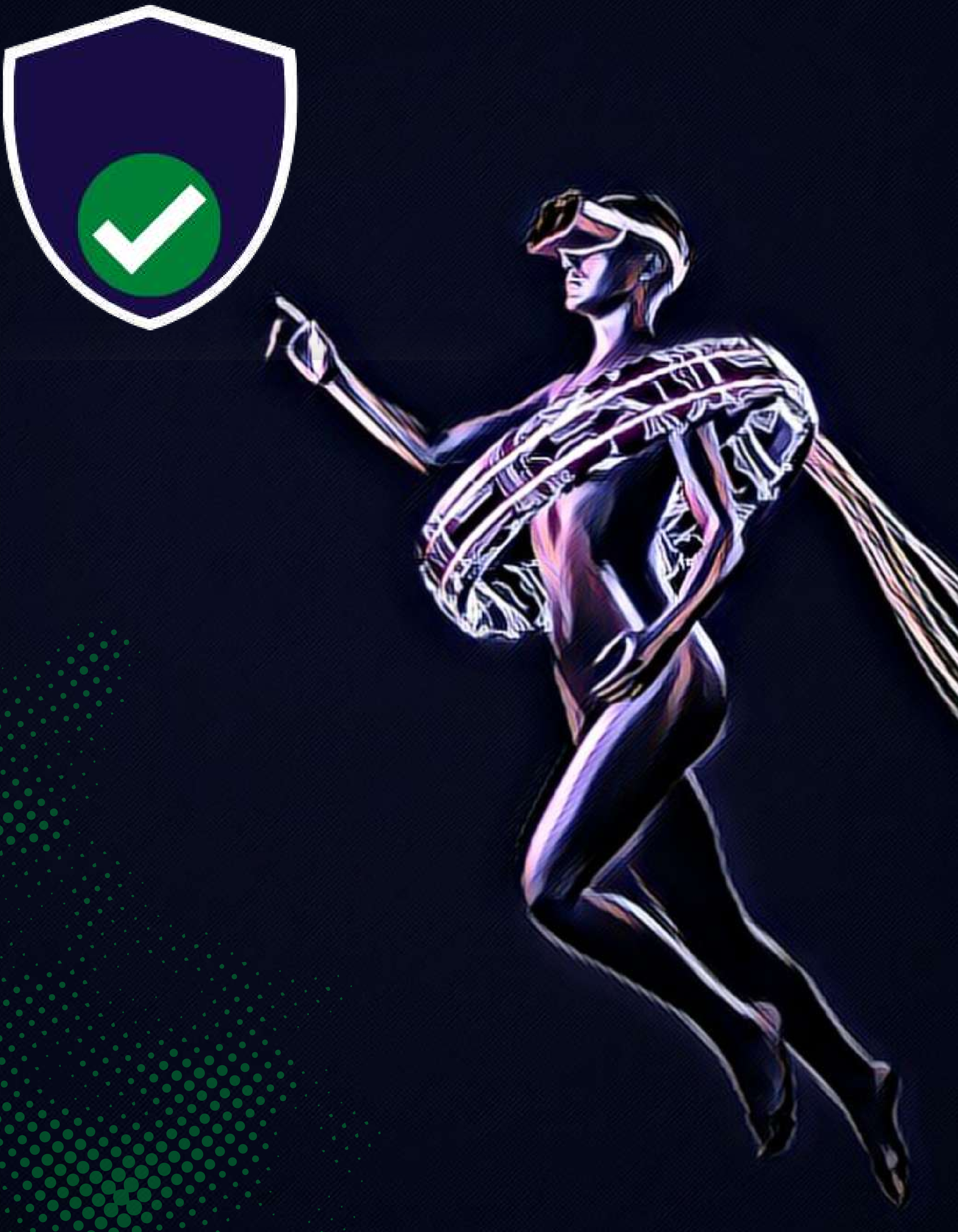
The analysis of the security is purely based on the smart contracts alone. No applications or operations were reviewed for security. No product code has been reviewed.

This report should not be considered as an endorsement or disapproval of any project or team. The information provided in this report does not constitute investment advice, financial advice, trading advice, or any other sort of advice and you should not treat any of the report's content as such. Do conduct your own due diligence and consult your financial advisor before making any investment decisions.

About Bit Control

Bit Control provides intelligent blockchain solutions. Bit Control is developing an ecosystem that is seamless and responsive. Some of our services: Blockchain Security, Token Launchpad, NFT Marketplace, etc. Bit Control's mission is to interconnect multiple services like Blockchain Security, DeFi, Gaming, and Marketplace under one ecosystem that is seamless, multi-chain compatible, scalable, secure, fast, responsive, and easy-to-use.

Bit Control is built by a decentralized team of UI experts, contributors, engineers, and enthusiasts from all over the world. Our team currently consists of 3+ core team members, and 6+ casual contributors. Bit Control provides manual, static, and automatic smart contract analysis, to ensure that the project is checked against known attacks and potential vulnerabilities.



@BIT_CONTROL