



# Security Assessment SYNCDEX

Vital Block **Verified** on July 11<sup>TH</sup>, 2023

 @Vital-Block

 @VB\_Audit

 info@vitalblock.org




 www.vitalblock.org



PREPARED FOR:  
**SYNCDEX**



## INTRODUCTION

|                    |   |
|--------------------|---|
| Auditing Company   |  <b>VITAL BLOCK SECURITY</b> |
| Client Project     |  <b>SYNCDEx</b>              |
| Methodology        | Automated Analysis, Manual Code Review  |
| Language           | Solidity  |
| License            | MIT   |
| Contracts Address  | <b>SYNCDExTOKEN:</b> 0xe12575BAdf4F3cf659F4D2e6A68D6A671523EAf2   |
| Network            |  <b>ZKSYNC ERA</b>           |
| Compiler Version   | 0.6.12  |
| Zksolc Version     | v1.3.10   |
| Website            | <a href="https://Syncdex.io">https://Syncdex.io</a>   |
| Telegram           | <a href="https://t.me/SyncDexAnnouncement">https://t.me/SyncDexAnnouncement</a>                               |
| Twitter            | <a href="https://twitter.com/Sync_Dex">https://twitter.com/Sync_Dex</a>                                       |
| Discord            | <a href="https://discord.gg/ubEbNDASnu">https://discord.gg/ubEbNDASnu</a>                                     |
| Doc                | <a href="https://medium.com/@syncdex">https://medium.com/@syncdex</a>   |
| Prelim Report Date | July 10 <sup>TH</sup> 2023  |
| Final Report Date  | July 11 <sup>th</sup> 2023  |



Verify the authenticity of this report on our GitHub Repo: <https://www.github.com/vital-block>

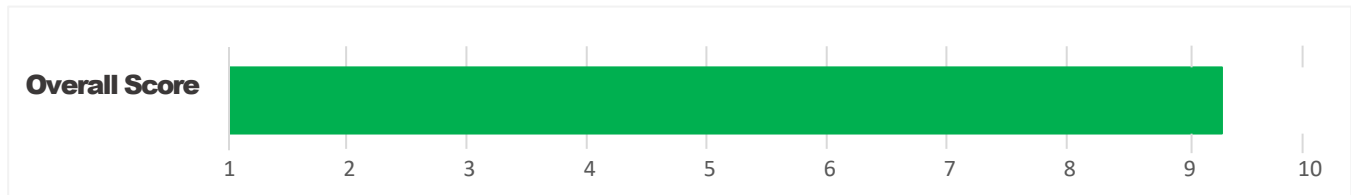



## EXECUTIVE SUMMARY

Vital Block has performed the automated and manual analysis of the SYNCDEX TOKEN Sol code. The code was reviewed for common contract vulnerabilities and centralized exploits. Here's a quick audit summary:

| Status                                | Critical ! 🔴  | Major " 🟡 | Medium # 🟡 | Minor \$ 🟢 | Unknown % 🟤 |
|---------------------------------------|---|-----------|------------|------------|-------------|
| Open                                  | 0   | 0         | 0          | 2          | 2           |
| Acknowledged                          | 0   | 0         | 2          | 2          | 1           |
| Resolved                              | 0   | 0         | 0          | 0          | 0           |
|                                       |   |           |            |            |             |
| Noteworthy<br>OnlyOwner<br>Privileges | Set Taxes and Ratios, Airdrop, Set Protection Settings, Set Reward Properties, Set Reflector Settings, Set Swap Settings, Set Pair and Router |           |            |            |             |

**SYNCDEX TOKEN** Smart contract has achieved the following score: **92.5**



 Please note that smart contracts deployed on blockchains aren't resistant to exploits, vulnerabilities and/or hacks. Blockchain and cryptography assets utilize new and emerging technologies. These technologies present a high level of ongoing risks. For a detailed understanding of risk severity, source code vulnerability, and audit limitations, kindly review the audit report thoroughly.

 Please note that centralization privileges regardless of their inherited risk status - constitute an elevated impact on smart contract safety and security.



## SCOPE OF WORK

Vital Block was consulted by SYNCDEX TOKEN to conduct the smart contract audit of its .Sol source code. The audit scope of work is strictly limited to mentioned .SOL file only:

O SYNCDEXTOKEN.Sol





External contracts and/or interfaces dependencies are not checked due to being out of scope.

Verify audited contract's contract address and deployed link below:

### Contract Address:

0xe12575BAdf4F3cf659F4D2e6A68D6A671523EAf2

Contract Code: SYNCDEXTOKEN.Sol

|              |   |
|--------------|---|
| Project Name |  SYNCDEX TOKEN |
| Blockchain   |  ZKSYNC ERA    |
| Max supply   | 10,000,000  |
| Ticker       | SDN   |

## AUDIT METHODOLOGY

Smart contract audits are conducted using a set of standards and procedures. Mutual collaboration is essential to performing an effective smart contract audit. Here's a brief overview of Vital Block auditing process and methodology:

### CONNECT

- The onboarding team gathers source codes, and specifications to make sure we understand the size, and scope of the smart contract audit.

### AUDIT

- Automated analysis is performed to identify common contract vulnerabilities. We may use the following third-party frameworks and dependencies to perform the automated analysis:
  - Remix IDE Developer Tool
  - Open Zeppelin Code Analyzer
  - SWC Vulnerabilities Registry
  - DEX Dependencies, e.g., Pancakeswap, Uniswap
- Simulations are performed to identify centralized exploits causing contract and/or trade locks.
- A manual line-by-line analysis is performed to identify contract issues and centralized privileges.

We may inspect below mentioned common contract vulnerabilities, and centralized exploits:

|                      |   |
|----------------------|---|
| Centralized Exploits | <ul style="list-style-type: none"><li>○ Token Supply Manipulation</li><li>○ Access Control and Authorization</li><li>○ Assets Manipulation</li><li>○ Ownership Control</li><li>○ Liquidity Access</li><li>○ Stop and Pause Trading</li><li>○ Ownable Library Verification</li></ul> |
|----------------------|---|



### **Common Contract Vulnerabilities**

- **Integer Overflow**
- **Lack of Arbitrary limits**
- **Incorrect Inheritance Order**
- **Typographical Errors**
- **Requirement Violation**
- **Gas Optimization**
- **Coding Style Violations**
- **Re-entrancy**
- **Third-Party Dependencies**
- **Potential Sandwich Attacks**
- **Irrelevant Codes**
- **Divide before multiply**
- **Conformance to Solidity Naming Guides**
- **Compiler Specific Warnings**
- **Language Specific Warnings**

### **REPORT**

- **The auditing team provides a preliminary report specifying all the checks which have been performed and the findings thereof.**
- **The client's development team reviews the report and makes amendments to the codes.**
- **The auditing team provides the final comprehensive report with open and unresolved issues.**

### **PUBLISH**

- **The client may use the audit report internally or disclose it publicly.**






 **It is important to note that there is no pass or fail in the audit, it is recommended to view the audit**

**as an unbiased assessment of the safety of solidity codes.**



## RISK CATEGORIES

Smart contracts are generally designed to hold, approve, and transfer tokens. This makes them very tempting attack targets. A successful external attack may allow the external attacker to directly exploit. A successful centralization-related exploit may allow the privileged role to directly exploit. All risks which are identified in the audit report are categorized here for the reader to review:

| Risk Type  | Definition  |
|--|---|
| <b>Critical</b> !   | These risks could be exploited easily and can lead to asset loss, data loss, asset, or data manipulation. They should be fixed right away.  |
| <b>Major</b> "      | These risks are hard to exploit but very important to fix, they carry an elevated risk of smart contract manipulation, which can lead to high-risk severity.  |
| <b>Medium</b> #     | These risks should be fixed, as they carry an inherent risk of future exploits, and hacks which may or may not impact the smart contract execution. Low-risk re-entrancy-related vulnerabilities should be fixed to deter exploits. |
| <b>Minor</b> \$   | These risks do not pose a considerable risk to the contract or those who interact with it. They are code-style violations and deviations from standard practices. They should be highlighted and fixed nonetheless.                 |
| <b>Unknown</b> %  | These risks pose uncertain severity to the contract or those who interact with it. They should be fixed immediately to mitigate the risk uncertainty.   |

All statuses which are identified in the audit report are categorized here for the reader to review:

| Status Type         | Definition                             |
|---------------------|--|
| <b>Open</b>         | Risks are open.                        |
| <b>Acknowledged</b> | Risks are acknowledged, but not fixed. |
| <b>Resolved</b>     | Risks are acknowledged and fixed.      |



## CENTRALIZED PRIVILEGES

**Centralization risk is the most common cause of cryptography asset loss. When a smart contract has a privileged role, the risk related to centralization is elevated.**

**There are some well-intended reasons have privileged roles, such as:**

- **Privileged roles can be granted the power to `pause()` the contract in case of an external attack.**
- **Privileged roles can use functions like, `include()`, and `exclude()` to add or remove wallets from fees, swap checks, and transaction limits. This is useful to run a presale and to list on an exchange.**

**Authorizing privileged roles to externally-owned-account (EOA) is dangerous. Lately, centralization-related losses are increasing in frequency and magnitude.**

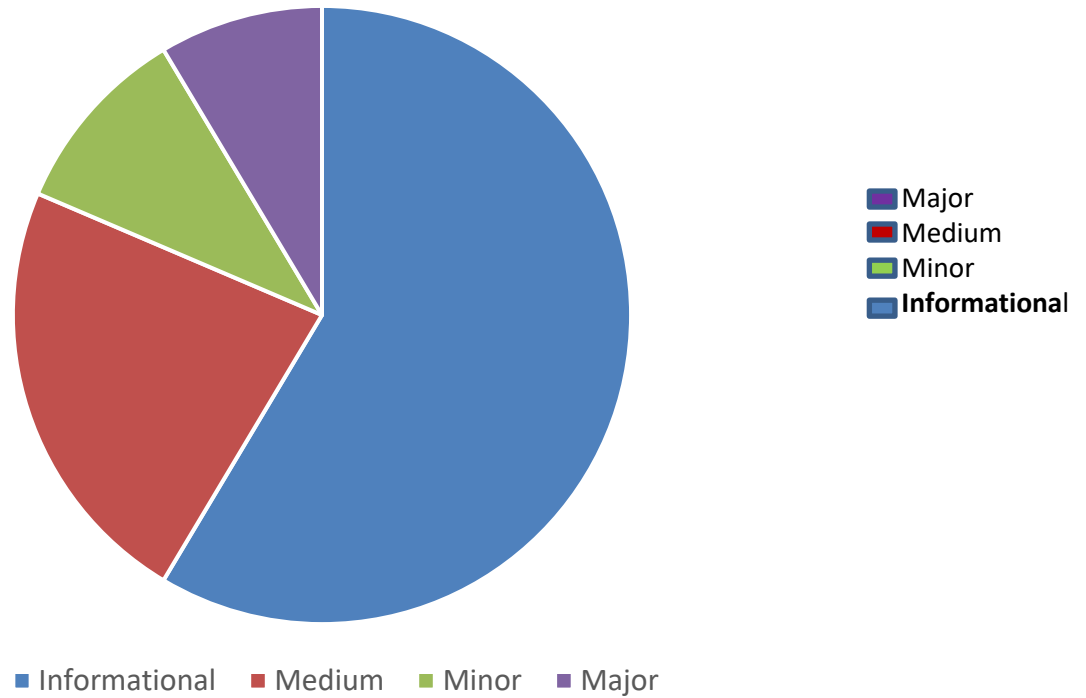
- **The client can lower centralization-related risks by implementing below mentioned practices:**
- **Privileged role's private key must be carefully secured to avoid any potential hack.**
- **Privileged role should be shared by multi-signature (multi-sig) wallets.**
- **Authorized privilege can be locked in a contract, user voting, or community DAO can be introduced to unlock the privilege.**
- **Renouncing the contract ownership, and privileged roles.**
- **Remove functions with elevated centralization risk.**

 **Understand the project's initial asset distribution. Assets in the liquidity pair should be locked. Assets outside the liquidity pair should be locked with a release schedule.**











## Finding Summary








## Status Icon Definitions

|   |              |   |             |   |               |
|---|--------------|---|-------------|---|---------------|
|  | Resolved     |  | In Progress |  | Ignored (pro) |
|  | Not Resolved |  | Incorrect   |  | Ignored (con) |

## Contract Ownership

**0xFa7fF073Ef20c660e75036359E433AC512493290** Is The Owner Of The Contracts.

## Summary

-  Owner is not able to change or set taxes
-  Owner is not able to set a max amount for buys/sells/transfer
-  Owner is not able to pause trades
-  Owner is not able to mint new tokens
-  Owner is not able to blacklist an arbitrary address






## Issues Found

Vital Block Security found that the **SYNCDEX TOKEN** contracts contain no critical issue, no major issues, and 3 minor issue, in addition to 4 informational notes.

We recommend all issues are amended, while the notes are up to the team's discretion, as it refers to best practices.



## AUTOMATED ANALYSIS

| Symbol  | Definition              |
|---|-------------------------|
|  | Function modifies state |
|  | Function is payable     |
|  | Function is internal    |
|  | Function is private     |
|  | Function is important   |

```

| **SYNCDEXTOKEN** | Interface | |||
| L | totalSupply | External | | ! | NO |
| L | decimals | External | | ! | NO |
| L | symbol | External | | ! | NO |
| L | name | External | | ! | NO |
| L | getOwner | External | | NO |
| L | balanceOf | External | | ! | NO |
| L | transfer | External | " | ! | NO |
| L | allowance | External | | ! | NO |
| L | approve | External | " | ! | NO |
| L | transferFrom | External | " | NO |
|||||
| **IFactoryV2** | Interface | |||
| L | getPair | External | | NO |
| L | createPair | External | " | NO |
|||||
| **IV2Pair** | Interface | |||
| L | factory | External | | NO |
| L | getReserves | External | | NO |
| L | sync | External | " | NO |

```



```
|||||
```

```

**IRouter01** | Interface | |||
| L | factory | External | | |NO| |
| L | ETH | External | | |NO| |
| L | addLiquidityETH | External | | # |NO| |
| L | addLiquidity | External | | " |NO| |
| L | swapExactAPTFForTokens | External | | # |NO| |
| L | getAmountsOut | External | | |NO| |
| L | getAmountsIn | External | | |NO| |

```

```
|||||
```

```

**IRouter02** | Interface | IRouter01 |||
| L | swapExactTokensForETHSupportingFeeOnTransferTokens | External | | " |NO| |
| L | swapExactETHForTokensSupportingFeeOnTransferTokens | External | | # |NO| |
| L | swapExactTokensForTokensSupportingFeeOnTransferTokens | External | | " ! |NO| |
| L | swapExactTokensForTokens | External | | " |NO| |

```

```
|||||
```

```

**Protections** | Interface | |||
| L | checkUser | External | | " ! |NO| |
| L | setLaunch | External | | " |NO| |
| L | setLpPair | External | | " |NO| |
| L | SND | External | | " |NO| |
| L | removeSniper | External | | " |NO| |

```

```
|||||
```

```

**Cashier** | Interface | |||
| L | setRewardsProperties | External | | " |NO| |
| L | tally | External | | " |NO| |
| L | load | External | | # |NO| |
| L | cashout | External | | " |NO| |
| L | giveMeWelfarePlease | External | | " |NO| |
| L | getTotalDistributed | External | | |NO| |
| L | getUserInfo | External | | |NO| |
| L | getUserRealizedRewards | External | | |NO| |

```



```

| L | getPendingRewards | External | | | NO |
| L | initialize | External | | " | NO |
| L | getCurrentReward | External | | | NO |
|||||
| **SOL** | Implementation | SafeMath | |||
| L | <Constructor> | Public | | # | NO |
| L | transferOwner | External | | " | onlyOwner |
| L | renounceOwnership | External | | " | NO |
| L | setOperator | Public | | " | NO |
| L | renounceOriginalDeployer | External | | " | NO |
| L | <Receive Ether> | External | | # | NO |
| L | totalSupply | External | | | NO |
| L | decimals | External | | | NO |
| L | symbol | External | | | NO |
| L | name | External | | | NO |
| L | getOwner | External | | ! | NO |
| L | balanceOf | Public | | ! | NO |
| L | allowance | External | | ! | NO |
| L | approve | External | | " ! | NO |
| L | _approve | Internal | $ | " | |
| L | approveContractContingency | Public | | " ! | onlyOwner |
| L | transfer | External | | " | NO |
| L | transferFrom | External | | " | NO |
| L | setNewRouter | External | | " | onlyOwner |
| L | setLpPair | External | | " | onlyOwner |
| L | setInitializers | External | | " | onlyOwner |
| L | isExcludedFromFees | External | | | NO |
| L | isExcludedFromDividends | External | | | NO |
| L | isExcludedFromProtection | External | | | NO |
| L | setDividendExcluded | Public | | " | onlyOwner |
| L | setExcludedFromFees | Public | | " | onlyOwner |

```



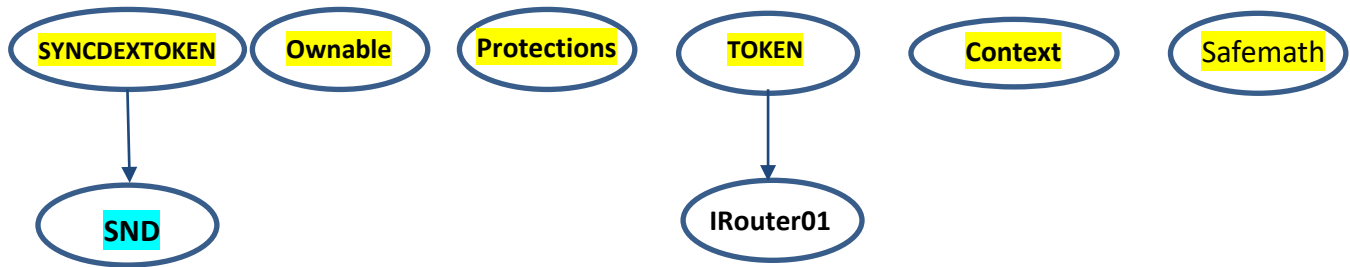
# AUDIT SCOPE

## SYNCDEx TOKEN

| ID  | Repo                       | Comment | File                  | SHM211 Checksum   |
|-----|----------------------------|---------|-----------------------|---|
| FTM | contracts/Syncdextoken.sol | cC51D89 | Strings.sol           | 85f15802c6be0fd50f8632d8433cccc9d<br>b6f4b39f9e566d1fa78de54b84baddr54  |
| FRY | contracts/Syncdextoken.sol | cC51D33 | MetadataHelper.sol    | 8oippkjjk96be0fd50f8632d8433cccc9<br>db6f4b39f9e566d1yhhg8765ffckiuybb  |
| FTV | contracts/Syncdextoken.sol | cC51D09 | ERC20WithPermit.sol   | 3666778uj908766362fvyga98jdkl8864<br>8yhfbqt37409owehbgwhuyyyg223738    |
| FML | contracts/Syncdextoken.sol | cC51D21 | ReentrancyGuard.sol   | 98uuyriy399787390uhbiiuhghhdg7guu<br>30oi7799u9359ydfgdgygeigi3ioueyy78 |
| FTR | contracts/Syncdextoken.sol | cC51D05 | ECDSA.sol             | 4566efgywqutfeuh87872t1537883798<br>3639293763hhegetgjfwjk89336668862   |
| FOP | contracts/Syncdextoken.sol | cC51D42 | IUniswapV2Callee.sol  | 546363ttebnve88329973mvvdsggct47<br>8153ytdgfdxy792635fgdji1900990908   |
| FDP | contracts/Syncdextoken.sol | cC51D25 | IERC20Permit.sol      | 835656990327hudbinnjnr6729dchjld0<br>993ytyy3vq63235727879889073        |
| FWY | contracts/Syncdextoken.sol | cC51D90 | SNDPair.so            | cc089692343d1cc36eaf196046d7a528<br>d153abd55ba20e82f1d57c22fcd92675    |
| FKB | contracts/Syncdextoken.sol | cC51D32 | Math.sol              | 8448b3af42497f5f74e53424ee3e6c55<br>1f51356945108d22a893d608a7990542    |
| FXY | contracts/Syncdextoken.sol | cC51D23 | IUniswapV2Factory.sol | 5c86aa1dd3889db5fcd17a80214b226f<br>c784f268ab9db82df97c1d2459467831    |
| FCB | contracts/Syncdextoken.sol | cC51D63 | IZFPair.sol           | b8244da33db171e5533d77bef4a3570<br>3df1de2cebea5f35cb38ce6a26c778cf1    |
| FWO | contracts/Syncdextoken.sol | cC51D60 | IWETH.sol             | 3d408b8f2cc56f9699a402b5151de906<br>71de089c3007afc9e4fc867c04152e7c    |
| FGT | contracts/Syncdextoken.sol | cC51D54 | IERC20Permit.sol      | 9d751621c3501102e4b50005ca3314ec<br>6e04e6ff8bbb30852d1c7edfff3f8cef    |
| FDF | contracts/Syncdextoken.sol | cC51D78 | ZFRouterInternal.sol  | 455687gfsadjknlpuihhg774580vgfxx<br>ki9876dhgvb990lkjhde444566788       |
| FTY | contracts/Syncdextoken.sol | cC51D94 | IERC20Metadata.sol    | 566HFFertyuijdsfggtyyykhgdrst<br>gioprdeyuiiyyt446789ysghn              |



## INHERITANCE GRAPH



| Identifier | Definition                                 | Severity   |
|------------|--|------------|
| CEN-12     | Centralization privileges of SYNCDEX TOKEN | Medium # 🟡 |

Vulnerability 0 : No important security issue detected.

Threat level: Low

```

458  * with `errorMessage` as a fallback revert reason when `target` reverts.
459  *
460  * _Available since v3.1._
461  */
462  function functionCallWithValue(
463      address target,
464      bytes memory data,
465      uint256 value,
466      string memory errorMessage
467  ) internal returns (bytes memory) {
468  ....require(address(this).balance >= value, 'Address: insufficient balance for call');
469  ....return _functionCallWithValue(target, data, value, errorMessage);
470  ....}
471
472  function _functionCallWithValue(
473      address target,
474      bytes memory data,
475      uint256 weiValue,
476      string memory errorMessage
477  ) private returns (bytes memory) {
478      require(isContract(target), 'Address: call to non-contract');
479
480      // solhint-disable-next-line avoid-low-level-calls
481      (bool success, bytes memory returndata) = target.call{value: weiValue}(data);
482      if (success) {
483          return returndata;
  
```

## FTV-01 POSSIBLE OVERFLOW

| Category                       | Severity ● | Location                   | Status       |
|--------------------------------|------------|----------------------------|--------------|
| Status Mathematical Operations | Minor      | contracts/Syncdextoken.sol | Acknowledged |

### Description

In `updateFormapping`, the following equation is used inside an unchecked block

```
contract ERC20 is Context, IERC20, Ownable {
    uint256 private constant _preMineSupply = 5800000 * 1e18;
    uint256 private constant _maxSupply = 10000000 * 1e18;

    using SafeMath for uint256;
    using Address for address;

    mapping(address => uint256) private _balances;

    mapping(address => mapping(address => uint256)) private _allowances;
```

Where parameters. `mapping` less Used is a this and less ride In is a this.  
As these two are multiplied together in an unchecked block, they may overflow.

### Recommendation

We recommend either checking for overflow in this case, or ensuring that the `PairsIn` is close enough it will never cause an overflow



## FZT-02 POSSIBLE OVERFLOW

| Category                       | Severity ● | Location                   | Status       |
|--------------------------------|------------|----------------------------|--------------|
| Status Mathematical Operations | Minor      | contracts/Syncdextoken.sol | Acknowledged |

### Description

In `updateForMinter`, the following equation is used inside an unchecked block

```
) internal {
    require(sender != address(0), 'ERC20: transfer from the zero address');
    require(recipient != address(0), 'ERC20: transfer to the zero address');

    _balances[sender] = _balances[sender].sub(amount, 'ERC20: transfer amount exceeds balance');
    _balances[recipient] = _balances[recipient].add(amount);
    emit Transfer(sender, recipient, amount);
}
```

Minter can not issue more `SYNCDEx TOKEN` tokens indefinitely.

Note that as of the date of publishing, the above review reflects the current understanding of known security patterns as they relate to the `SYNCDEx TOKEN` contract.

### Recommendation

We recommend either checking for overflow in this case, or ensuring that the `PairsIn` is close enough it will never cause an overflow.



## FTZ-03 POSSIBLE OVERFLOW

| Category                       | Severity ● | Location                        | Status       |
|--------------------------------|------------|---------------------------------|--------------|
| Status Mathematical Operations | Minor      | contracts/code/Syncdextoken.sol | Acknowledged |

### Description

There seems to be no way to disable a particular transfer fee by setting spacing to 0. Probably not an issue.

```
function transferOwnership(address newOwner) public virtual onlyOwner {
    require(
        newOwner != address(0),
        "Ownable: new owner is the zero address"
    );
    emit OwnershipTransferred(_owner, newOwner);
    _owner = newOwner;
}
```

### Recommendation

This should be a named constant being equal to 1e6, which occurs in other contracts.



## FTZ-04 POSSIBLE OVERFLOW

| Category     | Severity <span>●</span> | Location                        | Status       |
|--------------|-------------------------|---------------------------------|--------------|
| Bad datatype | Minor                   | contracts/code/Syncdextoken.sol | Acknowledged |

### Description

The dev Sets `amount` as the allowance of `spender` over the `owner`'s tokens.

This is internal function is equivalent to `approve`, and can be used to e.g. set automatic allowances for certain subsystems,

#### Requirements:

- `owner` cannot be the zero address.
- `spender` cannot be the zero address.

```
function _approve(  
    address owner,  
    address spender,  
    uint256 amount  
) internal {  
    require(owner != address(0), 'ERC20: approve from the zero address');  
    require(spender != address(0), 'ERC20: approve to the zero address');  
  
    _allowances[owner][spender] = amount;  
    emit Approval(owner, spender, amount);  
}
```

### Recommendation

It is advised to Emits an {Approval} event.

## FTZ-05 POSSIBLE OVERFLOW

| Category   | Severity <span>●</span> | Location                  | Status        |
|------------|-------------------------|---------------------------|---------------|
| Bad naming | Minor                   | Contract/Syncdextoken.sol | INFORMATIONAL |

### Description

State **Token Supply** variables can be declared as constant using the constant keyword. This means that the value of the state variable cannot be changed after it has been set. Additionally, the constant variables decrease gas consumption of the corresponding transaction.

```
contract ERC20 is Context, IERC20, Ownable {
    uint256 private constant _preMineSupply = 5800000 * 1e18;
    uint256 private constant _maxSupply = 10000000 * 1e18;

    using SafeMath for uint256;
    using Address for address;
```

### Recommendation

Constant **Token** variables can be useful when the contract wants to ensure that the **value** of a state Token variable cannot be changed by any function in the contract. This can be useful for storing values that are important to the contract's behavior, such as the contract's address or the maximum number of times a certain function can be called. The team is advised to add the constant keyword to state variables that never change.

## Vulnerability Scan

### REENTRANCY

Severity

Major

Confidence Parameter

Certain

## Vulnerability Description

**NOTE:** In a re-entrance attack, a malicious contract calls back into the calling contract before the first invocation of the function is finished. This may cause the different invocations of the function to interact in undesirable ways, especially in cases where the function is updating state variables after the external calls.

## Scanning Line:

```
function _transfer(  
    address sender,  
    address recipient,  
    uint256 amount  
) internal {  
    require(sender != address(0), 'ERC20:  
transfer from the zero address');  
    require(recipient != address(0), 'ERC20:  
transfer to the zero address');  
  
    _balances[sender] =  
_balances[sender].sub(amount, 'ERC20: transfer  
amount exceeds balance');  
    _balances[recipient] =  
_balances[recipient].add(amount);  
    emit Transfer(sender, recipient, amount);  
}
```



## Vulnerability Run check

### Contract Info

Total supply

10000000

Transaction Tax

Buy 0.00 % / Sell 0.00 %

### Risk Analysis

✔ Contract source code verified

This token contract is open source. You can check the contract code for details. Unsourced token contracts are likely to have malicious functions to defraud their users of their assets.

✔ No Proxy

There is no proxy in the contract. The proxy contract means contract owner can modify the function of the token and possibly effect the price.

✔ No mint function

Mint function is transparent or non-existent. Hidden mint functions may increase the amount of tokens in circulation and effect the price of the token.

✔ No function to retrieve ownership

If this function exists, it is possible for the project owner to regain ownership even after relinquishing it.

✔ Owner cant change balance

The contract owner does not have the authority to modify the balance of tokens at other addresses.



### Honeypot Risk

✔ This does not appear to be a honeypot

We are not aware of any code that prevents the sale of tokens.

✔ No trading cooldown

The token contract has no trading cooldown function. If there is a trading cooldown function, the user will not be able to sell the token within a certain time or block after buying.

✔ No Anti Whale

There is no limit to the number of token transactions. The number of scam token transactions may be limited (honeypot risk).

✔ No blacklist function

No blacklist function is included.

✔ No whitelist function

Whitelist function found



## General Detectors



### Incorrect Solidity Version

This contract uses an unconventional or old version of Solidity.



Attention  
Required



### Public Functions Should be Declared External

Some functions in this contract should be declared as external in order to save gas.



Attention  
Required



### State Variables Should be Declared Constant

Some state variables in this contract should be declared as constant








Attention  
Required

- |   |  |
|---|--|
| ✓ No vulnerable withdrawal functions found                  | ✓ No dumping risks found                       |
| ✓ No reentrancy risk found                                  | ✓ No compiler version inconsistencies found    |
| ✓ No locks detected   | ✓ No unchecked call responses found            |
| ✓ Verified source code found                                | ✓ No vulnerable self-destruct functions found  |
| ✓ No mintable risks found                                   | ✓ No assertion vulnerabilities found           |
| ✓ Users can always transfer their tokens                    | ✓ No old solidity code found                   |
| ✓ Contract cannot be upgraded                               | ✓ No external delegated calls found            |
| ✓ Wallets cannot be blacklisted from transferring the token | ✓ No external call dependency found            |
| ✓ No transfer fees found                                    | ✓ No vulnerable authentication calls found     |
| ✓ Token can be sold through regular AMMs                    | ✓ No invalid character typos found             |
| ✓ No transfer limits found                                  | ✓ No RTL characters found                      |
| ✓ No ERC20 approval vulnerability found                     | ✓ No dead code found                           |
| ✓ Contract owner cannot abuse ERC20 approvals               | ✓ No risky data allocation found               |
| ✓ No ERC20 interface errors found                           | ✓ No uninitialized state variables found       |
| ✓ No blocking loops found                                   | ✓ No uninitialized storage variables found     |
| ✓ No centralized balance controls found                     | ✓ No vulnerable initialization functions found |
| ✓ No transfer cooldown times found                          | ✓ No risky data handling found                 |
| ✓ No approval restrictions found                            | ✓ No number accuracy bug found                 |
| ✓ No external calls detected                                | ✓ No out-of-range number vulnerability found   |



## OPTIMIZATIONS | SYNCDEX TOKEN

| ID       | Title                             | Category         | Status   |
|----------|-----------------------------------|------------------|--|
| GZT- 007 | Logarithm Refinement Optimization | Gas Optimization | Acknowledged  |
| GZT- 323 | Checks Can Be Performed Earlier   | Gas Optimization | Acknowledged  |
| GZT- 679 | Unnecessary Use Of SafeMath       | Gas Optimization | Acknowledged  |
| GZT- 122 | Struct Optimization               | Gas Optimization | Acknowledged  |
| GZT-067  | Unused State Variable             | Gas Optimization | Acknowledged  |



## Repository:

<https://github.com/SYNCDEXTOKEN/>

## All Audited Files

**SYNCDEXTOKEN.SOL**

| TYPE    | DATA   |
|---------|--|
| bytes32 | 0x00 |
| bytes32 | 0x010002fdf0a356d44aaede784f9641ccadba58e262dc83e92ab3d96c47973af7 |
| bytes   | 0x   |

## Contracts:

**Contract:**

**0xe12575BAdf4F3cf659F4D2e6A68D6A671523EAf2**

## Transaction Hash:

**0x5e16f06557733bc973bb98ec2d2110956078c258bafc1a43481652ef42378ac1**



## MANUAL REVIEW

**SYNCDEx:** is the first ecosystem-focused and community-driven DEX built on zkSync Era. Our custom-built protocol is designed to provide unmatched efficiency and flexibility. Whether you're a builder or user, our infrastructure enables you to tap into deep and sustainable pools of liquidity that can be easily adapted to your needs.

**Project:** SYNCDEx TOKEN

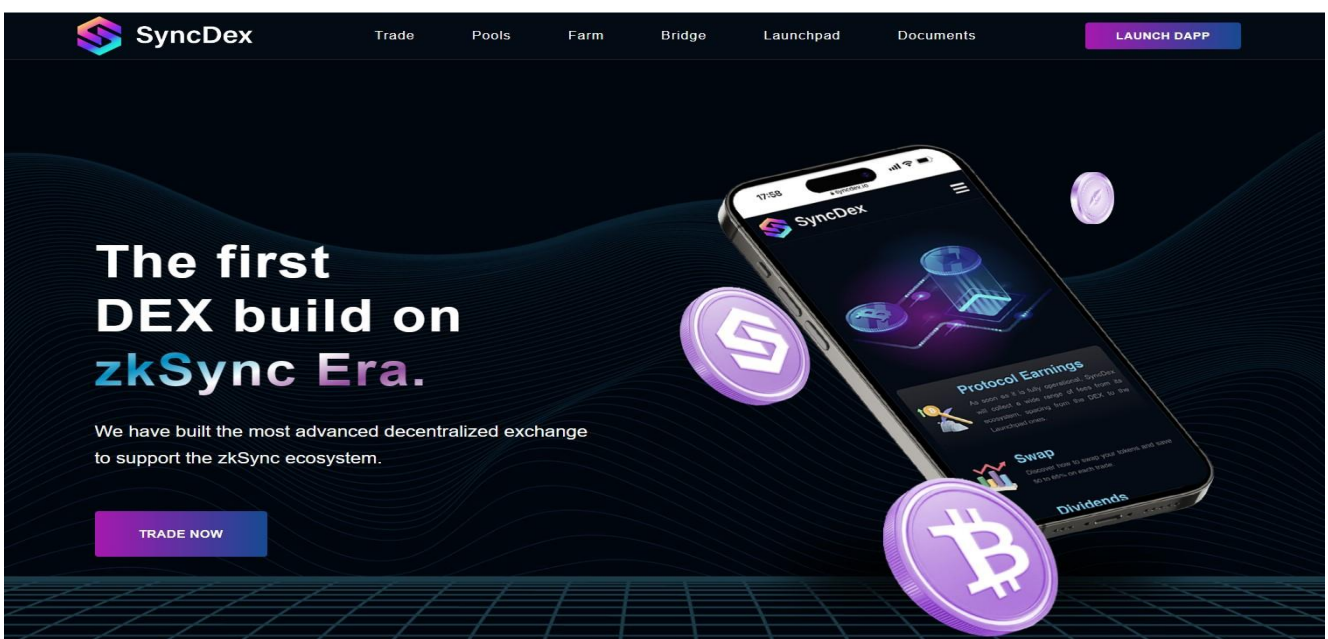
**Chain/Standard:** ZKSYNC NETWORK

**Max supply:** 10,000,000

**Ticker:** SND



### Outstanding Features of SYNCDEx TOKEN Is Launching On ZKSYNC ERA Network





# ISSUES CHECKING STATUS

Issue Description

Checking Status

|     |   |        |
|-----|---|--------|
| 1.  | Compiler errors.  | PASSED |
| 2.  | Race Conditions and reentrancy. Cross-Function Race Conditions. | PASSED |
| 3.  | Possible Delay In Data Delivery.                                | PASSED |
| 4.  | Oracle calls.   | PASSED |
| 5.  | Front Running.  | PASSED |
| 6.  | Sol Dependency.   | PASSED |
| 7.  | Integer Overflow And Underflow.                                 | PASSED |
| 8.  | DoS with Revert.  | PASSED |
| 9.  | Dos With Block Gas Limit.                                       | PASSED |
| 10. | Methods execution permissions.                                  | PASSED |
| 11. | Economy Model of the contract.                                  | PASSED |
| 12. | The Impact Of Exchange Rate On the solidity Logic.              | PASSED |
| 13. | Private use data leaks.   | PASSED |
| 14. | Malicious Event log.  | PASSED |
| 15. | Scoping and Declarations.                                       | PASSED |
| 16. | Uninitialized storage pointers.                                 | PASSED |
| 17. | Arithmetic accuracy.  | PASSED |
| 18. | Design Logic.   | PASSED |
| 19. | Cross-Function race Conditions                                  | PASSED |
| 20. | Save Upon solidity contract Implementation and Usage.           | PASSED |
| 21. | Fallback Function Security                                      | PASSED |



**AUDIT RESULT**

**PASSED**

SMART CONTRACT AUDIT OF SYNCDEX TOKEN

| Identifier | Definition              | Severity  |
|------------|-------------------------|---|
| TEN-02     | Transfers User's Tokens | Minor  |

```
constructor() internal {  
    address msgSender = _msgSender();  
    _owner = msgSender;  
    emit OwnershipTransferred(address(0), msgSender);  
}
```

## Alleviation:

Any user has the authority to transfer the balance of a user's address if the user has granted allowance. The contract does not subtract the allowance in the transferFrom() method, as a result, the transfer can be repeated until the user's balance go to zero...

## CERTIFICATE BY VITAL BLOCK SECURITY



## RECOMMENDATION

Deployer and/or contract owner private keys are secured carefully.

Please refer to PAGE-09 **CENTRALIZED PRIVILEGES** for a detailed understanding.

## ALLEVIATION

**SYNCDEx TOKEN** project team understands the centralization risk. Some functions are provided privileged access to ensure a good runtime behaviour in the project



| Identifier | Definition               | Severity  |
|------------|--------------------------|---|
| TDB-12     | Third Party Dependencies | Minor  |

A smart contract is interacting with third-party protocols e.g., Uniswap, Pancakeswap router, cashier contract,

And protections contract. The scope of the audit treats third-party entities as black boxes and assumes their functional correctness. However, in the real world, third parties can be compromised and exploited. Moreover, upgrades in third parties can create severe impacts, e.g., increased transactional fees, deprecation of previous routers, etc.

## RECOMMENDATION

Inspect and validate third party dependencies regularly, and mitigate severe impacts whenever necessary.



## DISCLAIMERS

Vital Block Security provides the easy-to-understand audit of Solidity, Move, and Raw source codes (commonly known as smart contracts).

The smart contract for this particular audit was analyzed for common contract vulnerabilities, and centralization exploits. This audit report makes no statements or warranties on the security of the code. This audit report does not provide any warranty or guarantee regarding the absolute bug-free nature of the smart contract analyzed, nor do they provide any indication of the client's business, business model, or legal compliance. This audit report does not extend to the compiler layer, any other areas beyond the programming language, or other programming aspects that could present security risks. Cryptographic tokens are emergent technologies, they carry high levels of technical risks and uncertainty. You agree that your access and/or use, including but not limited to any services, reports, and materials, will be at your sole risk on an as-is, where-is, and as-available basis. This audit report could include false positives, false negatives, and other unpredictable results.

## CONFIDENTIALITY

This report is subject to the terms and conditions (including without limitations, description of services, confidentiality, disclaimer and limitation of liability) outlined in the scope of the audit provided to the client. This report should not be transmitted, disclosed, referred to, or relied upon by any individual for any purpose without InterFi Network's prior written consent.

## NO FINANCIAL ADVICE

This audit report does not indicate the endorsement of any particular project or team, nor guarantees its security. No third party should rely on the reports in any way, including to make any decisions to buy or sell a product, service or any other asset. The information provided in this report does not constitute investment advice, financial advice, trading advice, or any other sort of advice and you should not treat any of the report's content as such. This audit report should not be used in any way





to make decisions around investment or involvement. This report in no way provides investment advice, nor should be leveraged as investment advice of any sort.

**FOR AVOIDANCE OF DOUBT, SERVICES, INCLUDING ANY ASSOCIATED AUDIT REPORTS OR MATERIALS, SHALL NOT BE CONSIDERED OR RELIED UPON AS ANY FORM OF FINANCIAL, TAX, LEGAL, REGULATORY, OR OTHER ADVICE.**

### **TECHNICAL DISCLAIMER**

**ALL SERVICES, AUDIT REPORTS, SMART CONTRACT AUDITS, OTHER MATERIALS, OR ANY PRODUCTS OR RESULTS OF THE USE THEREOF ARE PROVIDED “AS IS” AND “AS AVAILABLE” AND WITH ALL FAULTS AND DEFECTS WITHOUT WARRANTY OF ANY KIND. TO THE MAXIMUM EXTENT PERMITTED UNDER APPLICABLE LAW, VITAL BLOCK HEREBY DISCLAIMS ALL WARRANTIES, WHETHER EXPRESSED, IMPLIED, STATUTORY, OR OTHERWISE WITH RESPECT TO SERVICES, AUDIT REPORT, OR OTHER MATERIALS. WITHOUT LIMITING THE FOREGOING, VITAL BLOCK SPECIFICALLY DISCLAIMS ALL IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT, AND ALL WARRANTIES ARISING FROM THE COURSE OF DEALING, USAGE, OR TRADE PRACTICE.**

**WITHOUT LIMITING THE FOREGOING, VITAL BLOCK MAKES NO WARRANTY OF ANY KIND THAT ALL SERVICES, AUDIT REPORTS, SMART CONTRACT AUDITS, OR OTHER MATERIALS, OR ANY PRODUCTS OR RESULTS OF THE USE THEREOF, WILL MEET THE CLIENT’S OR ANY OTHER INDIVIDUAL’S REQUIREMENTS, ACHIEVE ANY INTENDED RESULT, BE COMPATIBLE OR WORK WITH ANY SOFTWARE, SYSTEM, OR OTHER SERVICES, OR BE SECURE, ACCURATE, COMPLETE, FREE OF HARMFUL CODE, OR ERROR-FREE.**

### **TIMELINESS OF CONTENT**

**The content contained in this audit report is subject to change without any prior notice. Vital Block does not guarantee or warrant the accuracy, timeliness, or completeness of any report you access using the internet or other means, and assumes no obligation to update any information following the publication.**



## LINKS TO OTHER WEBSITES

**This audit report provides, through hypertext or other computer links, access to websites and social accounts operated by individuals other than Vital Block. Such hyperlinks are provided for your reference and convenience only and are the exclusive responsibility of such websites and social accounts owners. You agree that Vital block Security is not responsible for the content or operation of such websites and social accounts and that Vital Block shall have no liability to you or any other person or entity for the use of third-party websites and social accounts. You are solely responsible for determining the extent to which you may use any content at any other websites and social accounts to which you link from the report.**



## ABOUT VITAL BLOCK

**Vital Block provides intelligent blockchain Security Solutions. We provide solidity and Raw Code Review, testing, and auditing services. We have Partnered with 15+ Crypto Launchpads, audited 50+ smart contracts, and analyzed 200,000+ code lines. We have worked on major public blockchains e.g., Ethereum, Binance, Cronos, Doge, Polygon, Avalanche, Metis, Fantom, Bitcoin Cash, Aptos, Oasis, etc.**

**Vital Block is Dedicated to Making Defi & Web3 A Safer Place. We are Powered by Security engineers, developers, UI experts, and blockchain enthusiasts. Our team currently consists of 5 core members, and 4+ casual contributors.**

**Website:** <https://www.Vitalblock.org>

**Email:** [info@vitalblock.org](mailto:info@vitalblock.org)

**GitHub:** <https://github.com/vital-block>

**Telegram (Engineering):** [https://t.me/vital\\_block](https://t.me/vital_block)

**Telegram (Onboarding):** [https://t.me/vitalblock\\_cmo](https://t.me/vitalblock_cmo)





**vital-block**



**info@vitalblock.org**



**www.Vitalblock.org**



Vital Block Dedicated to securing Public and Private Blockchain Ecosystem