SOLIDProof

Bring trust into your projects

Blockchain Security | Smart Contract Audits | KYC

MADE IN GERMANY

Audit

Security Assessment

28. June, 2022

For

PUC

Disclaimer

SolidProof.io reports are not, nor should be considered, an “endorsement” or “disapproval” of any particular project or team. These reports are not, nor should be considered, an indication of the economics or value of any “product” or “asset” created by any team. SolidProof.io do not cover testing or auditing the integration with external contract or services (such as Unicrypt, Uniswap, PancakeSwap etc’...)

SolidProof.io Audits do not provide any warranty or guarantee regarding the absolute bug- free nature of the technology analyzed, nor do they provide any indication of the technology proprietors. SolidProof Audits should not be used in any way to make decisions around investment or involvement with any particular project. These reports in no way provide investment advice, nor should be leveraged as investment advice of any sort.

SolidProof.io Reports represent an extensive auditing process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology. Blockchain technology and cryptographic assets present a high level of ongoing risk. SolidProof’s position is that each company and individual are responsible for their own due diligence and continuous security. SolidProof in no way claims any guarantee of security or functionality of the technology we agree to analyze.

Version	Date	Description
1.0	03. January 2022 // 26. June 2022	• Layout project • Automated- /Manual-Security Testing • Summary

Network
Binance Smart Chain (BEP20)

Website
https://trypledge.org/

Telegram
https://t.me/ThePledgeCoin

Twitter
https://www.twitter.com/thepledgecoin

Facebook
https://www.facebook.com/thepledgecoin

Instagram
https://www.instagram.com/thepledgecoin

Description

Our mission is to be a force for the greater good. Specifically we intend to do this by powering charitable giving, and charitable initiatives through block chain solutions.

Project Engagement

During the 30th of December 2021, **Pledge Utility Coin Team** engaged Solidproof.io to audit smart contracts that they created. The engagement was technical in nature and focused on identifying security flaws in the design and implementation of the contracts. They provided Solidproof.io with access to their code repository and whitepaper.

Logo

Contract Link

v1.0

- Github
  - https://github.com/Pledge-Utility-Coin-Token/solidity-contracts/tree/dev/contracts
  - Commit: 129ff75a11edc9f3235f478493e7197a63ca4c9d

- Binance Smart Chain

  https://bscscan.com/address/0x9f7f13ff14ff45addc1df1ae6c91f6ee553d1200#code

  https://bscscan.com/address/0x9c6d9db24ed3b8350edb685377fdc0c4b6617715#code

Over Proxy contract

https://bscscan.com/address/0xc9dc172bB1f1afe84d303c2F84b7B70A134a5d38

Vulnerability & Risk Level

Risk represents the probability that a certain source-threat will exploit vulnerability, and the impact of that event on the organization or system. Risk Level is computed based on CVSS version 3.0.

Level	Value	Vulnerability	Risk (Required Action)
Critical	9 - 10	A vulnerability that can disrupt the contract functioning in a number of scenarios, or creates a risk that the contract may be broken.	Immediate action to reduce risk level.
High	7 – 8.9	A vulnerability that affects the desired outcome when using a contract, or provides the opportunity to use a contract in an unintended way.	Implementation of corrective actions as soon aspossible.
Medium	4 – 6.9	A vulnerability that could affect the desired outcome of executing the contract in a specific scenario.	Implementation of corrective actions in a certain period.
Low	2 – 3.9	A vulnerability that does not have a significant impact on possible scenarios for the use of the contract and is probably subjective.	Implementation of certain corrective actions or accepting the risk.
Informational	0 – 1.9	A vulnerability that have informational character but is not effecting any of the code.	An observation that does not determine a level of risk

Auditing Strategy and Techniques Applied

Throughout the review process, care was taken to evaluate the repository for security-related issues, code quality, and adherence to specification and best practices. To do so, reviewed line-by-line by our team of expert pentesters and smart contract developers, documenting any issues as there were discovered.

Methodology

The auditing process follows a routine series of steps:

1. Code review that includes the following:
    i) Review of the specifications, sources, and instructions provided to SolidProof to make sure we understand the size, scope, and functionality of the smart contract.
    ii) Manual review of code, which is the process of reading source code line-by-line in an attempt to identify potential vulnerabilities.
    iii) Comparison to specification, which is the process of checking whether the code does what the specifications, sources, and instructions provided to SolidProof describe.

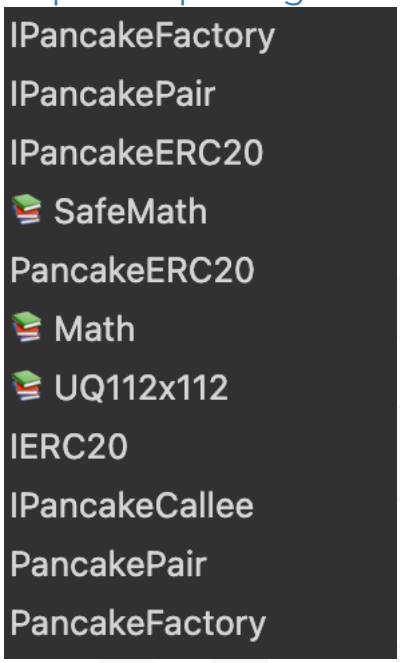
2. Testing and automated analysis that includes the following:
    i) Test coverage analysis, which is the process of determining whether the test cases are actually covering the code and how much code is exercised when we run those test cases.
    ii) Symbolic execution, which is analysing a program to determine what inputs causes 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.

Used Code from other Frameworks/Smart Contracts (direct imports)

Imported packages:



Tested Contract Files

This audit covered the following files listed below with a SHA-1 Hash.

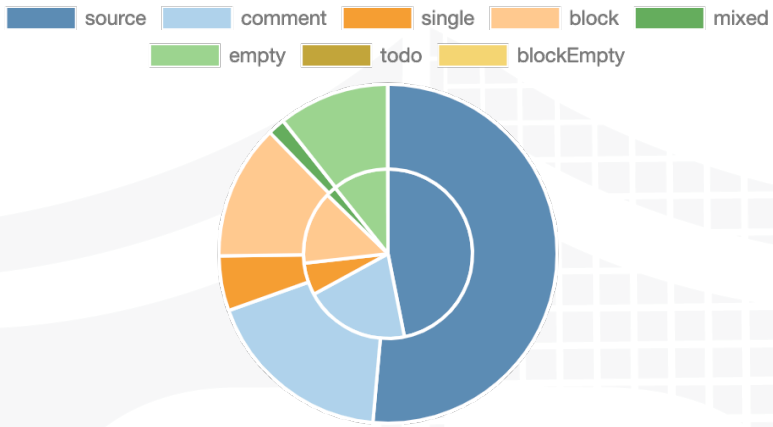
A file with a different Hash has been modified, intentionally or otherwise, after the security review. A different Hash could be (but not necessarily) an indication of a changed condition or potential vulnerability that was not within the scope of this review.

v1.0

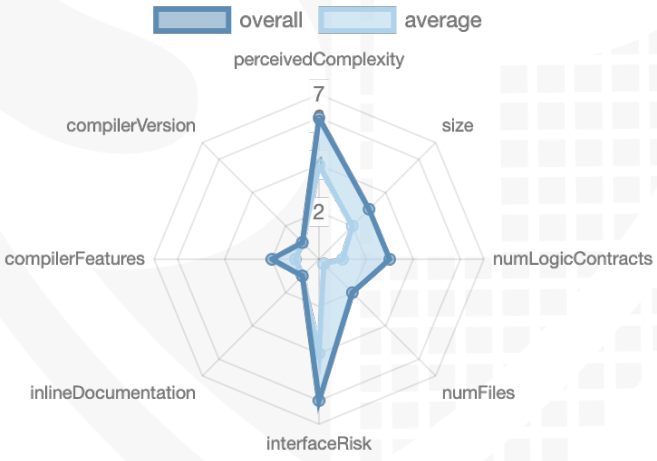
File Name	SHA-1 Hash
contracts/Pledge01Up.sol	2384a496b9271bb10b097fe06584fc23b84d26b3
contracts/weth.sol	a591cb13cf771aa6f0a05c5db82b11673a859dfb
contracts/PancakeRouter.sol	1306ed3fa9fe406a731f5115c214bb2d3221768b
contracts/Open-Zeppelin.sol	c31be077a9f134edaaade67b48ea63618e912b40
contracts/Pledge02Up.sol	11d3679ec6f18a0b28c0ebb3b2a98a792963562d
contracts/Safemath.sol	d7243fa694de96304176b5afee0e5881f32cc146
contracts/PancakeFactory.sol	97281475aec2393ae7f7cce2970423bf086b9566

Metrics

Source Lines
v1.0



Risk Level
v1.0



Capabilities

Components

Version	Contracts	Libraries	Interfaces	Abstract
1.0	11	10	18	4

Exposed Functions

This section lists functions that are explicitly declared public or payable. Please note that getter methods for public stateVars are not included.

Version	Public	Payable
1.0	314	13

Version	External	Internal	Private	Pure	View
1.0	255	303	11	65	118

State Variables

Version	Total	Public
1.0	108	79

Capabilities

Version	Solidity Versions observed	Experimental Features	Can Receive Funds	Uses Assembly	Has Destroyable Contracts
1.0	`0.6.12` `^0.6.12` `=0.6.12` `=0.5.16`	`ABIEncoderV2`	`yes`	`yes` (4 asm blocks)	

Version	Transfers ETH	Low-Level Calls	DelegateCall	Uses Hash Functions	ECRecover	New/Create/Create2
1.0	`yes`			`yes`	`yes`	`yes → AssemblyCall: Name: create2`

Scope of Work

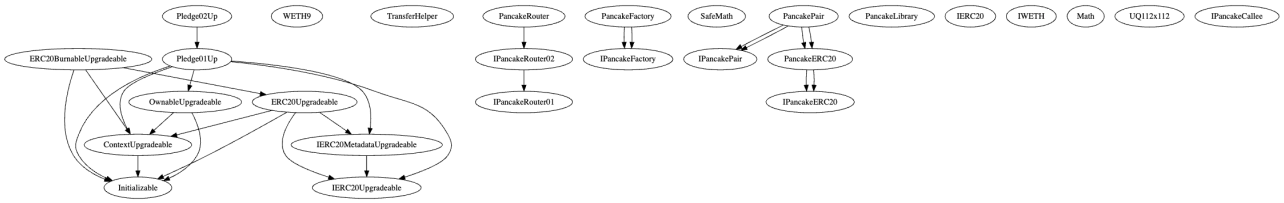
The above token Team provided us with the files that needs to be tested (Github, Bscscan, Etherscan, files, etc.). The scope of the audit is the main contract (usual the same name as team appended with .sol).

We will verify the following claims:

1. Correct implementation of Token standard
2. Deployer cannot mint any new tokens
3. Deployer cannot burn or lock user funds
4. Deployer cannot pause the contract
5. Overall checkup (Smart Contract Security)

Inheritance Graph

v1.0



Verify Claims

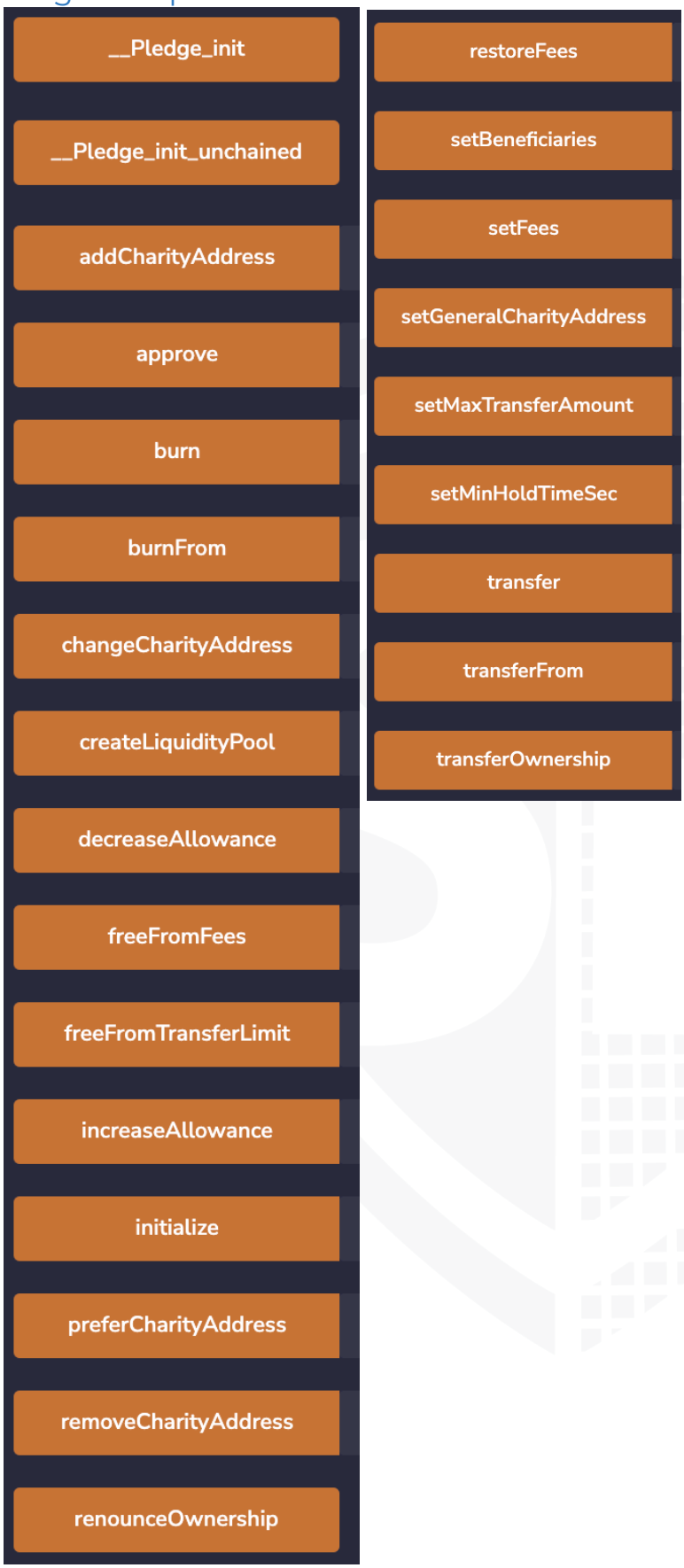
Correct implementation of Token standard

Tested	Verified
✓	✓

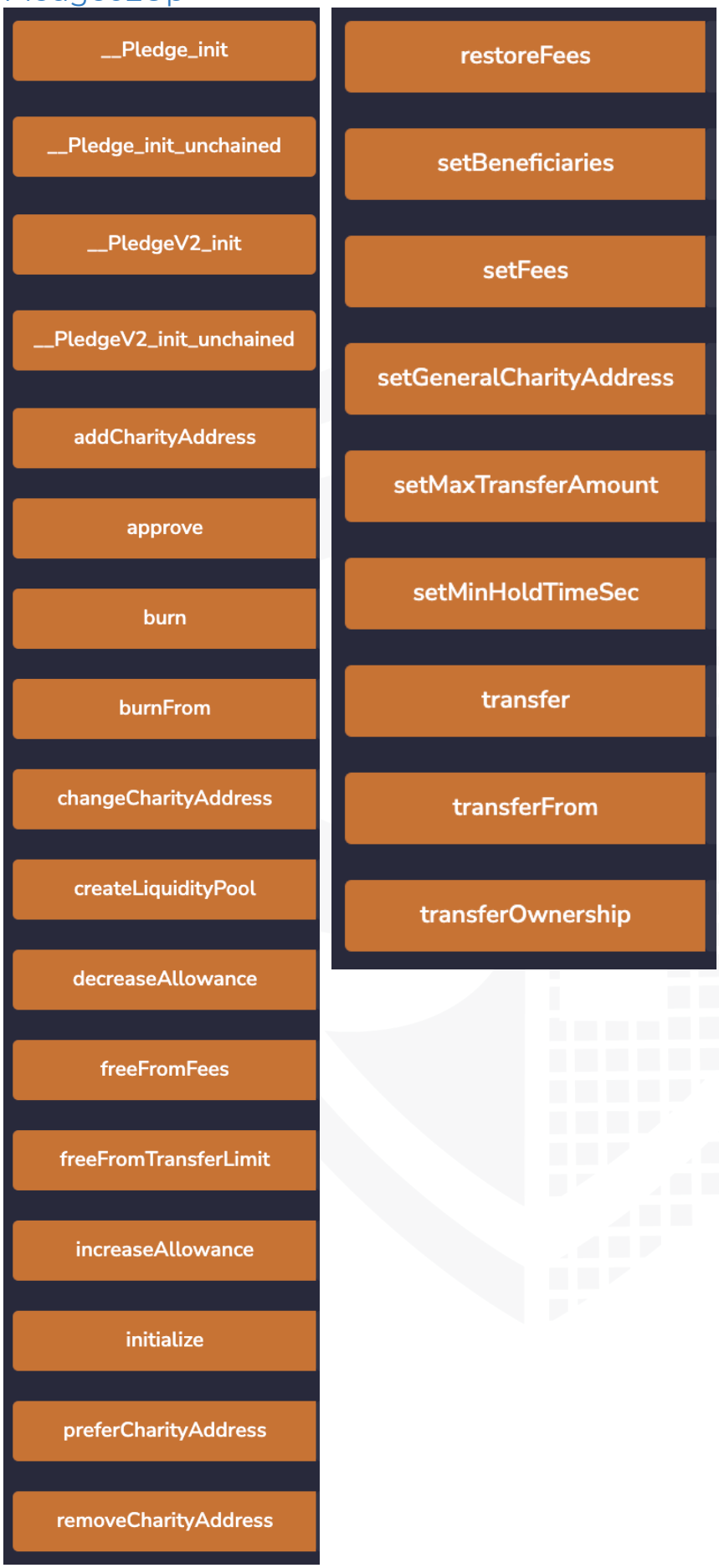
Function	Description	Exist	Tested	Verified
TotalSupply	provides information about the total token supply	✓	✓	✓
BalanceOf	provides account balance of the owner's account	✓	✓	✓
Transfer	executes transfers of a specified number of tokens to a specified address	✓	✓	✓
TransferFrom	executes transfers of a specified number of tokens from a specified address	✓	✓	✓
Approve	allow a spender to withdraw a set number of tokens from a specified account	✓	✓	✓
Allowance	returns a set number of tokens from a spender to the owner	✓	✓	✓

Write functions of contract

Plegde01Up



Pledge02Up


__Pledge_init
__Pledge_init_unchained
__PledgeV2_init
__PledgeV2_init_unchained
addCharityAddress
approve
burn
burnFrom
changeCharityAddress
createLiquidityPool
decreaseAllowance
freeFromFees
freeFromTransferLimit
increaseAllowance
initialize
preferCharityAddress
removeCharityAddress
restoreFees
setBeneficiaries
setFees
setGeneralCharityAddress
setMaxTransferAmount
setMinHoldTimeSec
transfer
transferFrom
transferOwnership

Deployer cannot mint any new tokens

Name	Exist	Tested	Verified
Deployer cannot mint	✓	✓	✓

Max / Total Supply: 1.000.000.000.000.000

Deployer cannot burn or lock user funds

Name	Exist	Tested	Verified
Deployer cannot lock	✓	✓	✗
Deployer cannot burn	✓	✓	✓

Comments:

v1.0

- Everybody can burn tokens
- Deployer can lock user funds
    - By setting maxTransferAmount to 0

```
732     function _settleCharityRelation(address sender⬆, address recipient⬆) internal virtual {
733         // transferring directly to a charity that is not yes preferred.
734         if(isCharityAddress[recipient⬆] == true && preferredCharityAddress[sender⬆] == address(0)) {
735             preferredCharityAddress[sender⬆] = recipient⬆;
736             beneficiaries.charity = recipient⬆;
737         } else {
738             // which charity to pay charity tees to?
739             beneficiaries.charity = generalCharityAddress;
740             address preferred = preferredCharityAddress[sender⬆];
741             if( preferred != address(0) && isCharityAddress[preferred] ) { // && preferredCharityAddress[sender] != generalCharityAddress) {
742                 // isCharityAddress[preferred] is required because the owner can freely remove a charity address without knowing if its a holder's preferred charity.
743                 beneficiaries.charity = preferredCharityAddress[sender⬆];
744             }
745             require(beneficiaries.charity != address(0), "Invalid charity");
746         }
747     }
```

Deployer cannot pause the contract

Name	Exist	Tested	Verified
Deployer cannot pause	-	-	-

Overall checkup (Smart Contract Security)

Tested	Verified
✓	✓

Legend

Attribute	Symbol
Verfified / Checked	✓
Partly Verified	⚑
Unverified / Not checked	✗
Not available	–

Modifiers

Pledger01Up

initialize	
Ⓜ initializer	

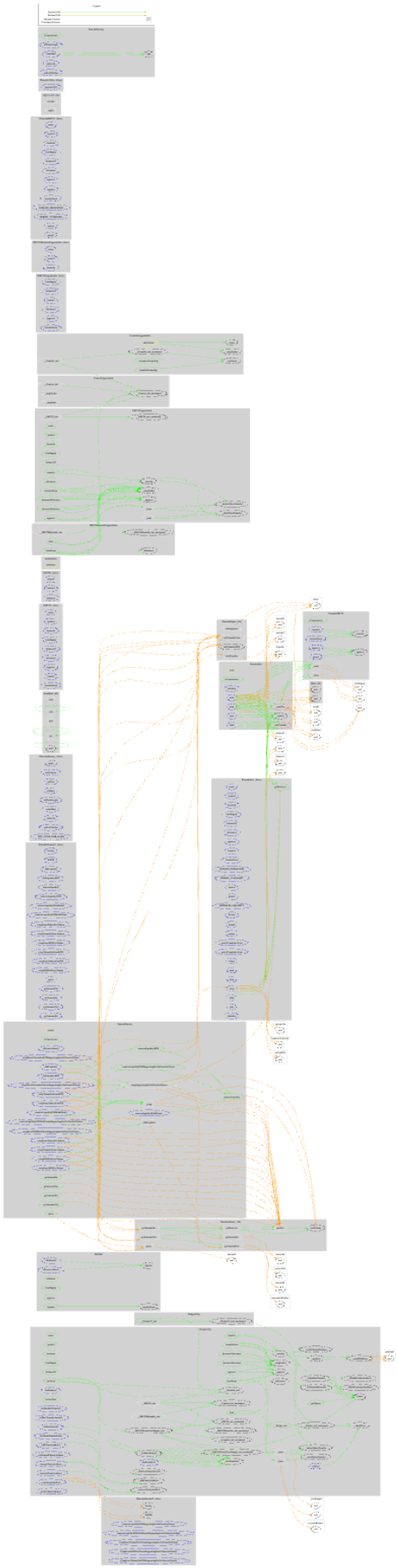
restoreFees	
Ⓜ onlyOwner	
setMinHoldTimeSec	
Ⓜ onlyOwner	
setMaxTransferAmount	
Ⓜ onlyOwner	
setBeneficiaries	
Ⓜ onlyOwner	
createLiquidityPool	
Ⓜ onlyOwner	
setFees	
Ⓜ onlyOwner	
freeFromFees	
Ⓜ onlyOwner	
freeFromTransferLimit	
Ⓜ onlyOwner	
setGeneralCharityAddress	
Ⓜ onlyOwner	
addCharityAddress	
Ⓜ onlyOwner	
removeCharityAddress	
Ⓜ onlyOwner	
changeCharityAddress	
Ⓜ onlyOwner	

Comments

- Deployer can set following state variables without any limitations
    - minHoldTimeSec
    - maxTransferAmount
    - beneficiaries
- Deployer can enable/disable following state variables
    - isFeeFree

- isTransferLimitFree
- isCharityAddress

CallGraph



Source Units in Scope
v1.0

Type	File	Logic Contracts	Interfaces	Lines	nLines	nSLOC	Comment Lines	Complex. Score	Capabilities
📝	contracts/Pledge01Up.sol	1	———	782	757	394	253	298	🧮
📝	contracts/weth.sol	1	———	75	72	50	7	45	💰💠
📝📚🔍	contracts/PancakeRouter.sol	4	6	819	434	360	40	581	💰💠🧱
📝🔍🪂	contracts/Open-Zeppelin.sol	5	2	575	487	194	286	176	🧮🔆
📝	contracts/Pledge02Up.sol	1	———	29	29	14	5	12	🧮
📚	contracts/Safemath.sol	1	———	139	139	40	86	10	🧮🔆
📝📚🔍	contracts/PancakeFactory.sol	12	10	1015	827	646	116	868	💻🧱🚀🌀
📝📚🔍🪂	**Totals**	**25**	**18**	**3434**	**2745**	**1698**	**793**	**1990**	💻🧮💰💠🧱🚀🌀🔆

Legend

Attribute	Description
Lines	total lines of the source unit
nLines	normalized lines of the source unit (e.g. normalizes functions spanning multiple lines)
nSLOC	normalized source lines of code (only source-code lines; no comments, no blank lines)
Comment Lines	lines containing single or block comments
Complexity Score	a custom complexity score derived from code statements that are known to introduce code complexity (branches, loops, calls, external interfaces, ...)

Audit Results

AUDIT PASSED

Critical issues

- no critical issues found -

High issues

- no high issues found -

Medium issues

Issue	File	Type	Line	Description
#1	PancakeRouter	Unchecked tokens transfer	486	Use `SafeERC20`, or ensure that the transfer/ transferFrom return value is checked
#2	Pledge01Up	The arithmetic operation can underflow	178 ,688	It is possible to cause an arithmetic underflow. Please use libraries like SafeMath from openZeppelin to prevent underflows

Low issues

Issue	File	Type	Line	Description
#1	Main	Contract doesn't import npm packages from source (like OpenZeppelin etc.)	-	We recommend to import all packages from npm directly without flatten the contract. Functions could be modified or can be susceptible to vulnerabilities
#2	Main	A floating pragma is set	2	The current pragma Solidity directive is „“^0.6.12””.
#3	Main	Missing Zero Address Validation (missing-zero-check)	396, 620	Check that the address is not zero

Informational issues

Issue	File	Type	Line	Description
#1	Weth	State variables that could be declared constant (constable-states)	6, 7, 8	Add the `constant` attributes to state variables that never change
#2	PancakeRouter	Unused return values	416	Ensure that all the return values of the function calls are used and handle both success and failure cases if needed by the business logic
#3	PancakeRouter	Functions that are not used	406	Remove unused functions
#4	PancakeFactory	SPDX License Identifier is missing	-	Add SPDX License Identifier to the source file

Audit Comments

03. January 2022:

- Deployer can lock user funds
- Read whole report for more information

SWC Attacks

ID	Title	Relationships	Status
SWC-136	Unencrypted Private Data On-Chain	CWE-767: Access to Critical Private Variable via Public Method	PASSED
SWC-135	Code With No Effects	CWE-1164: Irrelevant Code	PASSED
SWC-134	Message call with hardcoded gas amount	CWE-655: Improper Initialization	PASSED
SWC-133	Hash Collisions With Multiple Variable Length Arguments	CWE-294: Authentication Bypass by Capture-replay	PASSED
SWC-132	Unexpected Ether balance	CWE-667: Improper Locking	PASSED
SWC-131	Presence of unused variables	CWE-1164: Irrelevant Code	PASSED
SWC-130	Right-To-Left-Override control character (U+202E)	CWE-451: User Interface (UI) Misrepresentation of Critical Information	PASSED
SWC-129	Typographical Error	CWE-480: Use of Incorrect Operator	PASSED
SWC-128	DoS With Block Gas Limit	CWE-400: Uncontrolled Resource Consumption	PASSED

SWC-127	Arbitrary Jump with Function Type Variable	CWE-695: Use of Low-Level Functionality	PASSED
SWC-125	Incorrect Inheritance Order	CWE-696: Incorrect Behavior Order	PASSED
SWC-124	Write to Arbitrary Storage Location	CWE-123: Write-what-where Condition	PASSED
SWC-123	Requirement Violation	CWE-573: Improper Following of Specification by Caller	PASSED
SWC-122	Lack of Proper Signature Verification	CWE-345: Insufficient Verification of Data Authenticity	PASSED
SWC-121	Missing Protection against Signature Replay Attacks	CWE-347: Improper Verification of Cryptographic Signature	PASSED
SWC-120	Weak Sources of Randomness from Chain Attributes	CWE-330: Use of Insufficiently Random Values	PASSED
SWC-119	Shadowing State Variables	CWE-710: Improper Adherence to Coding Standards	PASSED
SWC-118	Incorrect Constructor Name	CWE-665: Improper Initialization	PASSED
SWC-117	Signature Malleability	CWE-347: Improper Verification of Cryptographic Signature	PASSED

SWC-116	Timestamp Dependence	CWE-829: Inclusion of Functionality from Untrusted Control Sphere	PASSED
SWC-115	Authorization through tx.origin	CWE-477: Use of Obsolete Function	PASSED
SWC-114	Transaction Order Dependence	CWE-362: Concurrent Execution using Shared Resource with Improper Synchronization ('Race Condition')	PASSED
SWC-113	DoS with Failed Call	CWE-703: Improper Check or Handling of Exceptional Conditions	PASSED
SWC-112	Delegatecall to Untrusted Callee	CWE-829: Inclusion of Functionality from Untrusted Control Sphere	PASSED
SWC-111	Use of Deprecated Solidity Functions	CWE-477: Use of Obsolete Function	PASSED
SWC-110	Assert Violation	CWE-670: Always-Incorrect Control Flow Implementation	PASSED
SWC-109	Uninitialized Storage Pointer	CWE-824: Access of Uninitialized Pointer	PASSED
SWC-108	State Variable Default Visibility	CWE-710: Improper Adherence to Coding Standards	PASSED
SWC-107	Reentrancy	CWE-841: Improper Enforcement of Behavioral Workflow	PASSED
SWC-106	Unprotected SELFDESTRUCT Instruction	CWE-284: Improper Access Control	PASSED

SWC-105	Unprotected Ether Withdrawal	CWE-284: Improper Access Control	PASSED
SWC-104	Unchecked Call Return Value	CWE-252: Unchecked Return Value	PASSED
SWC-103	Floating Pragma	CWE-664: Improper Control of a Resource Through its Lifetime	NOT PASSED
SWC-102	Outdated Compiler Version	CWE-937: Using Components with Known Vulnerabilities	PASSED
SWC-101	Integer Overflow and Underflow	CWE-682: Incorrect Calculation	NOT PASSED
SWC-100	Function Default Visibility	CWE-710: Improper Adherence to Coding Standards	PASSED

Blockchain Security | Smart Contract Audits | KYC

MADE IN GERMANY