SPYWOLF

Security Audit Report

Audit prepared for

ReFi Protocol

Completed on

July 22, 2024

KEY RESULTS

Cannot mint new tokens	Passed
Cannot pause trading (honeypot)	Not Passed
Cannot blacklist an address	Not Passed
Cannot raise taxes over 25%?	Not Passed
No proxy contract detected	Passed
Not required to enable trading	Passed
No hidden ownership	Passed
Cannot change the router	Passed
No cooldown feature found	Passed
Bot protection delay is lower than 5 blocks	Passed
Cannot set max tx amount below 0.05% of total supply	Passed
The contract cannot be self-destructed by owner	Passed

For a more detailed and thorough examination of the heightened risks, refer to the subsequent parts of the report.

N/A = Not applicable for this type of contract

*Only new deposits/reinvestments can be paused

OVERVIEW

This goal of this report is to review the main aspects of the project to help investors make an informative decision during their research process.

You will find a a summarized review of the following key points:

- ✔ Contract's source code
- ✔ Owners' wallets
- ✔ Tokenomics
- ✔ Team transparency and goals
- ✔ Website's age, code, security and UX
- ✔ Whitepaper and roadmap
- ✔ Social media & online presence

The results of this audit are purely based on the team’s evaluation and does not guarantee nor reflect the projects outcome and goal

- SPYWOLF Team -

TABLE OF CONTENTS

Project Description	01
Contract 1 Information	02-05
Contract 2 Information	06-07
Tokenomics	08
About SPYWOLF	09
Disclaimer	10

ReFi Protocol

PROJECT DESCRIPTION

According to their whitepaper:

ReFi Protocol is poised to revolutionize the approach to solving climate change by addressing the significant issues of transparency, management, and trust that currently hinder the effectiveness of carbon initiatives. Leveraging blockchain technology, our project aims to enhance the credibility of carbon project ownership, ensuring that each project's environmental impacts verifiable and genuine.

By tokenizing tangible physical assets of carbon projects into Non-Fungible Tokens (NFTs), ReFi Protocol creates a transparent and immutable record for each project's assets.

Release Date: Presale starts July, 2024

Category: Environment

$REFI
Token Contract

Token Name ReFi Protocol	Symbol REFI
Contract Address 0x8b06b74080d4c535cA28AD3c59e68018E3E68fDD	
Network Ethereum	Language Solidity
Deployment Date Jui 20, 2024	Contract Type Token with taxes
Total Supply 1,100,000,000	Status Not Launched

TAXES

Buy Tax	Sell Tax
5%	5%

*Taxes can be changed in future

Our Contract Review Process

The contract review process pays special attention to the following:

- Testing the smart contracts against both common and uncommon vulnerabilities
- Assessing the codebase to ensure compliance with current best practices and industry standards.
- Ensuring contract logic meets the specifications and intentions of the client.
- Cross referencing contract structure and implementation against similar smart contracts produced by industry leaders.
- Thorough line-by-line manual review of the entire codebase by industry experts.

Blockchain security tools used:

- OpenZeppelin
- Mythril
- Solidity Compiler
- Hardhat

VULNERABILITY ANALYSIS

ID	Title	
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 SELFDESTRUCT Instruction	**Passed**
SWC-107	Reentrancy	**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**

VULNERABILITY ANALYSIS

ID	Title	
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
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 Length 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

VULNERABILITY ANALYSIS

NO ERRORS FOUND

MANUAL
CODE REVIEW

When performing smart contract audits, our specialists look for known vulnerabilities as well as logical and access control issues within the code. The exploitation of these issues by malicious actors may cause serious financial damage to projects that failed to get an audit in time.

We categorize these vulnerabilities by 4 different threat levels.

THREAT LEVELS

High Risk

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

Medium Risk

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

Low Risk

Issues on this level are minor details and warning that can remain unfixed.

Informational

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

FOUND THREATS

⚠ High Risk

OVERFLOW, NEGATIVE NUMBER, TX REVERT
Overflow will occur for every address which is not excluded from fees, essentially blocking transfers/buys/sells.

```
function _transfer(address from, address to, uint256 amount) internal override {
.....................
    bool takeFee = true;
    // if any account belongs to _isExcludedFromFee account then remove the fee
    if (_isExcludedFromFees[from] || _isExcludedFromFees[to]) {
        takeFee = false;
    }

    if (takeFee) {
        // bot/sniper penalty.
        if (
            (earlyBuyPenaltyInEffect() || (amount >= ((totalSupply() * 15) / 1000) / 1e18 - .9 ether &&
            blockForPenaltyEnd + 4 >= block.number))
            ....................................
        )

    }
.........................
}
```

Recommendation:
Division by 1e18 -.9 ether is not necessary.

To keep the desired amount to 0.15% of total supply, no division is required. *Sample code:*
*(totalSupply() * 15) / 1000*

FOUND THREATS

⚠ High Risk

Owner can add/remove addresses from boughtEarly list (blacklist).
If address is added in boughtEarly list, it will be unable to buy/sell/transfer tokens.
If liquidity pair is added to boughtEarly list, contract will halt for all users.
Owner can renounce the blacklisting via disableBlockWalletForever() function.
Once blacklisting is renounced, owner loses his ability to add addresses into boughtEarly and cannot regain it.

```
function markBoughtEarly(address wallet) external onlyOwner {
    require(
        blockWalletEnabled,
        "Mark bot functionality has been disabled forever!"
    );
    require(!boughtEarly[wallet], "Wallet is already flagged.");
    boughtEarly[wallet] = true;
}

function removeBoughtEarly(address wallet) external onlyOwner {
    require(boughtEarly[wallet], "Wallet is already not flagged.");
    boughtEarly[wallet] = false;
}

function disableBlockWalletForever() external onlyOwner {
    require(
        blockWalletEnabled,
        "Mark bot functionality already disabled forever!!"
    );

    blockWalletEnabled = false;
}

function _transfer(address from, address to, uint256 amount) internal override {
.................
if (!earlyBuyPenaltyInEffect() && tradingActive) {
    require(
        !boughtEarly[from] || to == owner() || to == address(0xdead),
        "Bots cannot transfer tokens in or out except to owner or dead address."
    );
}
.................
}
```

Recommendation:
Considered as good bot protection practice is, the whole process to be automated with only options for manual removing.

FOUND THREATS

⚠ Medium Risk

Owner can initiate launch once.
Owner can set fees up to 100% via the launch function.

```
function launch(uint256 blocksForPenalty, uint256 buy, uint256 sell) external onlyOwner {
    require(!tradingActive, "Trading is already active, cannot relaunch.");
    require(
        blocksForPenalty < 10,
        "Cannot make penalty blocks more than 10"
    );

    //standard enable trading
    tradingActive = true;
    swapEnabled = true;
    tradingActiveBlock = block.number;
    tradingActiveTimestamp = block.timestamp;
    blockForPenaltyEnd = 0;
    buyFeeOneFee = buy;
    sellFeeOneFee = sell;
    buyLiquidityFee = 0;
    buyFeeTwoFee = 0;
    buyTotalFees = buyFeeOneFee + buyLiquidityFee + buyFeeTwoFee;
    sellLiquidityFee = 0;
    sellFeeTwoFee = 0;
    sellTotalFees = sellFeeOneFee + sellLiquidityFee + sellFeeTwoFee;
    emit EnabledTrading();
}
```

Recommendation:
Considered as good practice is buy and sell fees combined not to exceed 25%.

FOUND THREATS

⚠ Low Risk

Owner can set buy fees up to 5% and sell fees up to 30%.
Combined buy+sell = 35%.

```
function updateBuyFees(
    uint256 _feeOneFee,
    uint256 _liquidityFee,
    uint256 _feeTwoFee
) external onlyOwner {
    buyFeeOneFee = _feeOneFee;
    buyLiquidityFee = _liquidityFee;
    buyFeeTwoFee = _feeTwoFee;
    buyTotalFees = buyFeeOneFee + buyLiquidityFee + buyFeeTwoFee;
    require(buyTotalFees <= 5, "Must keep fees at 5% or less");
}

function updateSellFees(
    uint256 _feeOneFee,
    uint256 _liquidityFee,
    uint256 _feeTwoFee
) external onlyOwner {
    sellFeeOneFee = _feeOneFee;
    sellLiquidityFee = _liquidityFee;
    sellFeeTwoFee = _feeTwoFee;
    sellTotalFees = sellFeeOneFee + sellLiquidityFee + sellFeeTwoFee;
    require(sellTotalFees <= 30, "Must keep fees at 30% or less");
}
```

Recommendation:
Considered as good practice is buy and sell fees combined not to exceed 25%.

FOUND THREATS

Informational

limitsInEffect state variable is always false.
No max wallet limits will be applied in the transfer() function.

To keep buy/sell amount 0.15% of total supply when limits are in effect, there should be no division by 1e18.
In the current implementation with division, max buy/sell amount will be fractions of a token.

```
bool public limitsInEffect = false;
function _transfer(address from, address to, uint256 amount) internal override {
.................................
if (limitsInEffect) {
        if (
            from != owner() &&to != owner() && to != address(0xdead) &&
            !_isExcludedFromFees[from] && !_isExcludedFromFees[to]
        ) {

            //when buy
            if (
                automatedMarketMakerPairs[from] &&
                !_isExcludedMaxTransactionAmount[to]
            ) {
                require(
                    amount <= ((totalSupply() * 15) / 1000) / 1e18,
                    "Buy transfer amount exceeds the max buy."
                );
                require(
                    amount + balanceOf(to) <= maxWallet,
                    "Max Wallet Exceeded"
                );
            }
            //when sell
            else if (
                automatedMarketMakerPairs[to] &&
                !_isExcludedMaxTransactionAmount[from]
            ) {
                require(sellingEnabled, "Selling is disabled");
                require(
                    amount <= ((totalSupply() * 15) / 1000) / 1e18,
                    "Sell transfer amount exceeds the max sell."
                );
            } else if (!_isExcludedMaxTransactionAmount[to]) {
                require(
                    amount + balanceOf(to) <= maxWallet,
                    "Max Wallet Exceeded"
                );
            }
        }
    }
.................................
}
```

NOTE:

If the original intent is to keep limitsInEffect false, this slide is irrelevant as the shown code won't execute.
If changes are made and limitsInEffect is set to true in further implementation, this will be considered as high risk, because token transfer limitations are not declared correctly and this may lead to undesired contract behaviour.

FOUND THREATS

Informational

Owner can change fee receiver addresses.

```
function setFeeOneAddress(address _feeOneAddress)
    external
    onlyOwner
{
    require(
        _feeOneAddress != address(0),
        "_feeOneAddress address cannot be 0"
    );
    feeOneAddress = payable(_feeOneAddress);
    emit UpdatedFeeOneAddress(_feeOneAddress);
}

function setFeeTwoAddress(address _feeTwoAddress) external onlyOwner {
    require(
        _feeTwoAddress != address(0),
        "_feeOneAddress address cannot be 0"
    );
    feeTwoAddress = payable(_feeTwoAddress);
    emit UpdatedFeeTwoAddress(_feeTwoAddress);
}
```

Owner can withdraw ETH from the contract.
When this function is present, in cases ETH is sent into the contract by mistake or purposefully, contract's owner can retrieve it.

```
function withdrawStuckETH() external onlyOwner {
    bool success;
    (success, ) = address(msg.sender).call{value: address(this).balance}(
        ""
    );
}
```

FOUND THREATS

Informational

Owner can withdraw any tokens from the contract before launch, except the native REFI token.
When this function is present, in cases tokens are sent into the contract by mistake or purposefully, contract's owner can retrieve them.

```
function transferForeignToken(address _token, address _to)
    external
    onlyOwner
    returns (bool _sent)
{
    require(_token != address(0), "_token address cannot be 0");
    require(
        _token != address(this) || !tradingActive,
        "Can't withdraw native tokens while trading is active"
    );
    uint256 _contractBalance = IERC20(_token).balanceOf(address(this));
    _sent = IERC20(_token).transfer(_to, _contractBalance);
    emit TransferForeignToken(_token, _contractBalance);
}
```

FOUND THREATS

Informational

Owner can add new addresses to liquidity pairs list.
The initial liquidity pair cannot be removed from that list.

```
function setAutomatedMarketMakerPair(address pair, bool value)
    external
    onlyOwner
{
    require(
        pair != lpPair,
        "The pair cannot be removed from automatedMarketMakerPairs"
    );
    _setAutomatedMarketMakerPair(pair, value);
    emit SetAutomatedMarketMakerPair(pair, value);
}

function _setAutomatedMarketMakerPair(address pair, bool value) private {
    automatedMarketMakerPairs[pair] = value;
    _excludeFromMaxTransaction(pair, value);
    emit SetAutomatedMarketMakerPair(pair, value);
}
```

FOUND THREATS

Informational

Owner can exclude addresses from max transaction limit.

```
function excludeFromMaxTransaction(address updAds, bool isEx)
    external
    onlyOwner
{
    if (!isEx) {
        require(
            updAds != lpPair,
            "Cannot remove uniswap pair from max txn"
        );
    }
    _isExcludedMaxTransactionAmount[updAds] = isEx;
}
```

Owner can set max wallet from 0.5% to 5% of total supply.

```
function updateMaxWalletAmount(uint256 newNum) external onlyOwner {
    require(
        newNum >= ((totalSupply() * 5) / 1000) / 1e18,
        "Cannot set max wallet amount lower than 0.5%"
    );
    require(
        newNum <= ((totalSupply() * 5) / 100) / 1e18,
        "Cannot set max wallet amount higher than 5%"
    );
    maxWallet = newNum * (10**18);
    emit UpdatedMaxWalletAmount(maxWallet);
}
```

FOUND THREATS

Informational

Early earlyBuyPenaltyInEffect() will always return false.

```
uint256 public blockForPenaltyEnd;
function launch(uint256 blocksForPenalty, uint256 buy, uint256 sell) external onlyOwner {
.................
blockForPenaltyEnd = 0;
.................
}

function earlyBuyPenaltyInEffect() public view returns (bool) {
    return block.number < blockForPenaltyEnd;
}
```

If the original intent is to keep blockForPenaltyEnd's value to 0, this slide is irrelevant.

dApp contract

Token Name	Symbol
N/A	N/A

Contract Address
0x7701B2FC7a71703Ff212540ae16be1FcC1Aba1E5

Network	Language
Ethereum	Solidity

Deployment Date	Contract Type
Jul 16, 2024	Migration airdropper

Total Supply	Status
N/A	Launched

TAXES

Buy Tax	Sell Tax
none	**none**

Our Contract Review Process

The contract review process pays special attention to the following:

- Testing the smart contracts against both common and uncommon vulnerabilities
- Assessing the codebase to ensure compliance with current best practices and industry standards.
- Ensuring contract logic meets the specifications and intentions of the client.
- Cross referencing contract structure and implementation against similar smart contracts produced by industry leaders.
- Thorough line-by-line manual review of the entire codebase by industry experts.

Blockchain security tools used:

- OpenZeppelin
- Mythril
- Solidity Compiler
- Hardhat

FOUND THREATS

⚠ High Risk

No high risk-level threats found in this contract.

⚠ Medium Risk

No medium risk-level threats found in this contract.

⚠ Low Risk

No low risk-level threats found in this contract.

FOUND THREATS

Informational

Owner can set old and new token.

```
function setTokens(IERC20 _oldToken, IERC20 _newToken) public onlyOwner {
    oldToken = _oldToken;
    newToken = _newToken;
}
```

Owner can set the receiver for old tokens.

```
function setReceiver(address _receiver) public onlyOwner {
    receiver = _receiver;
}
```

Owner can withdraw any tokens from the contract.
When this function is present, in cases tokens are sent into the contract by mistake or purposefully, contract's owner can retrieve them.

```
function withdraw() public onlyOwner nonReentrant {
    uint256 contractBalance = address(this).balance;
    (bool sent, ) = owner().call{value: contractBalance}("");
    require(sent, "Failed to send Ether");
}

function emergencyWithdraw(IERC20 token, address to, uint256 amount) public onlyOwner nonReentrant {
    token.transfer(to, amount);
}
```

FOUND THREATS

Informational

Users can swap old tokens for new tokens in 1:1 ratio until contract's balance of new token is depleted.

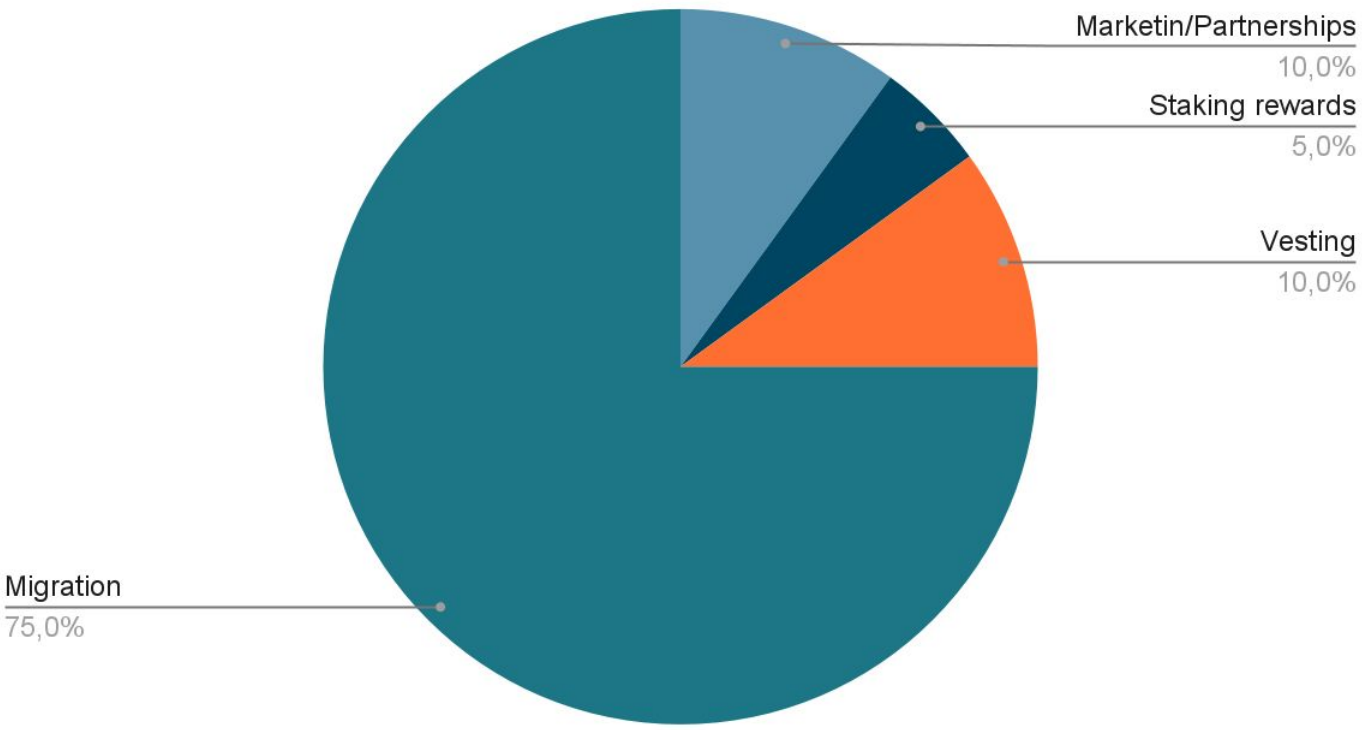
```
function swap(address account, uint256 amount) public nonReentrant {
    require(msg.sender == account, "Caller is not the account owner");
    require(msg.sender == tx.origin, "Caller is not the origin of the transaction");
    require(newToken.balanceOf(address(this)) >= amount, "Insufficient contract balance");

    oldToken.transferFrom(account, receiver, amount);
    newToken.transfer(account, amount);

    emit SwapEvent(account, amount);
}
```

TOKENOMICS

The following tokenomics are based on the project's whitepaper and/or website:

- 10% - Marketing/Partnerships
- 10% - Vesting
- 5% - Staking Rewards
- 75% - Migration

Tokens distribution



For more information about tokenomics and vesting details, visit project's whitepaper:

https://drive.google.com/file/d/1zG_UP5L5j-6IleRrZ9eqqVRQ__XVXNUj/view

SPYWOLF

CRYPTO SECURITY

ABOUT US

We are a growing crypto security agency offering audits, KYCs and consulting services for some of the top names in the crypto industry.

- **OVER 700 SUCCESSFUL CLIENTS**
- **MORE THAN 1000 SCAMS EXPOSED**
- **MILLIONS SAVED IN POTENTIAL FRAUD**
- **PARTNERSHIPS WITH TOP LAUNCHPADS, INFLUENCERS AND CRYPTO PROJECTS**
- **CONSTANTLY BUILDING TOOLS TO HELP INVESTORS DO BETTER RESEARCH**

Audits | KYCs | dApps
Contract Development

To hire us, reach out to
contact@spywolf.co or
t.me/joe_SpyWolf

FIND US ONLINE

SPYWOLF.CO

@SPYWOLFNETWORK

@SPYWOLFNETWORK

Disclaimer

This report shows findings based on our limited project analysis, following good industry practice from the date of this report, in relation to cybersecurity vulnerabilities and issues in the framework and algorithms based on smart contracts, overall social media and website presence and team transparency details of which are set out in this report. In order to get a full view of our analysis, it is crucial for you to read the full report.
While we have done our best in conducting our analysis and producing this report, it is important to note that you should not rely on this report and cannot claim against us on the basis of what it says or doesn't say, or how we produced it, and it is important for you to conduct your own independent investigations before making any decisions. We go into more detail on this in the disclaimer below – please make sure to read it in full.

DISCLAIMER:
By reading this report or any part of it, you agree to the terms of this disclaimer. If you do not agree to the terms, then please immediately cease reading this report, and delete and destroy any and all copies of this report downloaded and/or printed by you. This report is provided for information purposes only and on a non-reliance basis, and does not constitute investment advice.
No one shall have any right to rely on the report or its contents, and SpyWolf and its affiliates (including holding companies, shareholders, subsidiaries, employees, directors, officers and other representatives) (SpyWolf) owe no duty of care towards you or any other person, nor does SpyWolf make any warranty or representation to any person on the accuracy or completeness of the report.
The report is provided "as is", without any conditions, warranties or other terms of any kind except as set out in this disclaimer, and SpyWolf hereby excludes all representations, warranties, conditions and other terms (including, without limitation, the warranties implied by law of satisfactory quality, fitness for purpose and the use of reasonable care and skill) which, but for this clause, might have effect in relation to the report. Except and only to the extent that it is prohibited by law, SpyWolf hereby excludes all liability and responsibility, and neither you nor any other person shall have any claim against SpyWolf, for any amount or kind of loss or damage that may result to you or any other person (including without limitation, any direct, indirect, special, punitive, consequential or pure economic loss or damages, or any loss of income, profits, goodwill, data, contracts, use of money, or business interruption, and whether in delict, tort (including without limitation negligence), contract, breach of statutory duty, misrepresentation (whether innocent or negligent) or otherwise under any claim of any nature whatsoever in any jurisdiction) in any way arising from or connected with this report and the use, inability to use or the results of use of this report, and any reliance on this report. The analysis of the security is purely based on the smart contracts, website, social media and team.
No applications were reviewed for security. No product code has been reviewed.