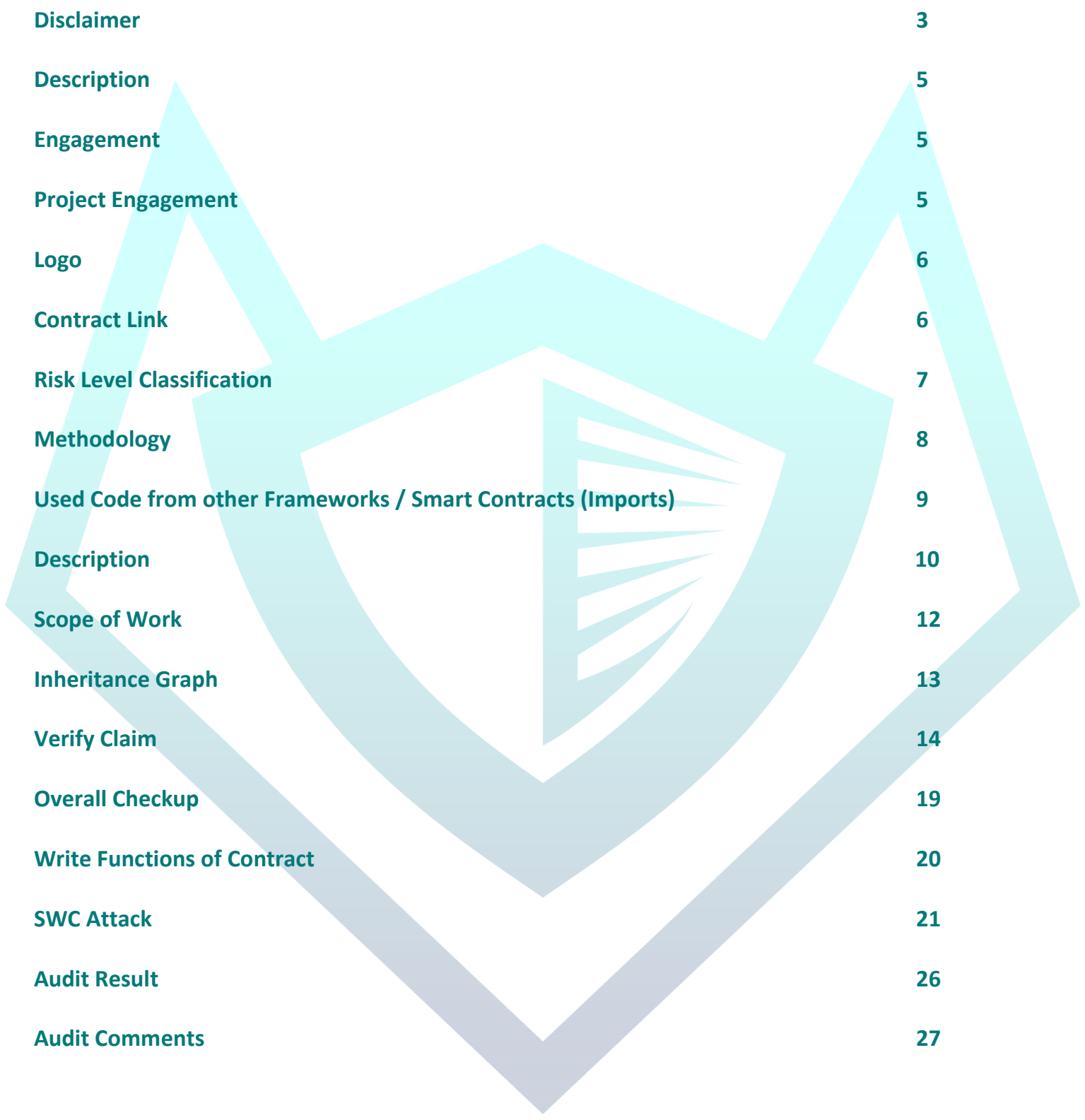# CONTRACT WOLF

Blockchain Security - Smart Contract Audits

Security Assessment

March 08, 2022

Disclaimer

ContractWolf.io audits and reports should not be considered as a form of project's "advertisement" and does not cover any interaction and assessment from "project's contract" to "external contracts" such as Pancakeswap or similar.

ContractWolf does not provide any warranty on its released reports.

ContractWolf should not be used as a <u>decision</u> to invest into an audited project and is not affiliated nor partners to its audited contract projects.

ContractWolf provides transparent report to all its "clients" and to its "clients participants" and will not claim any guarantee of bug-free code within it's **SMART CONTRACT**.

ContractWolf presence is to analyze, audit and assess the client's smart contract's code.

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

Network

Binance Smart Chain (BEP20)

Website

https://anunnakiaki.com/

Twitter

https://twitter.com/anunnaki_aki

Telegram

https://www.t.me/anunnakiakiofficiall

https://t.me/anunnakiaki

Description

Anunnaki, a BEP-20, is a decentralized cryptocurrency platform of the modern world based on the Binance smart chain (BSC), a sovereign blockchain. Anunnaki is designed as a high-performance to provide many services in one platform.

rm.ContractWolf Engagement

8th of March 2022, **Anunnaki** engaged and agrees to audit their smart contract's code by ContractWolf. The goal of this engagement was to identify if there is a possibility of security flaws in the implementation of the contract or system.

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

Logo

Contract link:

https://bscscan.com/address/0x3d8114ea0210237bA04830c8C2C36c663 AB2f772

Risk Level Classification

Risk Level represents the classification or the probability that a certain function or threat that can exploit vulnerability and have an impact within the system or contract.
Risk Level is computed based on CVSS Version 3.0

Level	Value	Vulnerability
Critical	9 - 10	An exposure that can affect the contract functions in several events that can risk and disrupt the contract
High	7 - 8.9	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
Medium	4 - 6.9	An opening that could affect the outcome in executing the contract in a specific situation
Low	0.1 - 3.9	An opening but doesn't have an impact on the functionality of the contract
Informational	0	An opening that consists of information's but will not risk or affect the contract

Auditing Approach

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

2. Testing and automated analysis that includes:

- Testing the smart contract functions with common test cases and scenarios, to ensure that it returns the expected results.

3. Best practices 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 secure the smart contract.

Used Code from other Frameworks/Smart Contracts (Direct Imports)

Imported Packages

- IBEP20
- IBEP20Metadata
- Context
- Safemath
- SafemathUint
- SafemathInt
- BEP20
- IPancakeswapV2Factory
- IPancakeswapV2Pair
- IPancakeswapV2Router01
- IPancakeswapV2Router02
- DividendPayingTokenInterface
- DividendPayingTokenOptionalInterface
- DividendPayingToken
- IterableMapping
- Ownable
- AkiDividendTracker
- Aki

Description

Optimization enabled: Yes

Version: v0.6.12

Decimal: 9

Symbol: $AKI

Capabilities

Components

Version	Contracts	Libraries	Interfaces	Abstract
1.0	4	4	8	1

Exposed Functions

Version	Public	Private
1.0	51	7

Version	External	Internal
1.0	93	29

State Variables

Version	Total	Public
1.0	34	18

Capabilities

Version	Solidity Versions Observed	Experimental Features	Can Receive Funds	Uses Assembly	Has Destroyable Contracts
1.0	^0.6.12		Yes	No	No

Scope of Work

Anunnaki's 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.

Inheritance Graph
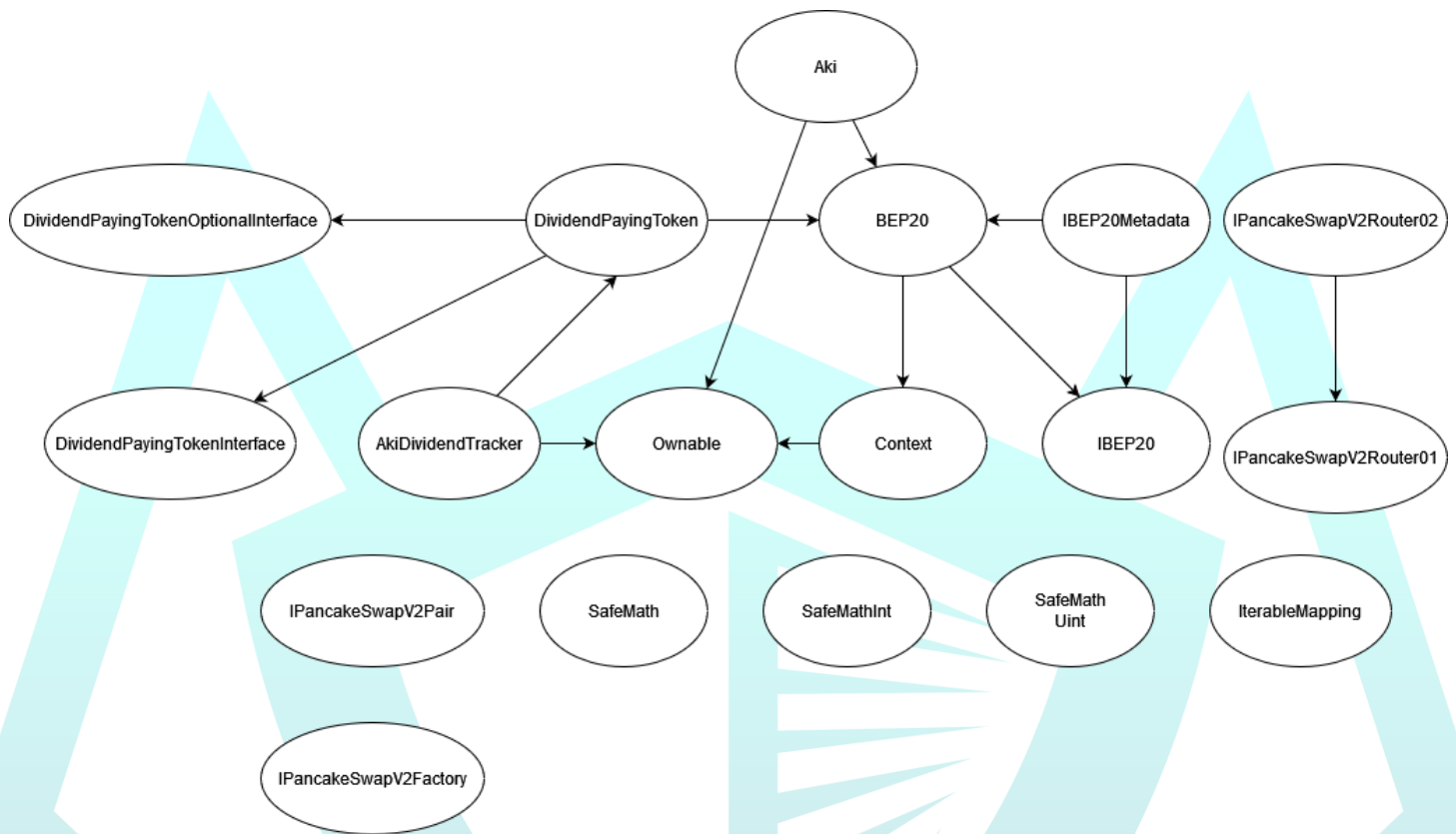
Verify Claims

Correct implementation of Token Standard

Tested	Verified
✓	✗

Function	Description	Exist	Tested	Verified
TotalSupply	Information about the total coin or token supply	✓	✓	✓
BalanceOf	Details on the account balance from a specified address	✓	✓	✓
Transfer	An action that transfers a specified amount of coin or token to a specified address	✓	✓	✓
TransferFrom	An action that transfers a specified amount of coin or token from a specified address	✓	✓	✓
Approve	Provides permission to withdraw specified number of coin or token from a specified address	✓	✓	✓

Optional implementation

Function	Description	Exist	Tested	Verified
renounceOwnership	Owner renounce ownership for more trust	✓	✓	✓

Deployer cannot mint any new tokens after deployment

Statement	Exist	Tested	Verified	File
Deployer can't mint	✓	✓	✓	Main
Deployer can't generate NFT	✓	✓	✓	Main

Max / Total supply: 1,000,000,000

Deployer cannot burn or lock user funds

Statement	Exist	Tested	Verified
Deployer cannot lock	✓	✓	✓
Deployer cannot burn	✓	✓	✓

Deployer cannot pause contract

Statement	Exist	Tested	Verified
Deployer cannot pause	✓	✓	✓

Overall Checkup (Smart Contract Security)

Tested	Verified
✓	✓

Legend

Attribute	Symbol
Verified / Checked	✓
Partly Verified	✗
Unverified / Not checked	⚑
Not Available	—

Write Functions of Contract

1. approve

2. claim

3. decreaseAllowance

4. excludeFromFees

5. excludeMultipleAccountsFromFees

6. increaseAllowance

7. processDividendTracker

8. renounceOwnership

9. setAutomatedMarketMakerPair

10. setMaxtxn

11. setNewFee

12. setSwapAt

13. transfer

14. transferFrom

15. transferOwnership

16. updateClaimWait

17. updateDividendTracker

18. updateGasForProcessing

19. updateLiquidityWallet

20. updatePancakeswapV2Router

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	NOT 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	NOT PASSED
SWC-127	Arbitrary Jump with Function Type Variable	CWE-695: Use of Low-Level Functionality	PASSED
SWC-126	Insufficient Gas Griefing	CWE-691: Insufficient Control Flow Management	NOT 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	NOT 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	NOT 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	Delegate call to Untrusted Callee	CWE-829: Inclusion of Functionality from Untrusted Control Sphere	PASSED
SWC-111	Use of Deprecated	CWE-477: Use of Obsolete Function	PASSED

	Solidity Functions		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	PASSED
SWC-102	Outdated Compiler Version	CWE-937: Using Components with Known Vulnerabilities	PASSED

SWC-101	Integer Overflow and Underflow	CWE-682: Incorrect Calculation	**PASSED**
SWC-100	Function Default Visibility	CWE-710: Improper Adherence to Coding Standards	**PASSED**

AUDIT PASSED

Critical Issues

No critical issues found

High Issues

No high issues found

Medium Issues

No medium issues found

Low Issues

No low issues found

Informational Issues

No informational issues found

Function Issues

No informational issues found

Audit Comments

Owner can't disable trading / Contract is pausable.

Owner can't mint new tokens.

Owner can set buy/sell fees with an indefinite amount.

Owner can set max transaction limit.

Read report for more information.