CONTRACT WOLF

Security Assessment

CHUCKLE

Verified on 03/20/2025

SUMMARY

Project	CHAIN	METHODOLOGY
CHUCKLE	Solana	Manual & Automatic Analysis
FILES	DELIVERY	TYPE
Single	03/20/2025	Standard Audit

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

🟥 0 Critical	An exposure that can affect the contract functions in several events that can risk and disrupt the contract
🟧 0 Major	An opening & exposure to manipulate 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
🟦 0 Informational	An opening that consists information but will not risk or affect the contract
🟩 0 Resolved	ContractWolf's findings has been acknowledged & resolved by the project

STATUS ✔️**AUDIT PASSED**

TABLE OF CONTENTS | CHUCKLE

Summary

Project Information

Findings

DISCLAIMER | CHUCKLE

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 | CHUCKLE

CHUCKLE's 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 **CHUCKLE**.

AUDITING APPROACH | CHUCKLE

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 | CHUCKLE

The Chuckle project is all about mixing crypto with some good ol' humor. We're not just another meme token; we're here to shake up the space and have a blast doing it!

Token Name	Symbol	Decimal	Total Supply	Chain
CHUCKLE	CHK	-	-	Solana

SOURCE

Source	*Sent Via local-files*

FINDINGS | CHUCKLE

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

This report has been prepared to state the issues and vulnerabilities for CHUCKLE 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
RCW-003	Integer overflow & underflow	-	Minor	● Pending

CW RUST ASSESSMENT | CHUCKLE

ContractWolf Vulnerability for Rust and Security Test Cases

Relevant & known up-to-date issues for rust language

ID	Name	Description	Status
RCW-001	Reentrancy	a malicious contract calls back into the calling contract before the first invocation of the function is finished.	✓
RCW-002	Undefined behavior	The Rust reference contains a non-exhaustive list of behaviors considered undefined in Rust	✓
RCW-003	Integer overflow & underflow	Overflow/Underflow of mathematical operations inside the rust smart contract	✓
RCW-004	Out of bounds read/write	The contract or function reads data past the end or before beginning of the intended buffer	✓
RCW-005	Memory Corruption	Wrong usage of memory model throughout the contract or within its functions.	✓
RCW-006	Typographical Error	Unintended error for contract, function names, code and arithmetic inputs	✓
RCW-007	Hash Collisions With Multiple Arguments	If used incorrectly, triggers a hash collision while calling a function within a function.	✓
CVE-2021-39137	Erroneous Computation	Incorrect math calculation	✓
CVE-2022-37450	Function Manipulation	Manipulation attack of time-difference values to increase rewards	✓
CVE-2022-23328	Denial of Service	DDoS Attack using pending transactions	✓
CVE-2022-29177	High verbosity logging	The product does not properly control the allocation and maintenance of a limited resource, thereby enabling an actor to influence the amount of resources consumed, eventually leading to the exhaustion of available resources.	✓

ContractWolf follows the safety protocols from ***NVD****(National Vulnerability Database) &* ***CVE Details*** *for* ***RUST Language*** *to assess and identify the security risk for rust smart contracts.*

FIXES & RECOMMENDATION

RCW-003 | Integer overflow & underflow

Precision Loss in Token Amounts:

Using `Number` and `Math.pow()` for token calculations can lead to precision loss with large values (e.g., high decimals). This can result in incorrect minting/transfers.

On Solana, token amounts are stored as integers (e.g., 1 SOL = 1_000_000_000 lamports).

It can loss precision & rounding if handled incorrectly (sample below)

```
// Incorrect way (using Number):
const amount = 100.5 * 10 ** 9; // 100.5 tokens with 9 decimals
console.log(amount); // Output: 100500000000 (correct)

const badAmount = 12345678901234567.89 * 10 ** 9;
console.log(badAmount); // Output: 12345678901234568000000000 (wrong!)
```

Recommendation :

Use `BigInt` for all token amount calculations (minting, transfers).

```
// In createToken():
createMintToInstruction(
  mintKeypair.publicKey,
  tokenATA,
  ownerPubkey,
  BigInt(totalSupply) * (10n ** BigInt(decimals)) // Use BigInt
);

// In disperseToken():
const amounts = [
  BigInt(process.env.PRIVATE_PRESALE_AMOUNT) * (10n ** BigInt(tokenDecimal)),
  // same with others under this const
];
```

AUDIT COMMENTS | CHUCKLE

Smart Contract audit comment for a non-technical perspective

🟢 Owner can change the contract settings after deployment

CONTRACTWOLF

Blockchain Security - Smart Contract Audits