SPYWOLF.CO

@SPYWOLFNETWORK

@SPYWOLFNETWORK

SPYWOLF Security Audit Report

Audit prepared for

Morpheus

Completed on

October 9, 2024

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

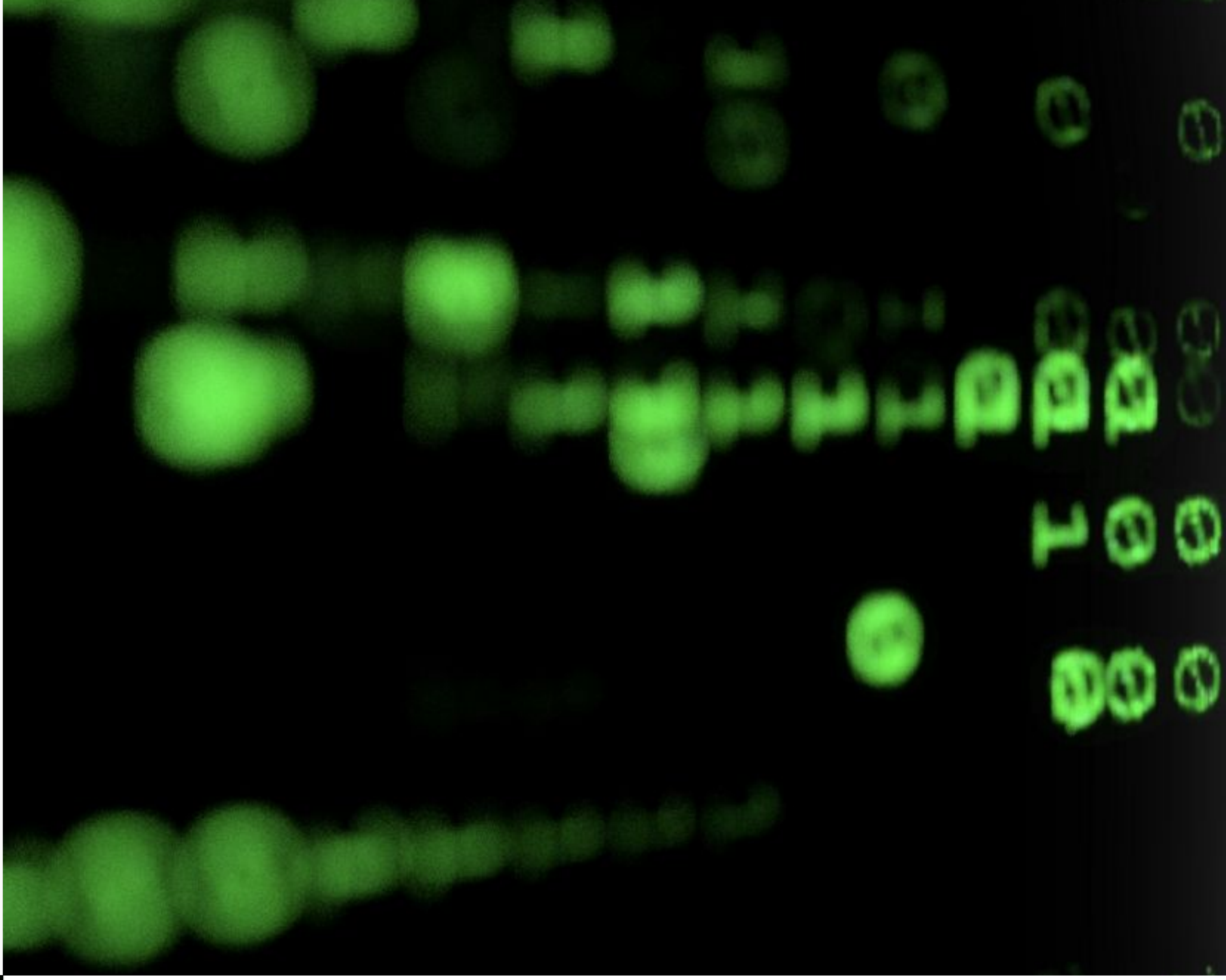
Cannot mint new tokens	*
Cannot pause trading (honeypot)	PASSED
Cannot blacklist an address	PASSED
Cannot raise taxes over 25%?	PASSED
No proxy contract detected	PASSED
Not required to enable trading	PASSED
No hidden ownership	PASSED
Cannot change the router	PASSED
No cooldown feature found	PASSED
Bot 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

Morpheus



PROJECT DESCRIPTION:

According to their whitepaper:

Welcome to Morpheus. An ecosystem designed not by chance, but by purpose.
A system crafted with precision, driven by three assets - TitanX, DragonX and Morpheus - each interacting seamlessly, building a future of decentralized value.
Everything you know about liquidity, scarcity, and power within finance is about to be redefined.
But first, you must see the path...

Release Date: TBA

Launchpad: Fairlaunch

Category: DeFi

CONTRACT 1 INFO

Token Name MORPHEUS	Symbol MORPH
Contract Address unavailable	
Network unavailable	Language Solidity
Deployment Date unavailable	Contract Type Mintable token
Total Supply 50,000,000,000	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.

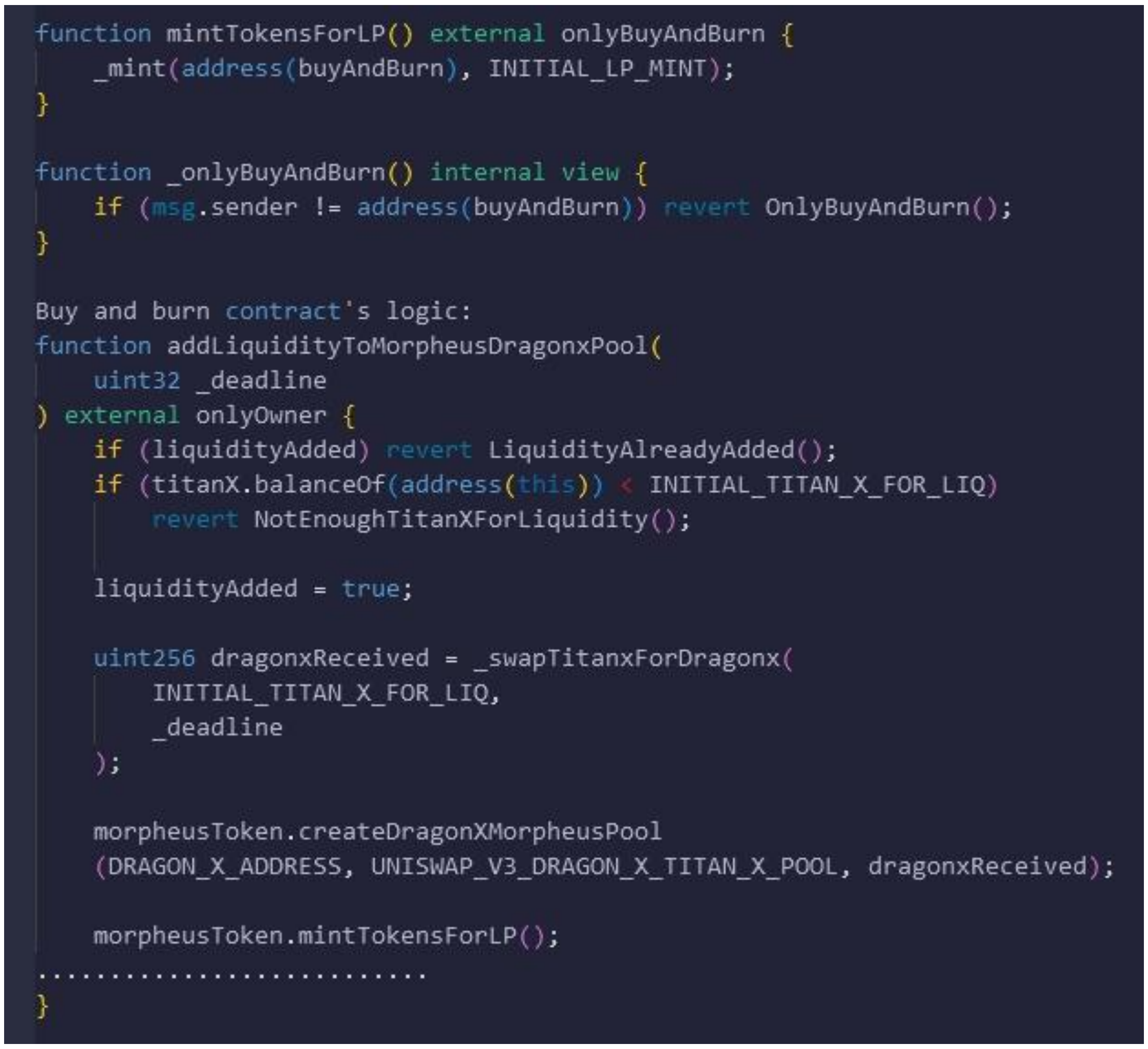
Low Risk

No low risk-level threats found in this contract.

FOUND THREATS

Informational

Buyandburn contract's owner can mint tokens for LP once.

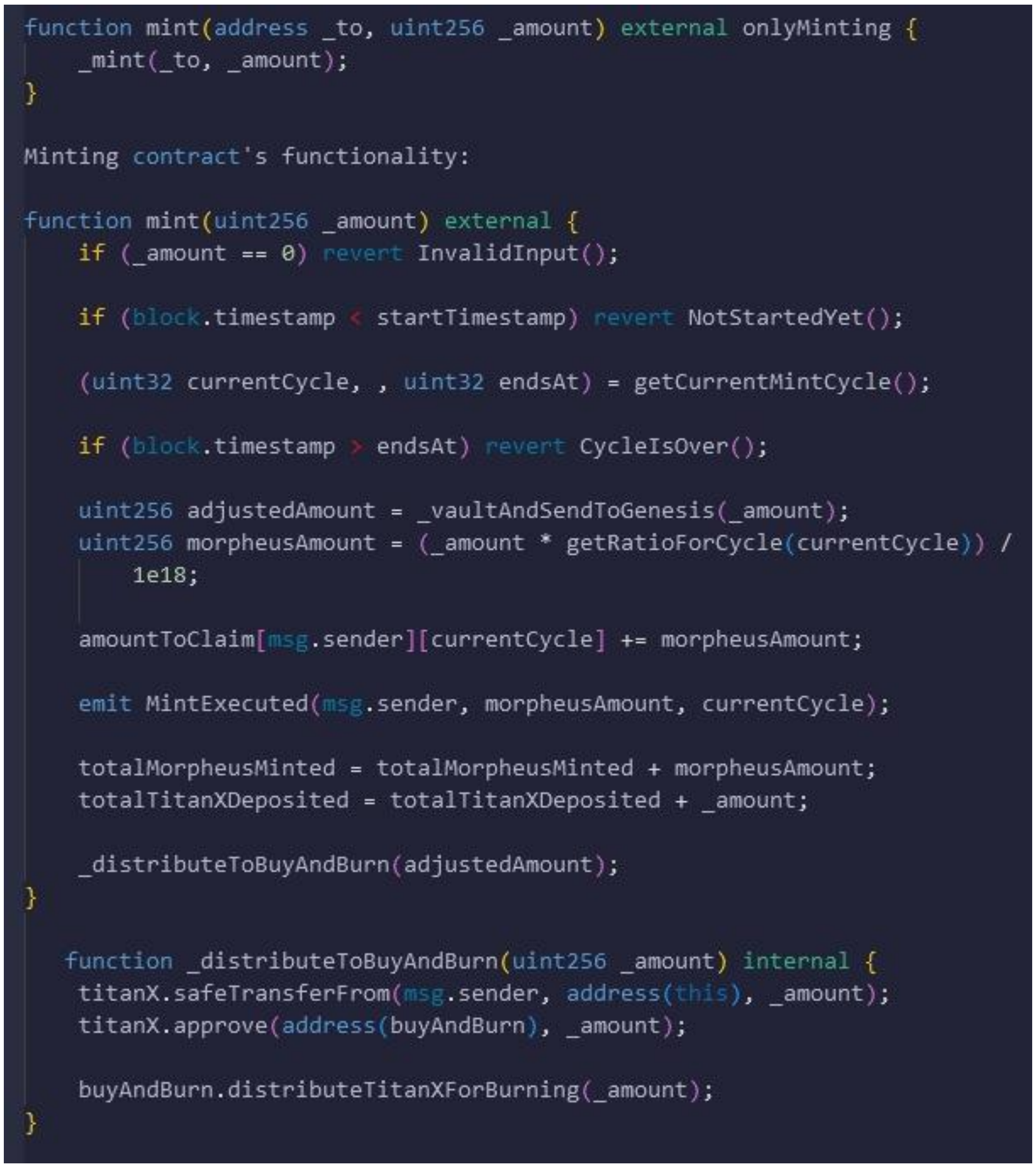


```
function mintTokensForLP() external onlyBuyAndBurn {
    _mint(address(buyAndBurn), INITIAL_LP_MINT);
}

function _onlyBuyAndBurn() internal view {
    if (msg.sender != address(buyAndBurn)) revert OnlyBuyAndBurn();
}

Buy and burn contract's logic:
function addLiquidityToMorpheusDragonxPool(
    uint32 _deadline
) external onlyOwner {
    if (liquidityAdded) revert LiquidityAlreadyAdded();
    if (titanX.balanceOf(address(this)) < INITIAL_TITAN_X_FOR_LIQ)
        revert NotEnoughTitanXForLiquidity();

    liquidityAdded = true;

    uint256 dragonxReceived = _swapTitanxForDragonx(
        INITIAL_TITAN_X_FOR_LIQ,
        _deadline
    );

    morpheusToken.createDragonXMorpheusPool
    (DRAGON_X_ADDRESS, UNISWAP_V3_DRAGON_X_TITAN_X_POOL, dragonxReceived);

    morpheusToken.mintTokensForLP();
..........................
}
```

FOUND THREATS

Informational

New tokens can be minted via minting contract during the minting phases which can be up to 14 days after startTimeStamp.



```
function mint(address _to, uint256 _amount) external onlyMinting {
    _mint(_to, _amount);
}

Minting contract's functionality:

function mint(uint256 _amount) external {
    if (_amount == 0) revert InvalidInput();

    if (block.timestamp < startTimestamp) revert NotStartedYet();

    (uint32 currentCycle, , uint32 endsAt) = getCurrentMintCycle();

    if (block.timestamp > endsAt) revert CycleIsOver();

    uint256 adjustedAmount = _vaultAndSendToGenesis(_amount);
    uint256 morpheusAmount = (_amount * getRatioForCycle(currentCycle)) /
        1e18;

    amountToClaim[msg.sender][currentCycle] += morpheusAmount;

    emit MintExecuted(msg.sender, morpheusAmount, currentCycle);

    totalMorpheusMinted = totalMorpheusMinted + morpheusAmount;
    totalTitanXDeposited = totalTitanXDeposited + _amount;

    _distributeToBuyAndBurn(adjustedAmount);
}

   function _distributeToBuyAndBurn(uint256 _amount) internal {
    titanX.safeTransferFrom(msg.sender, address(this), _amount);
    titanX.approve(address(buyAndBurn), _amount);

    buyAndBurn.distributeTitanXForBurning(_amount);
}
```

FOUND THREATS

Informational

DragonX and TitanX tokens are not in the scope of the current audit.

CONTRACT 2 INFO

Token Name unavailable	Symbol unavailable
Contract Address unavailable	
Network unavailable	Language Solidity
Deployment Date unavailable	Contract Type Buy and burn interface
Total Supply unavailable	Decimals unavailable

TAXES

Buy Tax	Sell Tax
0%	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.

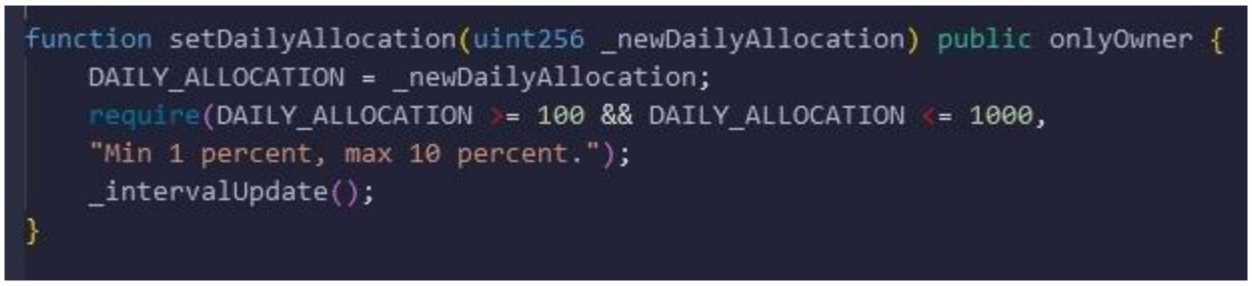
⚠ Low Risk

No low risk-level threats found in this contract.

FOUND THREATS

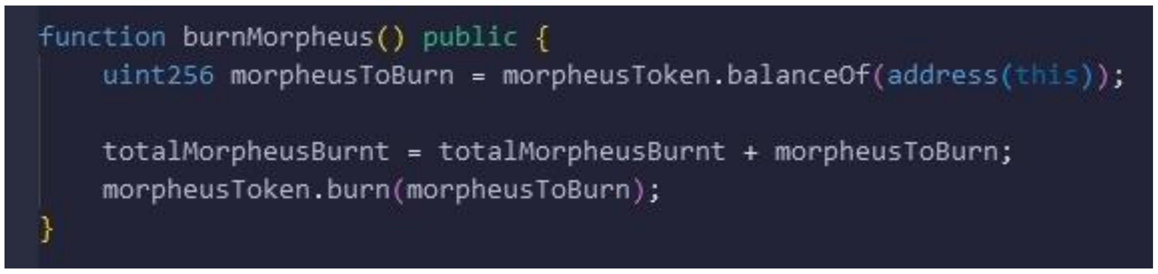
Informational

Owner can set daily TitanX allocation for distribution.
Daily allocation (tokens amount distributed per day) can be set between 1% and 10% of contract's current TitanX balances.



```
function setDailyAllocation(uint256 _newDailyAllocation) public onlyOwner {
    DAILY_ALLOCATION = _newDailyAllocation;
    require(DAILY_ALLOCATION >= 100 && DAILY_ALLOCATION <= 1000,
    "Min 1 percent, max 10 percent.");
    _intervalUpdate();
}
```

Anyone can call burnMorpheus() function and initiate Morpheus tokens burn.
This will burn the entire contract's Morpheus token balances.

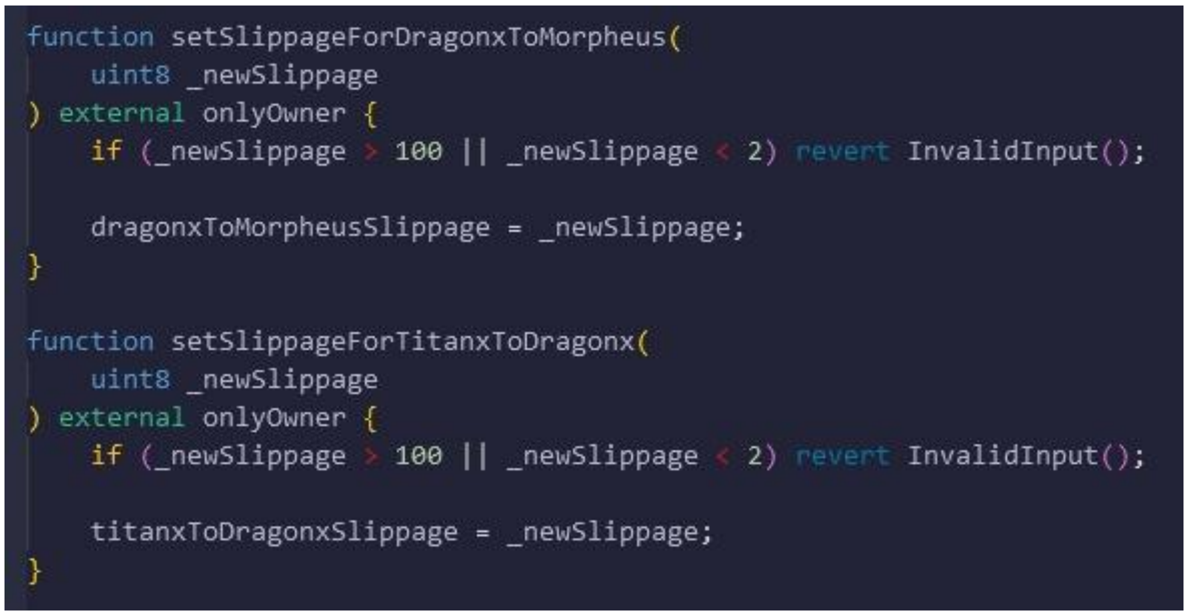


```
function burnMorpheus() public {
    uint256 morpheusToBurn = morpheusToken.balanceOf(address(this));

    totalMorpheusBurnt = totalMorpheusBurnt + morpheusToBurn;
    morpheusToken.burn(morpheusToBurn);
}
```

FOUND THREATS

Informational

Owner can set slippage for token swaps of DragonX to Morpheus and TitanX to DragonX.
Slippage can be set between 2% and 100%.

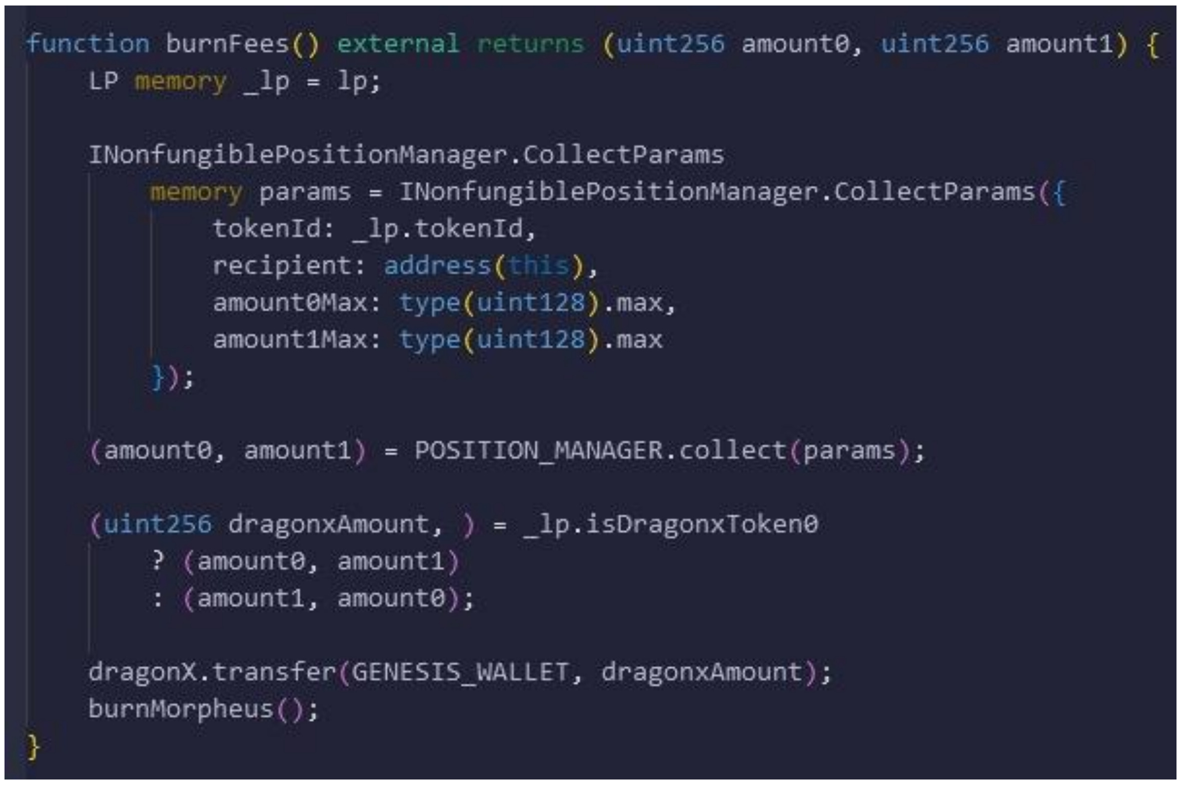


```
function setSlippageForDragonxToMorpheus(
    uint8 _newSlippage
) external onlyOwner {
    if (_newSlippage > 100 || _newSlippage < 2) revert InvalidInput();

    dragonxToMorpheusSlippage = _newSlippage;
}

function setSlippageForTitanxToDragonx(
    uint8 _newSlippage
) external onlyOwner {
    if (_newSlippage > 100 || _newSlippage < 2) revert InvalidInput();

    titanxToDragonxSlippage = _newSlippage;
}
```

FOUND THREATS

Informational

Anyone can use burnFees() function, which will sent the dragonX tokens to Genesis wallet and burn the available Morpheus tokens in the contract.

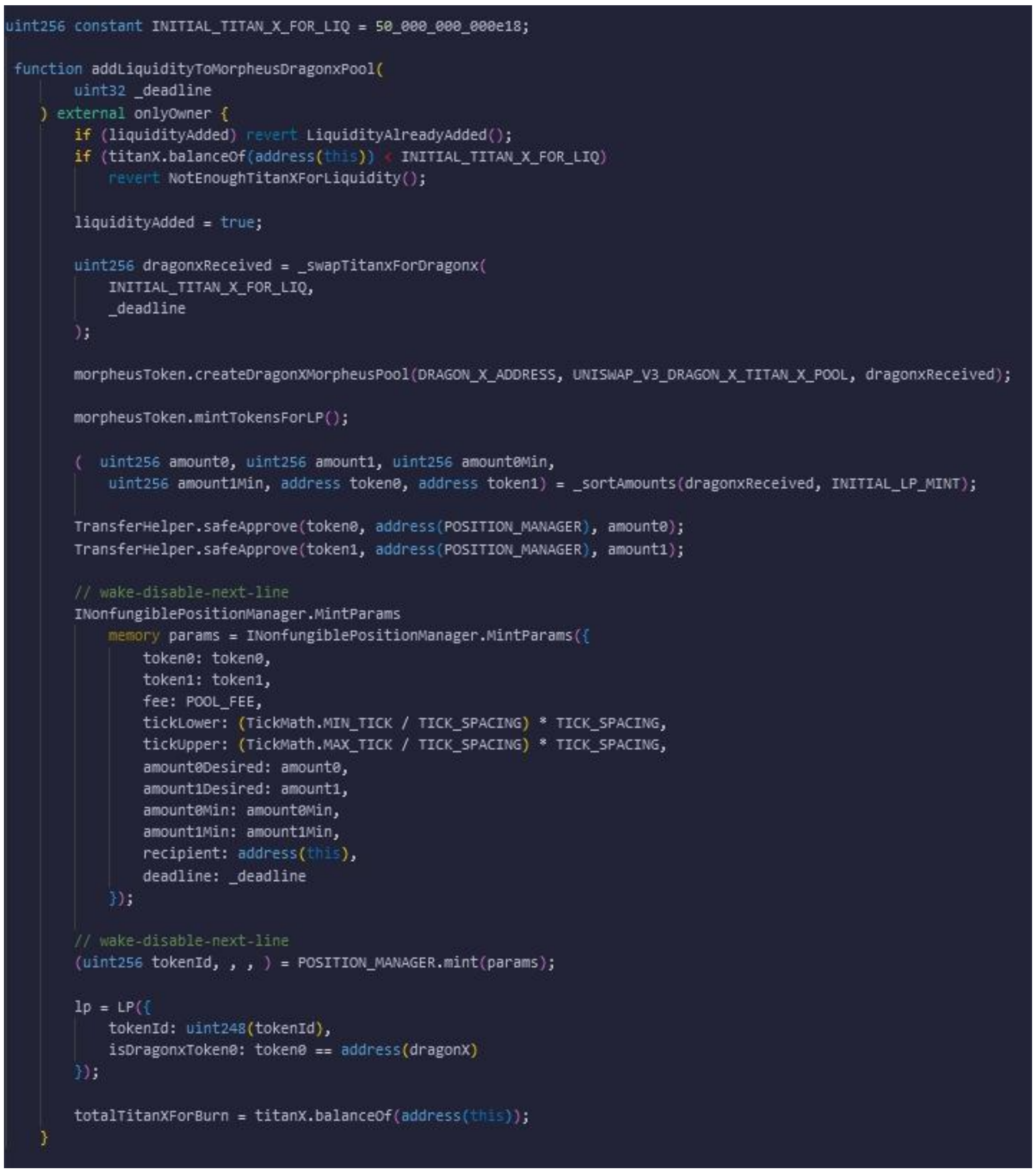


```
function burnFees() external returns (uint256 amount0, uint256 amount1) {
    LP memory _lp = lp;

    INonfungiblePositionManager.CollectParams
        memory params = INonfungiblePositionManager.CollectParams({
            tokenId: _lp.tokenId,
            recipient: address(this),
            amount0Max: type(uint128).max,
            amount1Max: type(uint128).max
        });

    (amount0, amount1) = POSITION_MANAGER.collect(params);

    (uint256 dragonxAmount, ) = _lp.isDragonxToken0
        ? (amount0, amount1)
        : (amount1, amount0);

    dragonX.transfer(GENESIS_WALLET, dragonxAmount);
    burnMorpheus();
}
```

FOUND THREATS

Informational

Owner can create DragonX-Morhpeus pool to Morpheus token and add liquidity once. BuyAndBurn contract's balances should not be less than 50,000,000,000 TitanX tokens in order to add liquidity.



```
uint256 constant INITIAL_TITAN_X_FOR_LIQ = 50_000_000_000e18;

 function addLiquidityToMorpheusDragonxPool(
        uint32 _deadline
    ) external onlyOwner {
        if (liquidityAdded) revert LiquidityAlreadyAdded();
        if (titanX.balanceOf(address(this)) < INITIAL_TITAN_X_FOR_LIQ)
            revert NotEnoughTitanXForLiquidity();

        liquidityAdded = true;

        uint256 dragonxReceived = _swapTitanxForDragonx(
            INITIAL_TITAN_X_FOR_LIQ,
            _deadline
        );

        morpheusToken.createDragonXMorpheusPool(DRAGON_X_ADDRESS, UNISWAP_V3_DRAGON_X_TITAN_X_POOL, dragonxReceived);

        morpheusToken.mintTokensForLP();

        (  uint256 amount0, uint256 amount1, uint256 amount0Min,
            uint256 amount1Min, address token0, address token1) = _sortAmounts(dragonxReceived, INITIAL_LP_MINT);

        TransferHelper.safeApprove(token0, address(POSITION_MANAGER), amount0);
        TransferHelper.safeApprove(token1, address(POSITION_MANAGER), amount1);

        // wake-disable-next-line
        INonfungiblePositionManager.MintParams
            memory params = INonfungiblePositionManager.MintParams({
                token0: token0,
                token1: token1,
                fee: POOL_FEE,
                tickLower: (TickMath.MIN_TICK / TICK_SPACING) * TICK_SPACING,
                tickUpper: (TickMath.MAX_TICK / TICK_SPACING) * TICK_SPACING,
                amount0Desired: amount0,
                amount1Desired: amount1,
                amount0Min: amount0Min,
                amount1Min: amount1Min,
                recipient: address(this),
                deadline: _deadline
            });

        // wake-disable-next-line
        (uint256 tokenId, , , ) = POSITION_MANAGER.mint(params);

        lp = LP({
            tokenId: uint248(tokenId),
            isDragonxToken0: token0 == address(dragonX)
        });

        totalTitanXForBurn = titanX.balanceOf(address(this));
    }
```

CONTRACT 3 INFO

Token Name unavailable	Symbol unavailable
Contract Address unavailable	
Network unavailable	Language Solidity
Deployment Date unavailable	Contract Type Minting interface
Total Supply unavailable	Decimals unavailable

TAXES

Buy Tax	Sell Tax
0%	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.

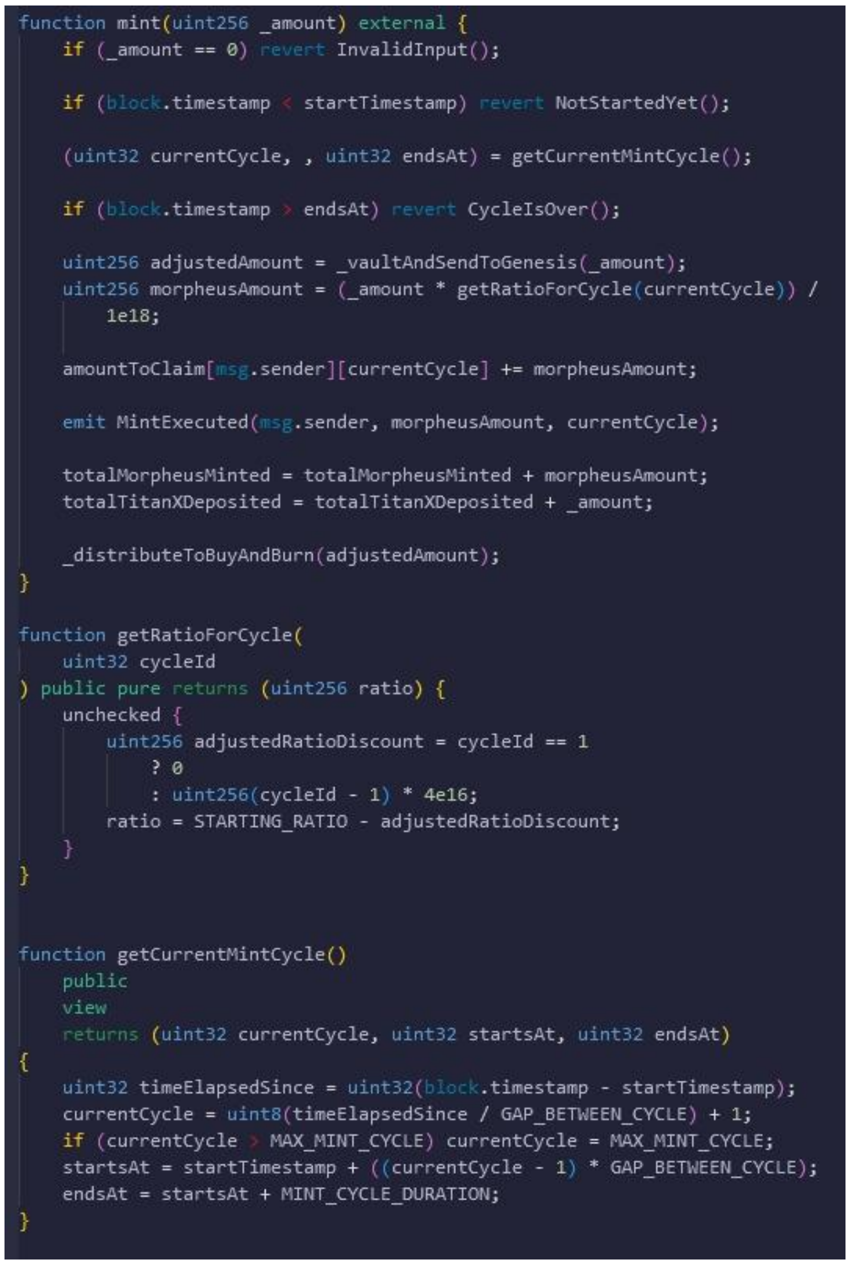
⚠ Low Risk

No low risk-level threats found in this contract.

FOUND THREATS

Informational

If getCurrentMintCycle() returns value higher than 25 for 'currentCycle' local variable, overflow occurs in getRatioForCycle() since it uses unchecked math.
This behavior can lead to excessive mint amounts of morpheus tokens.
Take above in mind if choose to change math formulas in further implementation.

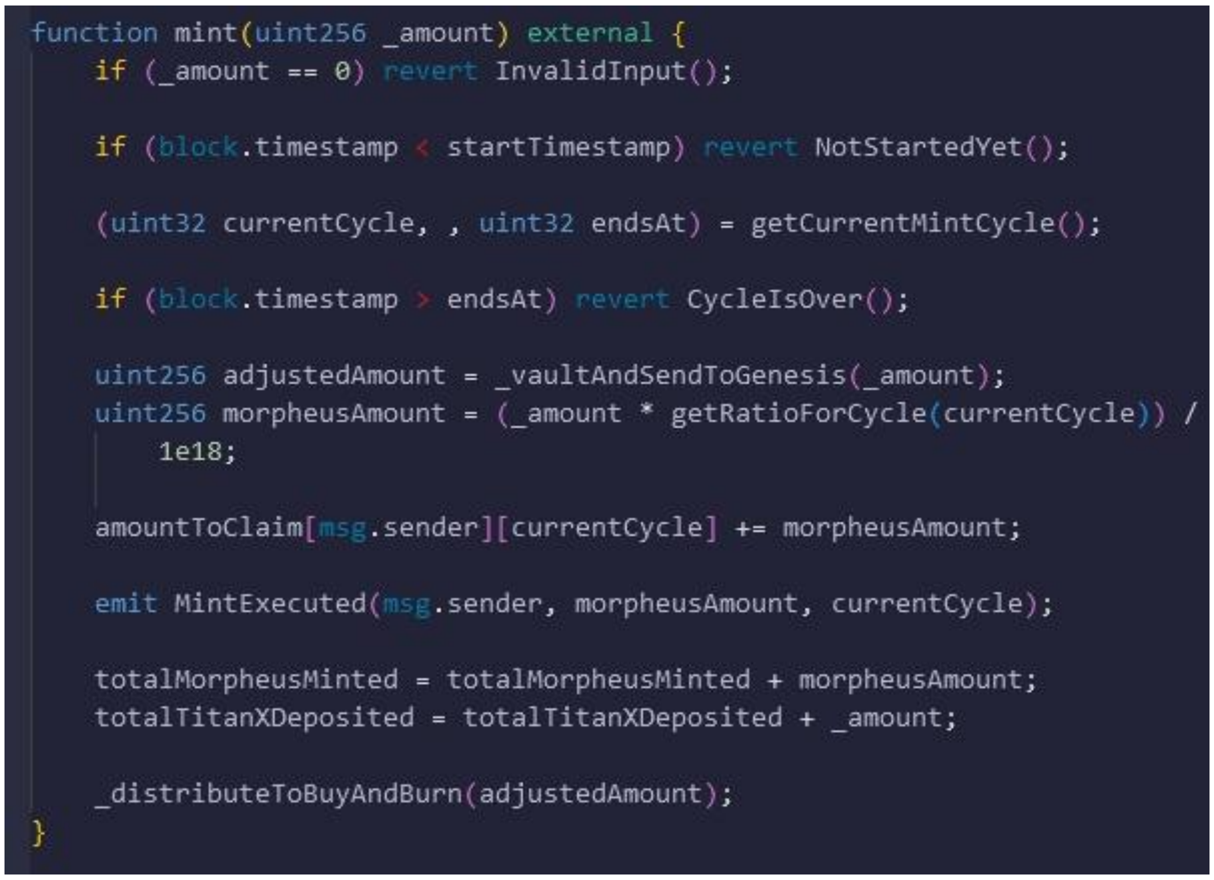


```
function mint(uint256 _amount) external {
    if (_amount == 0) revert InvalidInput();

    if (block.timestamp < startTimestamp) revert NotStartedYet();

    (uint32 currentCycle, , uint32 endsAt) = getCurrentMintCycle();

    if (block.timestamp > endsAt) revert CycleIsOver();

    uint256 adjustedAmount = _vaultAndSendToGenesis(_amount);
    uint256 morpheusAmount = (_amount * getRatioForCycle(currentCycle)) /
        1e18;

    amountToClaim[msg.sender][currentCycle] += morpheusAmount;

    emit MintExecuted(msg.sender, morpheusAmount, currentCycle);

    totalMorpheusMinted = totalMorpheusMinted + morpheusAmount;
    totalTitanXDeposited = totalTitanXDeposited + _amount;

    _distributeToBuyAndBurn(adjustedAmount);
}

function getRatioForCycle(
    uint32 cycleId
) public pure returns (uint256 ratio) {
    unchecked {
        uint256 adjustedRatioDiscount = cycleId == 1
            ? 0
            : uint256(cycleId - 1) * 4e16;
        ratio = STARTING_RATIO - adjustedRatioDiscount;
    }
}


function getCurrentMintCycle()
    public
    view
    returns (uint32 currentCycle, uint32 startsAt, uint32 endsAt)
{
    uint32 timeElapsedSince = uint32(block.timestamp - startTimestamp);
    currentCycle = uint8(timeElapsedSince / GAP_BETWEEN_CYCLE) + 1;
    if (currentCycle > MAX_MINT_CYCLE) currentCycle = MAX_MINT_CYCLE;
    startsAt = startTimestamp + ((currentCycle - 1) * GAP_BETWEEN_CYCLE);
    endsAt = startsAt + MINT_CYCLE_DURATION;
}
```

FOUND THREATS

Informational

Users can issue (mint) Morpheus tokens in exchange of TitanX tokens.

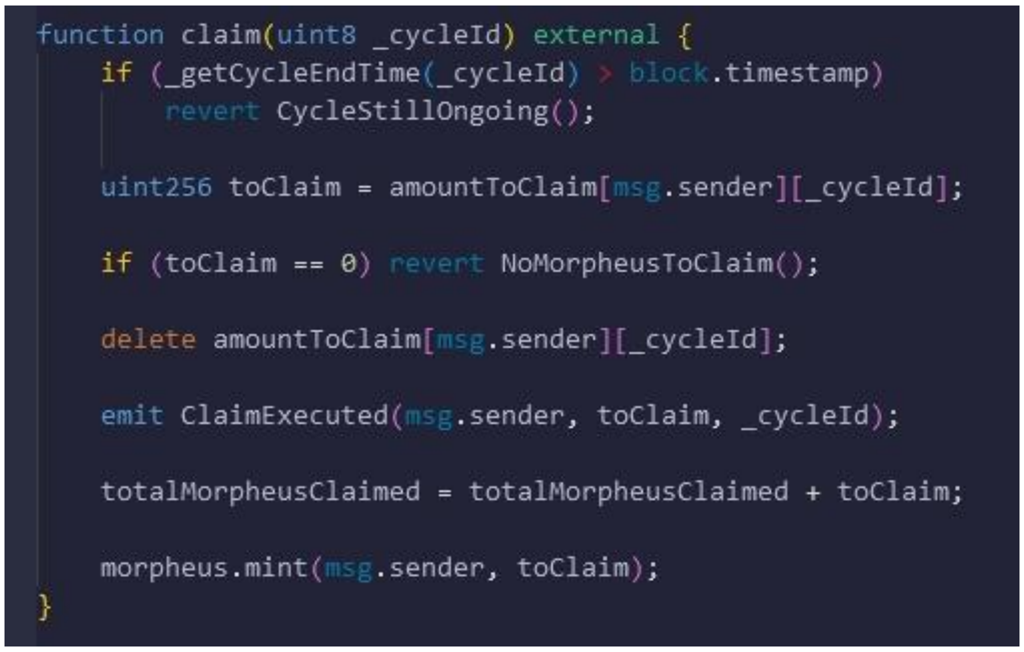


```
function mint(uint256 _amount) external {
    if (_amount == 0) revert InvalidInput();

    if (block.timestamp < startTimestamp) revert NotStartedYet();

    (uint32 currentCycle, , uint32 endsAt) = getCurrentMintCycle();

    if (block.timestamp > endsAt) revert CycleIsOver();

    uint256 adjustedAmount = _vaultAndSendToGenesis(_amount);
    uint256 morpheusAmount = (_amount * getRatioForCycle(currentCycle)) /
        1e18;

    amountToClaim[msg.sender][currentCycle] += morpheusAmount;

    emit MintExecuted(msg.sender, morpheusAmount, currentCycle);

    totalMorpheusMinted = totalMorpheusMinted + morpheusAmount;
    totalTitanXDeposited = totalTitanXDeposited + _amount;

    _distributeToBuyAndBurn(adjustedAmount);
}
```

FOUND THREATS

Informational

Users can claim their issued (minted) Morpheus tokens at any time after the current minting cycle is over.



```
function claim(uint8 _cycleId) external {
    if (_getCycleEndTime(_cycleId) > block.timestamp)
        revert CycleStillOngoing();

    uint256 toClaim = amountToClaim[msg.sender][_cycleId];

    if (toClaim == 0) revert NoMorpheusToClaim();

    delete amountToClaim[msg.sender][_cycleId];

    emit ClaimExecuted(msg.sender, toClaim, _cycleId);

    totalMorpheusClaimed = totalMorpheusClaimed + toClaim;

    morpheus.mint(msg.sender, toClaim);
}
```

TOKENOMICS

There is no information about the initial tokens distribution based on the project's whitepaper and/or website.

WEBSITE

Website URL:
https://www.morpheus.win

Domain Registry
https://www.godaddy.com

Domain Expiration
2025-09-24

Technical SEO Test
Passed

Security Test
Passed. SSL certificate present

Design
Website is under construction

Content
Website is under construction

Whitepaper
Explanatory, tokenomics mechanics can be more detailed.

Roadmap
No

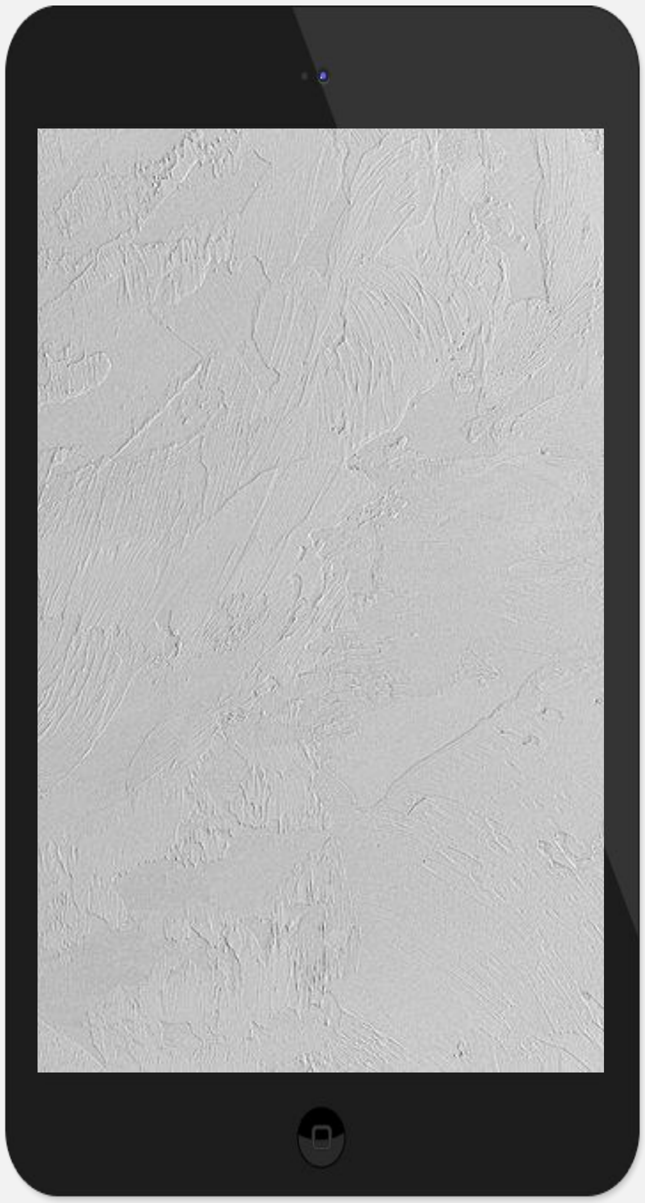
Mobile-friendly?
Yes

Under Construction

SOCIAL MEDIA

ANALYSIS

Project’s social media pages are under construction.



Twitter:

unavailable

Discord

unavailable

Telegram:

unavailable

Medium

unavailable

SPYWOLF

CRYPTO SECURITY

Audits | KYCs | dApps
Contract Development

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

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.