



# X365

## SMART CONTRACT AUDIT



March 28, 2022



# INTRODUCTION

---

<b>Client</b>	X365
<b>Language</b>	Solidity
<b>Contract Address</b>	0x01e94d96328D63e10Cc5c517Db4F4f4b1B873282
<b>Decimals</b>	18
<b>Supply</b>	5,000,000,000
<b>Platform</b>	Binance Smart Chain
<b>Compiler</b>	v0.7.4+commit.3f05b770
<b>Optimization</b>	Yes, with 200 runs

# TABLE OF CONTENTS

## 01 INTRODUCTION

---

Introduction	02
Approach	04
Risk classification	05

## 02 ABSTRACT

---

Abstract	06
----------	----

## 03 VULNERABILITIES TEST

---

Vulnerabilities Test	07
----------------------	----

## 04 MANUAL ANALYSIS

---

Manual analysis	09
Contract Inspection	10
Inheritance Tree	13
Important Snippets	14
Good Practices	15

## 05 CONCLUSIONS

---

Disclaimer	16
Audit Results	17
Score	18
Summary	19

# Approach

---



## Audit Details

Our comprehensive audit report provides a full overview of the audited system's architecture, smart contract codebase, and details on any vulnerabilities found within the system.

---



## Audit Goals

The audit goal is to ensure that the project is built to protect investors and users, preventing potentially catastrophic vulnerabilities after launch, that lead to scams and rugpulls.

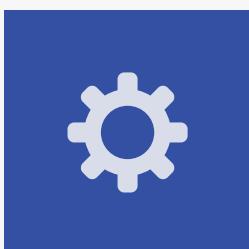
---



## Code Quality

Our analysis includes both automatic tests and manual code analysis for the following aspects:

- Exploits
  - Back-doors
  - Vulnerability
  - Accuracy
  - Readability
- 



## Tools

- Remix IDE
- MythX, Mytrhl
- SWC Registry
- Open Zeppelin Code Analyzer
- Solidity Code Complier

# RISK CLASSIFICATION

---

## CRITICAL

---

Issues on this level are critical to the smart contract's performance/functionality and should be fixed before moving to a live environment.

## MEDIUM

---

Issues on this level could potentially bring problems and should eventually be fixed.

## MINOR

---

Issues on this level are minor details and warning that can remain unfixed but would be better fixed at some point in the future

## INFORMATIONAL

---

Information level is to offer suggestions for improvement of efficacy or security for features with a risk free factor.

# ABSTRACT

---



# Vulnerabilities Test

SWC ID	Description	
<b>SWC-100</b>	Function Default Visibility	<b>Passed</b>
<b>SWC-101</b>	Integer Overflow and Underflow	<b>Passed</b>
<b>SWC-102</b>	Outdated Compiler Version	<b>Passed</b>
<b>SWC-103</b>	FloatingPragma	<b>Minor</b>
<b>SWC-104</b>	Unchecked Call Return Value	<b>Passed</b>
<b>SWC-105</b>	Unprotected Ether Withdrawal	<b>Passed</b>
<b>SWC-106</b>	Unprotected SELF-DESTRUCT Instruction	<b>Passed</b>
<b>SWC-107</b>	Re-entrancy	<b>Passed</b>
<b>SWC-108</b>	State Variable Default Visibility	<b>Minor</b>
<b>SWC-109</b>	Uninitialized Storage Pointer	<b>Passed</b>
<b>SWC-110</b>	Assert Violation	<b>Passed</b>
<b>SWC-111</b>	Use of Deprecated Solidity Functions	<b>Passed</b>
<b>SWC-112</b>	Delegate Call to Untrusted Callee	<b>Passed</b>
<b>SWC-113</b>	DoS with Failed Call	<b>Passed</b>
<b>SWC-114</b>	Transaction Order Dependence	<b>Passed</b>
<b>SWC-115</b>	Authorization through tx.origin	<b>Passed</b>

<b>SWC-116</b>	Block values as a proxy for time	<b>Passed</b>
<b>SWC-117</b>	Signature Malleability	<b>Passed</b>
<b>SWC-118</b>	Incorrect Constructor Name	<b>Passed</b>
<b>SWC-119</b>	Shadowing State Variables	<b>Passed</b>
<b>SWC-120</b>	Weak Sources of Randomness from Chain Attributes	<b>Passed</b>
<b>SWC-121</b>	Missing Protection against Signature Replay Attacks	<b>Passed</b>
<b>SWC-122</b>	Lack of Proper Signature Verification	<b>Passed</b>
<b>SWC-123</b>	Requirement Violation	<b>Passed</b>
<b>SWC-124</b>	Write to Arbitrary Storage Location	<b>Passed</b>
<b>SWC-125</b>	Incorrect Inheritance Order	<b>Passed</b>
<b>SWC-126</b>	Insufficient Gas Griefing	<b>Passed</b>
<b>SWC-127</b>	Arbitrary Jump with Function Type Variable	<b>Passed</b>
<b>SWC-128</b>	DoS With Block Gas Limit	<b>Passed</b>
<b>SWC-129</b>	Typographical Error	<b>Passed</b>
<b>SWC-130</b>	Right-To-Left-Override control character (U+202E)	<b>Passed</b>
<b>SWC-131</b>	Presence of unused variables	<b>Passed</b>
<b>SWC-132</b>	Unexpected Ether balance	<b>Passed</b>
<b>SWC-133</b>	Hash Collisions With Multiple Variable Length Arguments	<b>Passed</b>
<b>SWC-134</b>	Message call with the hardcoded gas amount	<b>Passed</b>
<b>SWC-135</b>	Code With No Effects (Irrelevant/Dead Code)	<b>Passed</b>
<b>SWC-136</b>	Unencrypted Private Data On-Chain	<b>Passed</b>

# MANUAL ANALYSIS

The contract is verified to check if functions do and work as they should and malicious code is not inserted.

	Tested	Result
<b>Transfer</b>	Yes	<b>Passed</b>
<b>Total Supply</b>	Yes	<b>Passed</b>
<b>Buy Back</b>	Yes	<b>N/A</b>
<b>Burn</b>	Yes	<b>N/A</b>
<b>Mint</b>	Yes	<b>N/A</b>
<b>Rebase</b>	Yes	<b>Low</b>
<b>Pause</b>	Yes	<b>N/A</b>
<b>Blacklist</b>	Yes	<b>N/A</b>
<b>Lock</b>	Yes	<b>N/A</b>
<b>Max Transaction</b>	Yes	<b>N/A</b>
<b>Transfer Ownership</b>	Yes	<b>Passed</b>
<b>Renounce Ownership</b>	Yes	<b>Passed</b>

MANUAL AUDIT

# CONTRACT INSPECTION



\*\*SafeMathInt\*\*	Library					
L	mul	Internal 🔒				
L	div	Internal 🔒				
L	sub	Internal 🔒				
L	add	Internal 🔒				
L	abs	Internal 🔒				
\*\*IERC20\*\*	Interface					
L	totalSupply	External !	NO!			
L	balanceOf	External !	NO!			
L	allowance	External !	NO!			
L	transfer	External !	○	NO!		
L	approve	External !	○	NO!		
L	transferFrom	External !	○	NO!		
\*\*SafeMath\*\*	Library					
L	add	Internal 🔒				
L	sub	Internal 🔒				
L	sub	Internal 🔒				
L	mul	Internal 🔒				
L	div	Internal 🔒				
L	div	Internal 🔒				
L	mod	Internal 🔒				
\*\*InterfaceLP\*\*	Interface					
L	sync	External !	○	NO!		
\*\*Roles\*\*	Library					
L	add	Internal 🔒	○			
L	remove	Internal 🔒	○			
L	has	Internal 🔒				
\*\*ERC20Detailed\*\*	Implementation	IERC20				
L	<Constructor>	Public !	○	NO!		
L	name	Public !	NO!			
L	symbol	Public !	NO!			
L	decimals	Public !	NO!			

```

| **IDEXRouter** | Interface | || | | |
| L | factory | External | | NO | |
| L | WETH | External | | NO | |
| L | addLiquidity | External | | ● | NO | |
| L | addLiquidityETH | External | | ● | NO | |
| L | swapExactTokensForTokensSupportingFeeOnTransferTokens | External | | ● | NO | |
| L | swapExactETHForTokensSupportingFeeOnTransferTokens | External | | ● | NO | |
| L | swapExactTokensForETHSupportingFeeOnTransferTokens | External | | ● | NO | |
|||||
| **IDEXFactory** | Interface | ||
| L | createPair | External | | ● | NO | |
|||||
| **Ownable** | Implementation | ||
| L | <Constructor> | Public | | ● | NO | |
| L | owner | Public | | NO | |
| L | renounceOwnership | Public | | ● | onlyOwner | |
| L | transferOwnership | Public | | ● | onlyOwner | |
| L | _transferOwnership | Internal | 🔒 | ● | |
|||||
| **X365** | Implementation | ERC20Detailed, Ownable ||
| L | <Constructor> | Public | | ● | ERC20Detailed | |
| L | <Receive Ether> | External | | ● | NO | |
| L | totalSupply | External | | NO | |
| L | allowance | External | | NO | |
| L | balanceOf | Public | | NO | |
| L | checkFeeExempt | External | | NO | |
| L | checkSwapThreshold | External | | NO | |
| L | shouldRebase | Internal | 🔒 | | |
| L | shouldTakeFee | Internal | 🔒 | | |
| L | shouldSwapBack | Public | | NO | |
| L | getCirculatingSupply | Public | | NO | |
| L | getLiquidityBacking | Public | | NO | |
| L | isOverLiquified | Public | | NO | |
| L | manualSync | Public | | ● | NO | |
| L | transfer | External | | ● | validRecipient | |
| L | _basicTransfer | Internal | 🔒 | ● | |
| L | _transferFrom | Internal | 🔒 | ● | |
| L | transferFrom | External | | ● | validRecipient | |
| L | _swapAndLiquify | Private | 🔒 | ● | |
| L | _addLiquidity | Private | 🔒 | ● | |

```

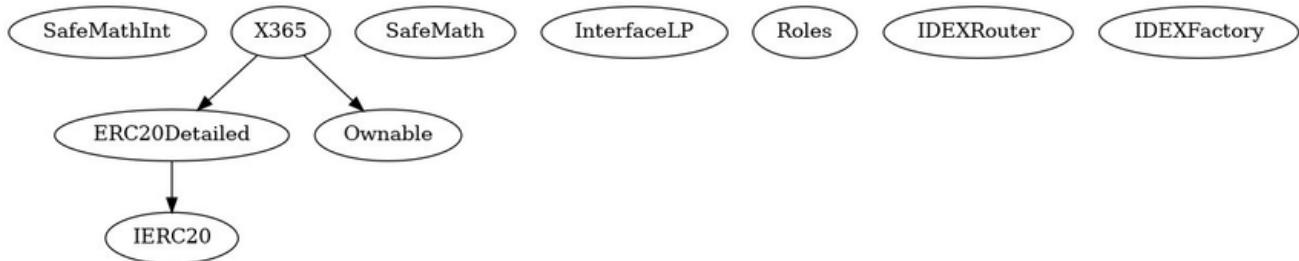
```

| L | _addLiquidityBUSD | Private 📰 | 🔴 | | |
| L | _swapTokensForBNB | Private 📰 | 🔴 | |
| L | _swapTokensForBUSD | Private 📰 | 🔴 | |
| L | swapBack | Internal 🔒 | 🔴 | swapping |
| L | takeFee | Internal 🔒 | 🔴 | |
| L | decreaseAllowance | External ! | 🔴 | NO! | |
| L | increaseAllowance | External ! | 🔴 | NO! | |
| L | approve | External ! | 🔴 | NO! | |
| L | _rebase | Private 📰 | 🔴 | |
| L | coreRebase | Private 📰 | 🔴 | |
| L | setAutomatedMarketMakerPair | Public ! | 🔴 | onlyOwner | |
| L | setFeeExempt | External ! | 🔴 | onlyOwner | |
| L | setTargetLiquidity | External ! | 🔴 | onlyOwner | |
| L | setSwapBackSettings | External ! | 🔴 | onlyOwner | |
| L | setFeeReceivers | External ! | 🔴 | onlyOwner | |
| L | clearStuckBalance | External ! | 🔴 | onlyOwner | |
| L | rescueToken | External ! | 🔴 | onlyOwner | |
| L | setAutoRebase | External ! | 🔴 | onlyOwner | |
| L | setRebaseFrequency | External ! | 🔴 | onlyOwner | |
| L | setRewardYield | External ! | 🔴 | onlyOwner | |
| L | setFeesOnNormalTransfers | External ! | 🔴 | onlyOwner | |
| L | setIsLiquidityInBnb | External ! | 🔴 | onlyOwner | |
| L | setNextRebase | External ! | 🔴 | onlyOwner |

```

Symbol	Meaning
🔴	Function can modify state
💵	Function is payable
🔒	Private function
🔓	Internal function
NO!	Function has no modifier

# INHERITANCE TREE



Inheritance is a feature of the object-oriented programming language. It is a way of extending the functionality of a program, used to separate the code, reduces the dependency, and increases the re-usability of the existing code. Solidity supports inheritance between smart contracts, where multiple contracts can be inherited into a single contract.

# Important Snippets



## Owner can set rebase frequency

```
function setRebaseFrequency(uint256 _rebaseFrequency) external onlyOwner {  
    require(_rebaseFrequency <= MAX_REBASE_FREQUENCY, "Too high");  
    rebaseFrequency = _rebaseFrequency;  
}  
function setNextRebase(uint256 _nextRebase) external onlyOwner {  
    nextRebase = _nextRebase;  
}
```

## Owner can set reward Yield

```
function setRewardYield(uint256 _rewardYield, uint256 _rewardYieldDenominator) external onlyOwner {  
    rewardYield = _rewardYield;  
    rewardYieldDenominator = _rewardYieldDenominator;  
}
```

# GOOD PRACTICES ✓

---

- The owner cannot set max Tx amount
- The owner cannot stop or pause the smart contract
- The owner cannot set fees
- The smart contract utilizes "SafeMath" to prevent overflows

```
library SafeMath {  
    function add(uint256 a, uint256 b) internal pure returns (uint256) {  
        uint256 c = a + b;  
        require(c >= a, "SafeMath: addition overflow");  
  
        return c;  
    }  
  
    function sub(uint256 a, uint256 b) internal pure returns (uint256) {  
        return sub(a, b, "SafeMath: subtraction overflow");  
    }  
  
    function sub(  
        uint256 a,  
        uint256 b,  
        string memory errorMessage  
    ) internal pure returns (uint256) {  
        require(b <= a, errorMessage);  
        uint256 c = a - b;  
  
        return c;  
    }  
  
    function mul(uint256 a, uint256 b) internal pure returns (uint256) {  
        if (a == 0) {  
            return 0;  
        }  
  
        uint256 c = a * b;  
        require(c / a == b, "SafeMath: multiplication overflow");  
  
        return c;  
    }  
  
    function div(uint256 a, uint256 b) internal pure returns (uint256) {  
        return div(a, b, "SafeMath: division by zero");  
    }  
  
    function div(  
        uint256 a,  
        uint256 b,  
        string memory errorMessage  
    ) internal pure returns (uint256) {  
        require(b > 0, errorMessage);  
        uint256 c = a / b;  
  
        return c;  
    }  
  
    function mod(uint256 a, uint256 b) internal pure returns (uint256) {  
        require(b != 0);  
        return a % b;  
    }  
}
```

# DISCLAIMER

---

SafuAudit.com is not a financial institution and the information provided on this website does not constitute investment advice, financial advice, trading advice or any other sort of advice. You should not treat any of the website's content as such. Investing in crypto assets carries a high level of risk and does not hold guarantees for not sustaining financial loss due to their volatility.

## Accuracy of Information

SafuAudit will strive to ensure accuracy of information listed on this website although it will not hold any responsibility for any missing or wrong information. SafuAudit provides all information as is. You understand that you are using any and all information available here at your own risk. Any use or reliance on our content and services is solely at your own risk and discretion.

The purpose of the audit is to analyse the on-chain smart contract source code, and to provide basic overview of the project.

While we have used all the information available to us for this straightforward investigation, you should not rely on this report only – we recommend proceeding with several independent audits. Be aware that smart contracts deployed on a blockchain aren't secured enough against 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, SafuAudit 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.

# AUDIT RESULTS

---

## CRITICAL

---

No critical severity issues have been found.

## MEDIUM

---

No medium severity issues have been found.

## MINOR

---

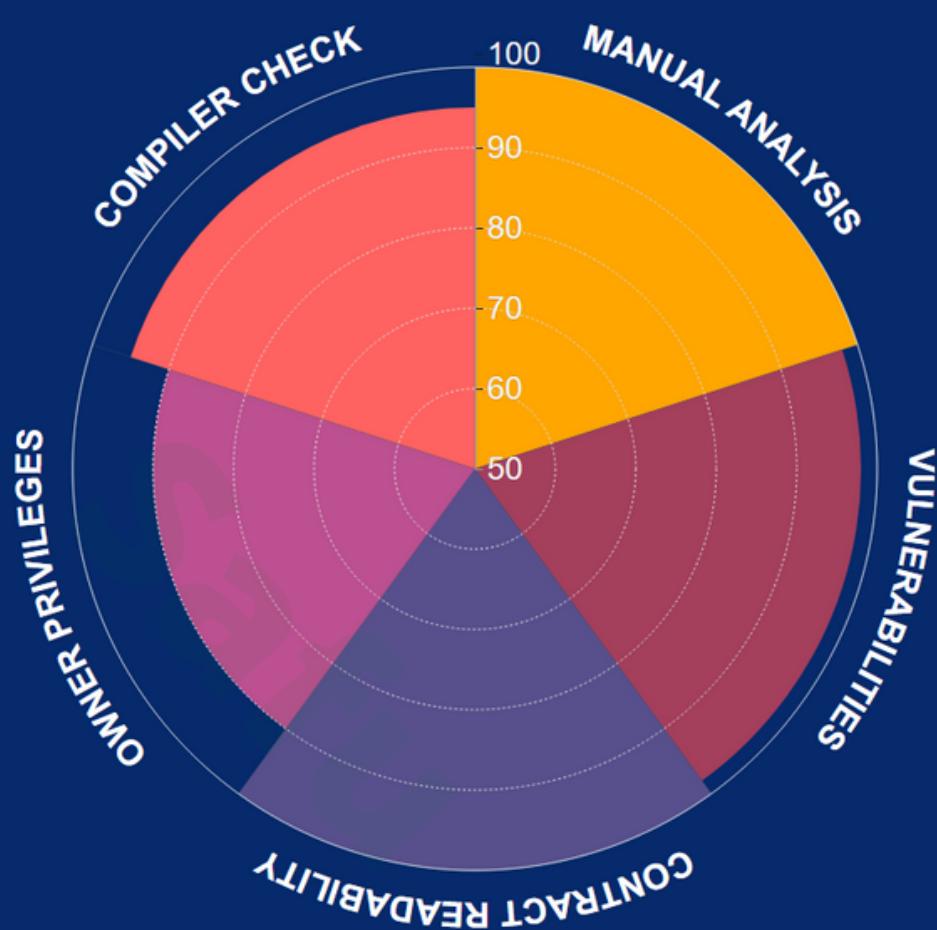
- A floating pragma is set. It is recommended to specify a fixed compiler version to ensure that the bytecode produced does not vary between builds. This is especially important if you rely on bytecode-level verification of the code.
- State variable visibility is not set. It is best practice to set the visibility of state variables explicitly. The default visibility for "\_isFeeExempt, DEAD, ZERO, targetLiquidity, targetLiquidityDenominator, inSwap" is internal.

## INFORMATIONAL

---

The standard audit model does not offer suggestions and consulting for improvements of efficacy.

# SCORE



Manual Analysis



Vulnerabilities



Contract Readability



Owner Privileges



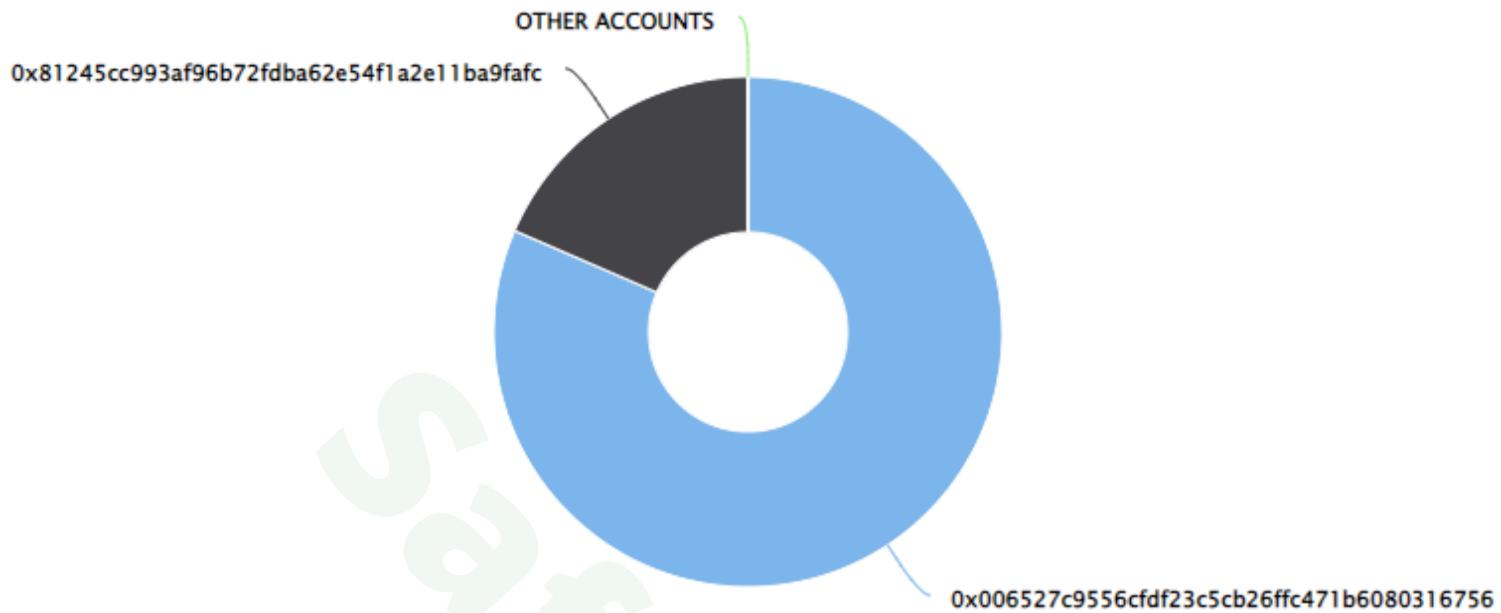
Compiler Check

**Final Score: 93.6**

# SUMMARY

---

## Top 10 holders



Rank	Address	Quantity (Token)	Percentage
1	0x006527c9556cfdf23c5cb26ffc471b6080316756	4,075,000,000	81.5000%
2	0x81245cc993af96b72fdb62e54f1a2e11ba9fafc	925,000,000	18.5000%

## Conclusion

---

Project X365 does not contain any severe issues.

It utilizes automatic rebase function that increases or decreases total supply and can be adjusted according to some parameters that have arbitrary limits set.

SafuAudit has tested the security based on manual and automated tests.

Please note that we don't offer any warranties for business model.



**SafuAudit.com**

