SPYWOLF

Security Audit Report

TESTNET

Audit prepared for

Pulsar

Completed on

April 26, 2025

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

KEY RESULTS

Cannot mint new tokens	*
Cannot pause trading (honeypot)	PASSED
Cannot blacklist an address	PASSED
Cannot raise taxes over 25%?	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

*New tokens can be minted only the minting contract in exchange of dragonX/titanX tokens

PULSAR

PROJECT DESCRIPTION:

According to their whitepaper:

Pulsar's inception is driven by the need to address prevalent issues in the DeFi space, such as inflationary pressures and the lack of incentivization mechanisms that reward long-term holders. With its Perma-Bull tokenomic nature and unique relationship with TitanX, Pulsar sets out to provide a new paradigm for value generation and retention in the cryptocurrency world.

The token's utility is intricately tied to the TitanX ecosystem, where it serves as a testament to the power of community-driven initiatives and the potential of collaborative financial ecosystems. By enabling users to mint Pulsar through the burning of TitanX, the token not only honors its roots but also contributes to the controlled circulation supply of its progenitor token, fostering a symbiotic environment that benefits stakeholders across both platforms.

Release Date: TBA

Launchpad: Fairlaunch

Category: DeFi

CONTRACT 1 INFO

Token Name
Pulsar

Symbol
PULSAR

Contract Address
0xf72960Fb725C8188b8F11aA1b6141CcBcF3EBD67

Network
ETH Sepolia testnet

Language
Solidity

Deployment Date
Oct 02, 2024

Contract Type
Mintable token

Total Supply
4,347,815,476

Decimals
18

TAXES

Buy Tax
0%

Sell Tax
0%

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

SMART CONTRACT STATS

Calls Count	unavailable
External calls	unavailable
Internal calls	unavailable
Transactions count	unavailable
Last transaction time	unavailable
Deployment Date	unavailable
Create TX	unavailable
Owner	unavailable
Deployer	unavailable

TOKEN TRANSFERS STATS

Transfer Count	unavailable
Total Amount	unavailable
Median Transfer Amount	unavailable
Average Transfer Amount	unavailable
First transfer date	unavailable
Last transfer date	unavailable
Days token transferred	unavailable

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

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

When WETH amount is higher than 0, WETH spending (approval) should be granted before initiating addToBurnPool() function in BuyAndBurn contract.

```
function distribute() external {
    IERC20 titanx = IERC20(titanxAddress);
    address buyNBurn = _buyAndBurn();
    titanx.transfer(buyNBurn, toSendToBuyAndBurn);
    toSendToBuyAndBurn = 0;
    uint256 distributableTitanxBalance = titanx.balanceOf(address(this)) - _titanxToCross;
    uint256 halfTitanxBalance = (distributableTitanxBalance * 5_000) / 10_000;
    if (halfTitanxBalance > 0) {
      titanx.transfer(helios, halfTitanxBalance);
      titanx.transfer(dragonx, distributableTitanxBalance - halfTitanxBalance);
      // try IVault(dragonx).updateVault() {} catch (bytes memory b) {}
      dragonx.call(abi.encodeWithSelector(IVault.updateVault.selector));
    }
    // calculate fee amount and verify that it was sent
    BuyAndBurn(payable(buyNBurn)).addToBurnPool{value: address(this).balance}(IERC20(weth).balanceOf(address(this)));
}

BuyAndBurn Contract's code:

function addToBurnPool(uint256 wethAmount) external payable {
    (uint256 wethToBurn, uint256 wethFees) = _hoistWrapped(wethAmount);
    (uint256 nativeToBurn, uint256 nativeFees) = _hoistNative(msg.value);
    _incrementAccumulated(wethFees + nativeFees);
    emit AddToBurnPool(msg.sender, wethToBurn + nativeToBurn);
}

function _hoistWrapped(uint256 amount) internal returns (uint256 toBurn, uint256 fees) {
    if (amount > 0) {
      weth.transferFrom(msg.sender, address(this), amount);
      // we are assuming that weth is not a tax token
      fees = (amount / 4);
      weth.withdraw(fees);
      toBurn = amount - fees;
    }
}
```

FOUND THREATS

Informational

Users can mint Pulsar tokens given that they provide valid signature from current _priceOperator.
Total 50% of TitanX amount is minted as Pulsar.
25% Pulsar goes towards Pulsar's contract.
25% Pulsar goes towards user.

```
function mint(
    uint256 titanxAmount,
    uint256 priceDenominator,
    uint256 nonce,
    uint256 deadline,
    bytes calldata signature
) external payable {
    _verifyPriceInfo(priceDenominator, nonce, deadline, signature, _priceOperator);
    // check the fee first
    // old: 6 transfers
    // new: 0 transfers - deferred until buy and burn is desired
    _verifyFee(titanxAmount, priceDenominator);

    (uint256 toBurn, uint256 toDistribute, uint256 toBridge) = _divvy(titanxAmount);
    // 1 transfer TitanX to address(this)
    IERC20(titanxAddress).transferFrom(msg.sender, address(this), toDistribute + toBridge);
    // accumulating "to distribute" is held implicitly by the balance of the remaining titanx tokens
    // 1 burn
    _burnAndEarmarkTitanx(toBurn);
    // 2 mint
    _mintPulsar(titanxAmount);
    // 2 approvals + 2 bridges (transfer with a bunch of operations)
    // new: 2 bridges
    _titanxToCross += toBridge;
    emit Mint(msg.sender, titanxAmount, msg.value);
}

function _mintPulsar(uint256 titanxAmount) internal returns (uint256) {
    // mint 25% of titanX amount as pulsar to caller and this contract
    uint256 pulsarAmount = titanxAmount / 4;
    _mint(msg.sender, pulsarAmount);
    _mint(address(this), pulsarAmount);
    return pulsarAmount;
}
```

CONTRACT 2 INFO

Token Name
unavailable

Symbol
unavailable

Contract Address
0x3bD84fcdC6b9C28761968174aF0af744A05649d1

Network
ETH Seploia testnet

Language
Solidity

Deployment Date
Oct 02, 2024

Contract Type
Buy and burn interface

Total Supply
unavailable

Decimals
unavailable

TAXES

Buy Tax
0%

Sell Tax
0%

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 add new genesis address on the place of already existing one.

```
function updateGenesisAddress(address current, address next) external onlyOwner {
    if (_genesisMembers[next] != 0) {
      revert InvalidRecipient();
    }
    _genesisMembers[next] = _genesisMembers[current];
    _genesisMembers[current] = 0;
}
```

Owner can create Pulsar/WETH and Pulsar/TitanX Uniswap V3 pools once.

```
function init(address _pulsarAddress, uint160 initialPulsarWethSqrtPriceX96,
uint160 initialPulsarTitanxSqrtPriceX96) external onlyOwner {
    if (pulsarAddress != address(0)) {
      revert InitStateMismatch();
    }
    pulsarAddress = _pulsarAddress;
    _createPools(initialPulsarWethSqrtPriceX96, initialPulsarTitanxSqrtPriceX96);
}
```

Owner can set burn interval.
Current limit is 0 sec.

```
function setBurnInterval(uint256 newBurnInterval) external payable onlyOwner {
    burnInterval = newBurnInterval;
}
```

FOUND THREATS

Informational

Owner can set burn operator.

```
function setBurnOperator(address burnOperator_) external onlyOwner {
    _burnOperator = burnOperator_;
}
```

Owner can set WETH burn limit per interval.
Current limit is 1 WETH.

```
function setBurnLimitWeth(uint256 newBurnLimit) external payable onlyOwner {
    burnLimitWeth = newBurnLimit;
}
```

Owner can set reward percent up to 100%.

```
function setBurnCommitRewardRatio(uint256 newBurnCommitRewardRatio) external payable onlyOwner {
    burnCommitRewardRatio = newBurnCommitRewardRatio;
}

function _getAmounts() internal view returns (uint256 wethRemaining, uint256 titanxBalance, uint256 reward) {
    titanxBalance = titanx.balanceOf(address(this));
    uint256 wethBalance = weth.balanceOf(address(this));
    wethRemaining = wethBalance > burnLimitWeth ? burnLimitWeth : wethBalance;
    reward = (wethRemaining * burnCommitRewardRatio) / 10_000;
    reward = reward > wethRemaining ? wethRemaining : reward;
    wethRemaining -= reward;
    return (wethRemaining, titanxBalance, reward);
}
```

Owner can set slippage for tokens swaps.

```
function setSlippage(uint256 portionSqrtX96Lower, uint256 portionSqrtX96Upper) external payable onlyOwner {
    uniswapSlippageSqrtX96Lower = portionSqrtX96Lower;
    uniswapSlippageSqrtX96Upper = portionSqrtX96Upper;
}
```

FOUND THREATS

Informational

Users can burn Pulsar tokens and receive ETH reward, given that they provide valid signature from current _priceOperator.

```
function commitBurn(uint256 minBurn, uint256 nonce, uint256 deadline, bytes calldata signature) external payable {
    _verifyBurnInfo(minBurn, nonce, deadline, signature, _burnOperator);
    // burns can only happen at most once per interval
    if (block.timestamp < lastBurnTime + burnInterval) {
      revert BurnIntervalNotPassed();
    }

    (uint256 swapAmountWeth, uint256 swapAmountTitanx, uint256 reward) = _getAmounts();

    // pay the caller: unwrap WETH and send ETH to caller
    weth.withdraw(reward);
    payable(msg.sender).transfer(reward);

    // swap WETH for Pulsar if needed
    if (swapAmountWeth > 0) {
      _swapWethForPulsar(swapAmountWeth);
    }

    // swap TITANX for Pulsar if needed
    if (swapAmountTitanx > 0) {
      _swapTitanxForPulsar(swapAmountTitanx);
    }

    // burn whatever PULSAR is in this contract
    Pulsar pulsar = Pulsar(pulsarAddress);
    uint256 pulsarAmount = pulsar.balanceOf(address(this));
    if (pulsarAmount == 0) {
      return;
    }
    if (pulsarAmount < minBurn) {
      revert MoreBurn();
    }

    pulsar.burn(pulsarAmount);
    // update the timestamp
    lastBurnTime = block.timestamp;
    // emit the event
    emit CommitBurn(msg.sender, reward, swapAmountWeth, swapAmountTitanx, pulsarAmount);
}
```

FOUND THREATS

Informational

Unused functions and variables:

```
uint256 public burnLimitTitanx = 1 ether;
function setBurnLimitTitanx(uint256 newBurnLimit) external payable onlyOwner {
    burnLimitTitanx = newBurnLimit;
}

uint256 public maxSlippage;
function setMaxSlippage(uint256 newMaxSlippage) external payable onlyOwner {
    maxSlippage = newMaxSlippage;
}
```

CONTRACT 3 INFO

Token Name
unavailable

Symbol
unavailable

Contract Address
0x160c553ABFe8c645a3b6DFa56E8bFeFe2a3b5ce5

Network
PulseChain V4 testnet

Language
Solidity

Deployment Date
Oct 02, 2024

Contract Type
Bridge interface

Total Supply
unavailable

Decimals
unavailable

TAXES

Buy Tax
0%

Sell Tax
0%

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 initiate the contract once.
After initiation, ownership is transferred to address(0).

```
function init(address _eTitanxAddress, address _ePulsarAddress) external {
    if (eTitanxAddress != address(0) || ePulsarAddress != address(0)) {
      revert InitStateMismatch();
    }
    if (_ePulsarAddress == address(0) || _ePulsarAddress == address(0)) {
      revert InvalidInput();
    }

    if (msg.sender != owner) {
      revert OnlyOwner();
    }

    eTitanxAddress = _eTitanxAddress;
    ePulsarAddress = _ePulsarAddress;
    _createPool();
    owner = address(0);
}
```

FOUND THREATS

Informational

Only liquidity operator can add Pulse/TitanX liquidity via addPulseXLiquidity().

```
function addPulseXLiquidity(
    uint256 pulsarAmount,
    uint256 titanxAmount,
    uint256 slippage,
    uint256 nonce,
    uint256 deadline,
    bytes calldata signature
  ) external initialized {
    if (liquidityOperator == address(0)) {
      revert InvalidInput();
    }
    if (IERC20(ePulsarAddress).balanceOf(address(this)) < pulsarAmount) {
      revert InsufficientBalance();
    }
    if (IERC20(eTitanxAddress).balanceOf(address(this)) < titanxAmount) {
      revert InsufficientBalance();
    }
    _verifyLiquidityInfo(pulsarAmount, titanxAmount, slippage, nonce, deadline, signature, liquidityOperator);

    TransferHelper.safeApprove(ePulsarAddress, UNISWAPV2ROUTER, pulsarAmount);
    TransferHelper.safeApprove(eTitanxAddress, UNISWAPV2ROUTER, titanxAmount);

    // deadline is checked here
    IUniswapV2Router(UNISWAPV2ROUTER).addLiquidity(
      ePulsarAddress,
      eTitanxAddress,
      pulsarAmount,
      titanxAmount,
      (pulsarAmount * (tenK - slippage)) / tenK,
      (titanxAmount * (tenK - slippage)) / tenK,
      address(this),
      deadline
    );
}
```

According to project’s whitepaper:

Pulsar and TitanX: Symbiotic Tokenomics
Pulsar’s minting mechanics are deeply intertwined with the virtual mining approach of TitanX, a process where users can mint TitanX by engaging in a system that becomes progressively challenging over time.
The difficulty increases in TitanX mining inherently affects Pulsar’s minting conditions, ensuring that Pulsar’s accessibility is balanced with the evolving dynamics of the TitanX ecosystem.

Total supply and distribution model.
Minting process and the 4:1 ratio with TitanX.
Fee structure and its distribution.
Eth/TitanX to Pulsar Arbitrage.

For more detailed information, visit project’s whitepaper page:
https://pulsar.win/wp-content/uploads/2024/09/PulsarWhitePaper_V2.pdf

TOKENOMICS

WEBSITE

Website URL:
https://pulsar.win/

Domain Registry
https://www.networksolutions.com

Domain Expiration
2025-12-28

Technical SEO Test
Passed

Security Test
Passed. SSL certificate present

Design
Single page design with appropriate color scheme and graphics.

Content
The information helps new investors understand what the product does right away. No grammar mistakes found.

Whitepaper
Well written and explanatory.

Roadmap
No

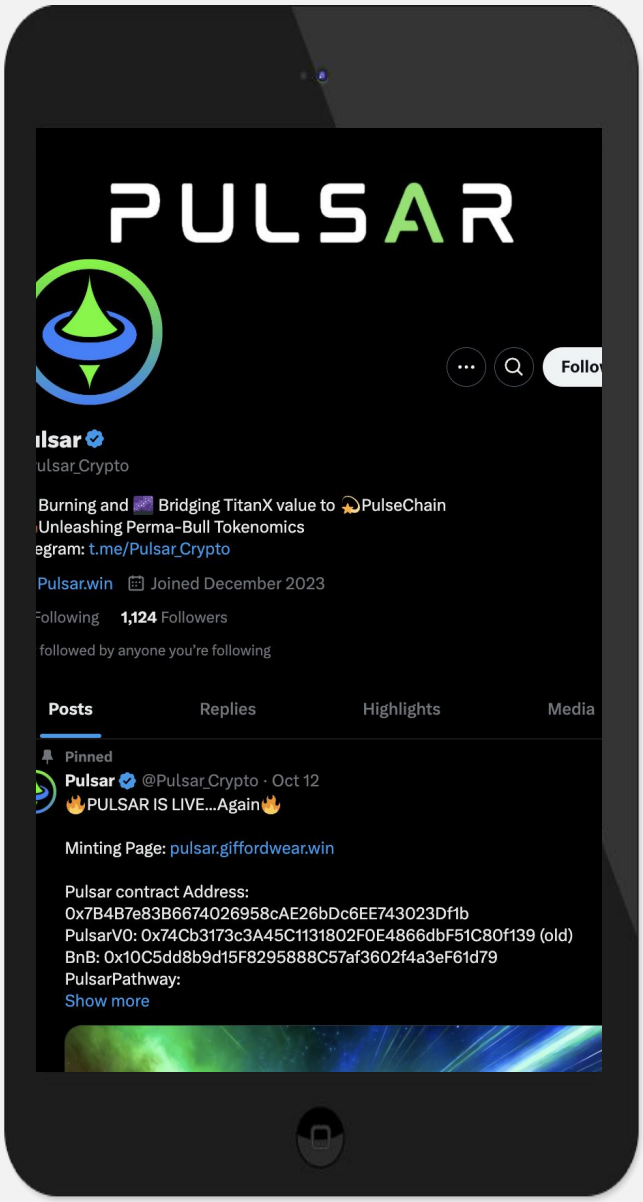
Mobile-friendly?
Yes

pulsar.win

SOCIAL MEDIA

ANALYSIS

The project's social media pages are active.



Twitter:

@Pulsar_Crypto

- 1 155 Followers
- Active

Discord

unavailable

Telegram:

@Pulsar_Crypto

- 970 members
- Active members
- Active mods

Medium

unavailable

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**

SPYWOLF

CRYPTO SECURITY

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.