



**CONTRACT**  
WOLF

## SUMMARY

Project

BlockJack

CHAIN

Arbitrum

METHODOLOGY

Manual &amp; Automatic Analysis

FILES

Single

DELIVERY

04/14/2023

TYPE

Standard Audit



1

0

1

0

0

0

Total Findings

Critical

Major

Medium

Minor

Informational

0 Critical

0 Pending

An exposure that can affect the contract functions in several events that can risk and disrupt the contract

0 Major

0 Pending

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

1 Medium

0 Pending

An opening that could affect the outcome in executing the contract in a specific situation

0 Minor

0 Pending

An opening but doesn't have an impact on the functionality of the contract

0 Informational

5 Pending

An opening that consists information but will not risk or affect the contract

**STATUS**
**✓ AUDIT PASSED**

# TABLE OF CONTENTS | BlockJack

## | Summary

Project Summary  
Findings Summary  
Disclaimer  
Scope of Work  
Auditing Approach

## | Project Information

Token/Project Details  
Inheritance Graph  
Call Graph

## | Findings

Issues  
SWC Attacks  
CW Assessment  
Fixes & Recommendation  
Audit Comments

## DISCLAIMER | BlockJack

**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.

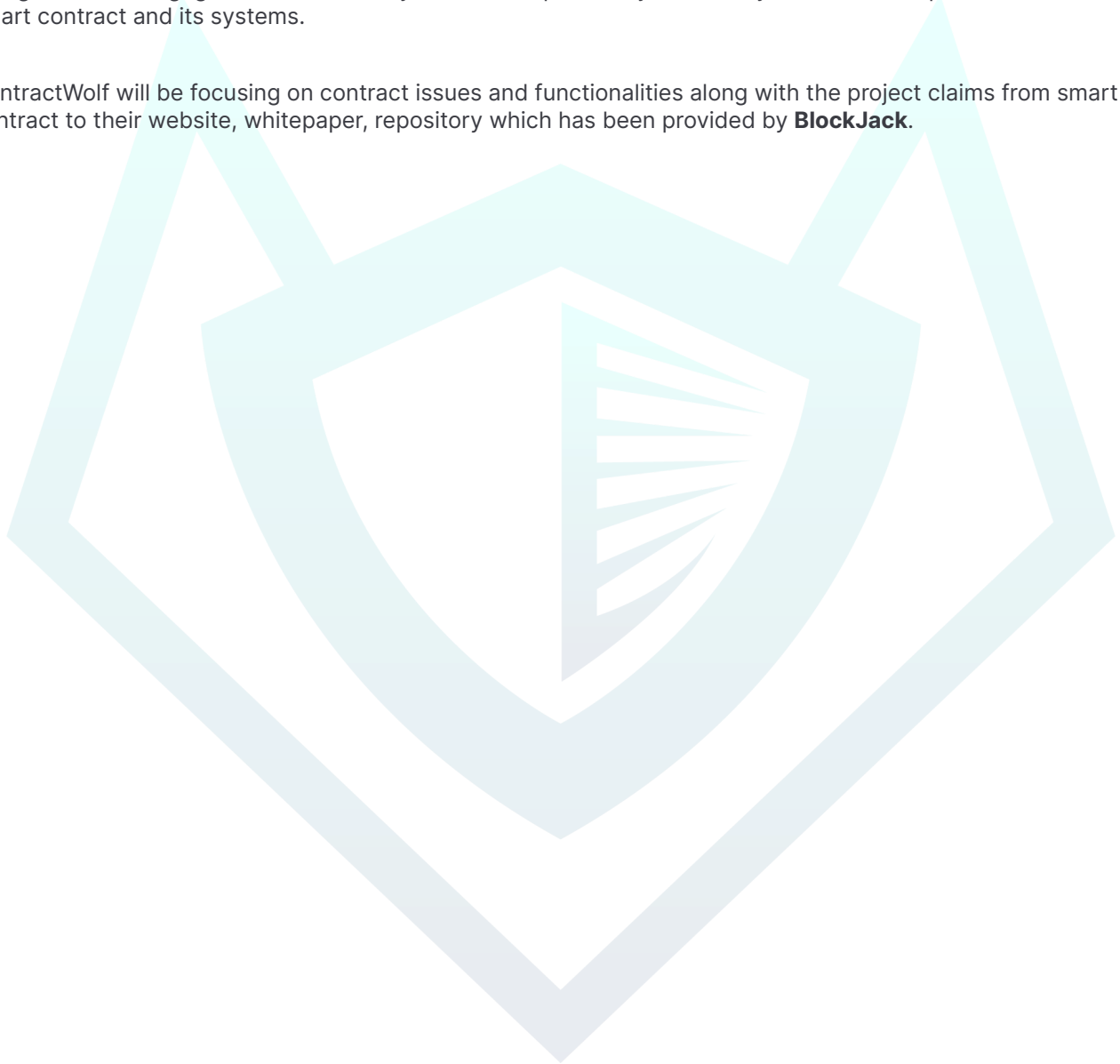
*Each company or project should be liable to its security flaws and functionalities.*

## SCOPE OF WORK | BlockJack

**BlockJack** 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 **BlockJack**.



## AUDITING APPROACH | BlockJack

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 | BlockJack



BlockJack provides the most user-friendly, fair and fun blackjack experience. Powered by the blockchain and completely playable within Telegram, BlockJack brings the casino experience right to your hands.

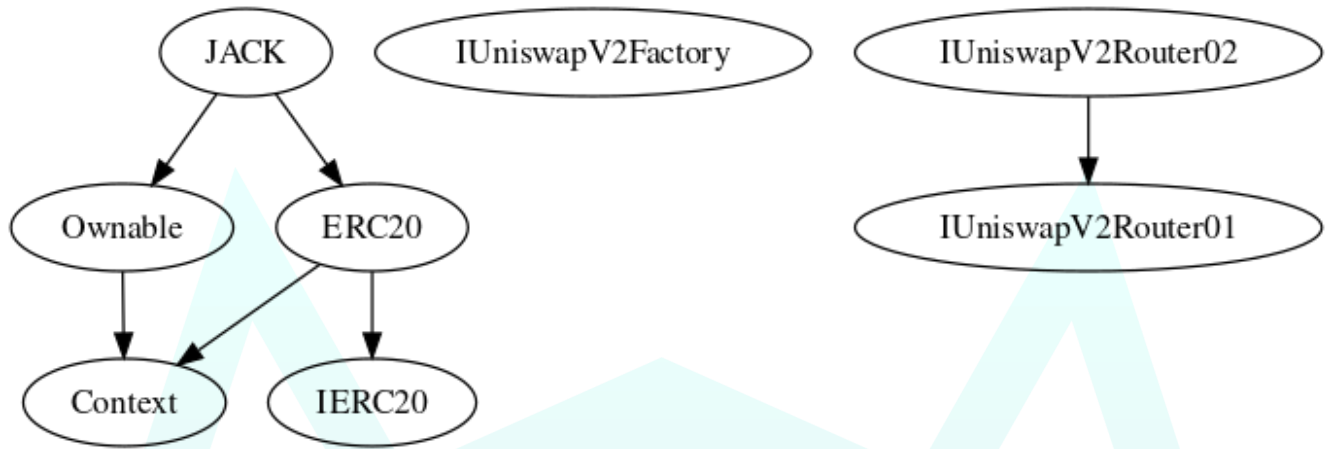
Token Name	Symbol	Decimal	Total Supply	Chain
MyToken	JACK	18	100,000,000	Arbitrum

## SOURCE

Source *Sent Via local-files*

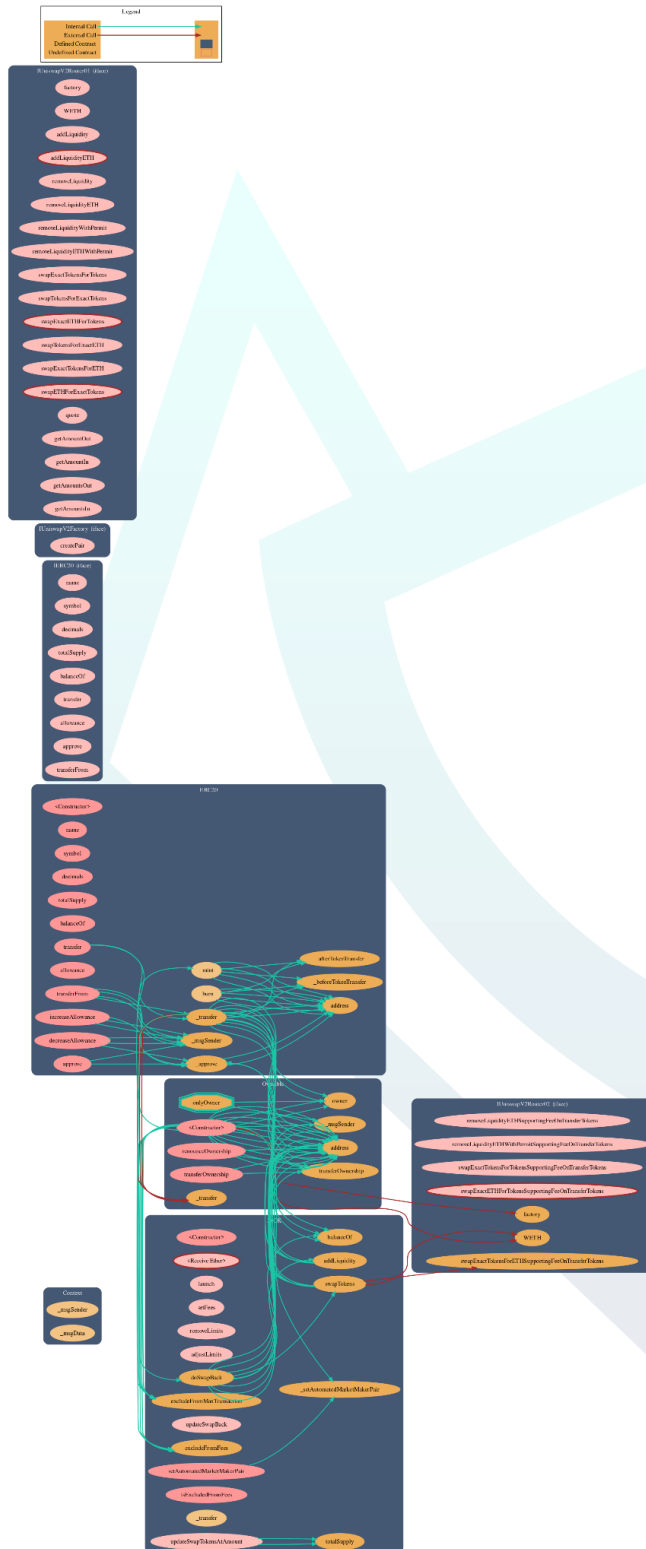
# INHERITANCE GRAPH | BlockJack

Inheritance Graph of Contract Functions





## Call Graph of Contract Functions



# FINDINGS | BlockJack



1

0

1

0

0

0

Total Findings

Critical

Major

Medium

Minor

Informational

This report has been prepared to state the issues and vulnerabilities for BlockJack 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-107	Reentrancy	BlockJack.sol, L: 574, L: 721	Major	<ul style="list-style-type: none"> <li>Pending</li> </ul>

# SWC ATTACKS | BlockJack

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	● Passed
SWC-103	Floating Pragma	● Passed
SWC-104	Unchecked Call Return Value	● Passed
SWC-105	Unprotected Ether Withdrawal	● Passed
SWC-106	Unprotected SELF DESTRUCT Instruction	● Passed
SWC-107	Reentrancy	● Not 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 | BlockJack

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 Solidity v0.8+. The compiler now has Built in overflow checking.	✓

## FIXES & RECOMMENDATION

### SWC-107 | Reentrancy

Functions `_transfer` and `doSwapBack` Reentrancy detected through external calls.

#### Recommendation

Make sure all internal state changes are performed before the call is executed or;  
Use OpenZeppelin's ReentrancyGuard



Taxes can be updated to 100%

Owner can update fees up to 100%

```
function setFees(
    uint8 _buyLiquidityFee,
    uint8 _buyPoolFee,
    uint8 _buyDevelopmentFee,
    uint8 _sellLiquidityFee,
    uint8 _sellPoolFee,
    uint8 _sellDevelopmentFee,
    uint8 _buyLotteryFee,
    uint8 _sellLotteryFee
) external onlyOwner {
    buyLiquidityFee = _buyLiquidityFee;
    buyPoolFee = _buyPoolFee;
    buyDevelopmentFee = _buyDevelopmentFee;
    buyLotteryFee = _buyLotteryFee;
    buyTotalFees = buyLiquidityFee + buyPoolFee + buyDevelopmentFee +
buyLotteryFee;

    sellLiquidityFee = _sellLiquidityFee;
    sellPoolFee = _sellPoolFee;
    sellDevelopmentFee = _sellDevelopmentFee;
    sellLotteryFee = _sellLotteryFee;
    sellTotalFees = sellLiquidityFee + sellPoolFee + sellDevelopmentFee +
sellLotteryFee;
}
```

### Recommendation

Adding validation checks to the setFees function to ensure that the sum of all fees is not greater to a specific percentage. This will help prevent errors and ensure that the contract works as intended.

Max transaction amount and wallet amount can be set to zero can be used to pause transactions

Owner can update max transaction amount and max wallet amount with an indefinite amount

```
function adjustLimits(uint32 newMaxTx, uint32 newMaxWallet) external onlyOwner {  
    maxTransactionAmount = newMaxTx;  
    maxWallet = newMaxWallet;  
}
```

### Recommendation

Adding validation checks to the maxTransactionAmount and maxWallet parameters to ensure they are not equal to zero and are greater than zero can prevent unintended behavior or error.



## AUDIT COMMENTS | BlockJack

Smart Contract audit comment for a non-technical perspective

- Owner can renounce and transfer ownership
- Owner can launch contract
- Owner can update fees up to 100%
- Owner can remove limits for max transaction and max wallet amount
- Owner can update max transaction amount and max wallet amount with an indefinite amount
- Owner can update minimum swap amount between 0.001% and 0.05% of total supply
- Owner can exclude/include addresses from max transaction amount
- Owner can toggle swap back
- Owner can exclude/include addresses from fees
- Owner can update and toggle automated market maker pair address
- Owner cannot burn tokens
- Owner cannot pause contract
- Owner cannot mint after initial deployment
- Owner cannot set max transaction limit
- Owner cannot block users



# CONTRACTWOLF

**Blockchain Security - Smart Contract Audits**