SPYWOLF.CO
@SPYWOLFNETWORK
@SPYWOLFNETWORK

SPYWOLF

Security Audit Report

Audit prepared for

Bonfire

Completed on

December 24, 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

BonFire

PROJECT DESCRIPTION:

According to their whitepaper:

BonFire (BFX) is an innovative decentralized finance (DeFi) project built on the X28 ecosystem, leveraging cutting-edge Smart Burn Technology to revolutionize traditional buy-and-burn protocols. BonFire is designed to maximize the efficiency and impact of buy-and-burn mechanisms, elevating deflationary tokenomics to new heights.

Key Features:

Fair Launch: Combines a virtual mining and auction hybrid model for equitable access.

Decentralized & Deflationary: Focused on community empowerment and a robust burning mechanism.

Smart Burn Power: The ultimate engine driving BonFire's unique deflationary model.

BonFireTest Info

Token Name BonFireTest	Symbol TBFX
Contract Address 0x1F4560aDAB22b59DA654b03BDeac286a7E18E4D5	
Network ETH - Sepolia testnet	Language Solidity
Deployment Date Dec 18, 2024	Contract Type Mintable token
Total Supply 2,761,725,720	Decimals 18

TAXES

Buy Tax	Sell Tax
16%	0%

*"Buy tax" to be readed as "Mint tax".
Mint taxes are distributed towards: LP (8%), Dev (4%) and GENESIS (4%) addresses.

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.

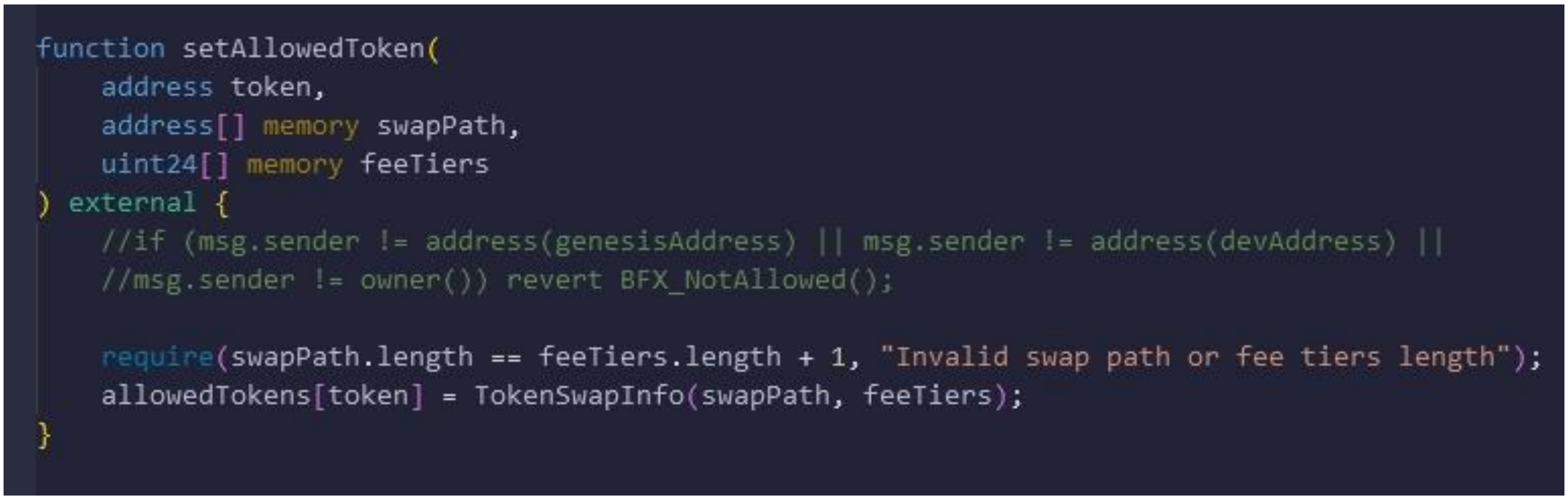
FOUND THREATS

⚠ Medium Risk

Anyone can set swapPath for tokens.

The setAllowedToken function lacks access control, allowing anyone to modify swap paths (swapPath) and pool fees (feeTiers).

Ensure to always insert the correct swap path and pool fees between tokens. Incorrectly set path and/or pool fees might lead to transactions revert and/or loss of funds if wrong pool is targeted.



```
function setAllowedToken(
    address token,
    address[] memory swapPath,
    uint24[] memory feeTiers
) external {
    //if (msg.sender != address(genesisAddress) || msg.sender != address(devAddress) ||
    //msg.sender != owner()) revert BFX_NotAllowed();

    require(swapPath.length == feeTiers.length + 1, "Invalid swap path or fee tiers length");
    allowedTokens[token] = TokenSwapInfo(swapPath, feeTiers);
}
```

- Recommendation:
    - Add Access Control: Restrict function to trusted roles.
    - Validate Inputs: Ensure swapPath and feeTiers define valid pools.
    - Emit Events: Log changes for traceability.

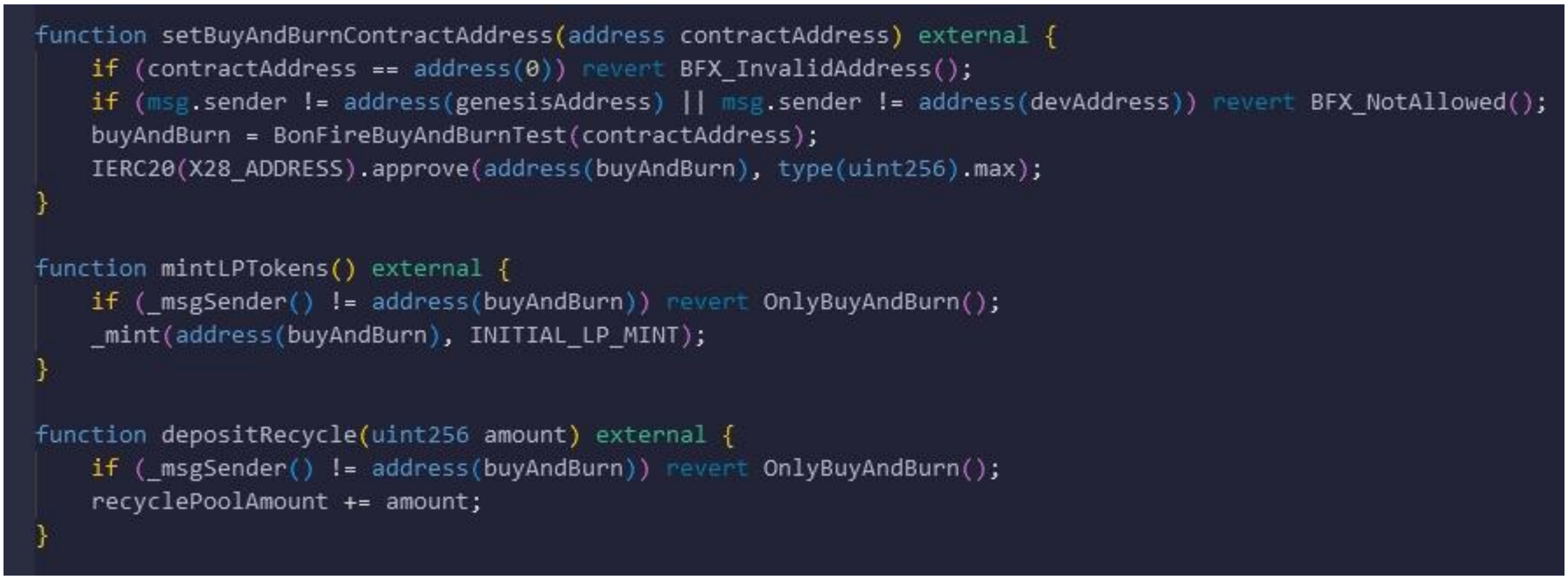
FOUND THREATS

Informational

Genesis and devAddress have the ability to update the buyAndBurn contract address using the setBuyAndBurnContractAddress function, which grants significant control over the system's operations.

The buyAndBurn address itself holds key privileges, including the ability to mint new LP tokens via mintLPTokens and adjust the recyclePoolAmount through depositRecycle. While these features align with the intended design, they emphasize the importance of securing these privileged roles.

Developers should ensure that proper access controls are in place and that the buyAndBurn contract is audited to prevent misuse or unintended behavior. Additionally, logging changes to these critical variables can enhance traceability and operational transparency.



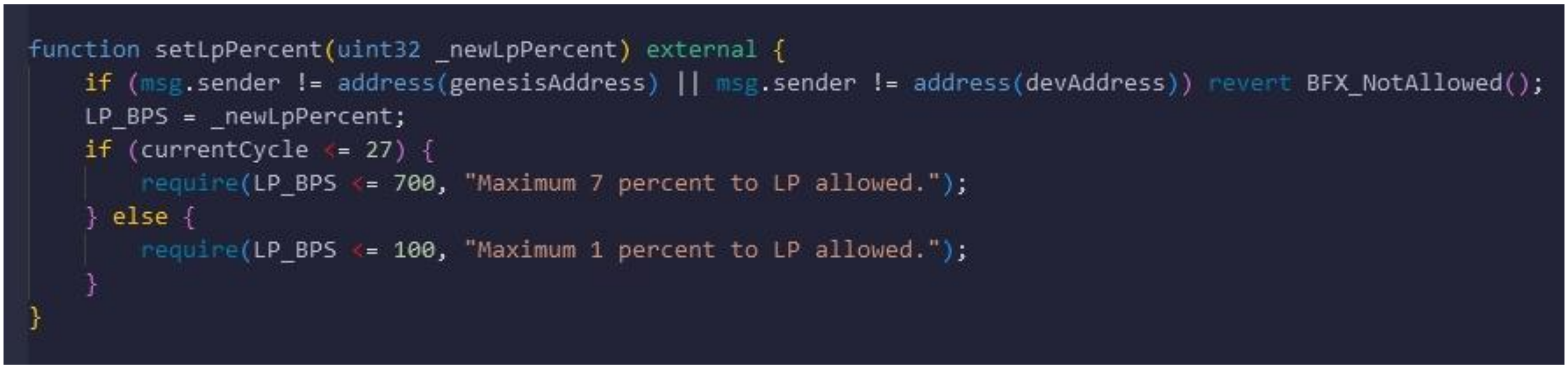
```
function setBuyAndBurnContractAddress(address contractAddress) external {
    if (contractAddress == address(0)) revert BFX_InvalidAddress();
    if (msg.sender != address(genesisAddress) || msg.sender != address(devAddress)) revert BFX_NotAllowed();
    buyAndBurn = BonFireBuyAndBurnTest(contractAddress);
    IERC20(X28_ADDRESS).approve(address(buyAndBurn), type(uint256).max);
}

function mintLPTokens() external {
    if (_msgSender() != address(buyAndBurn)) revert OnlyBuyAndBurn();
    _mint(address(buyAndBurn), INITIAL_LP_MINT);
}

function depositRecycle(uint256 amount) external {
    if (_msgSender() != address(buyAndBurn)) revert OnlyBuyAndBurn();
    recyclePoolAmount += amount;
}
```

FOUND THREATS

Informational

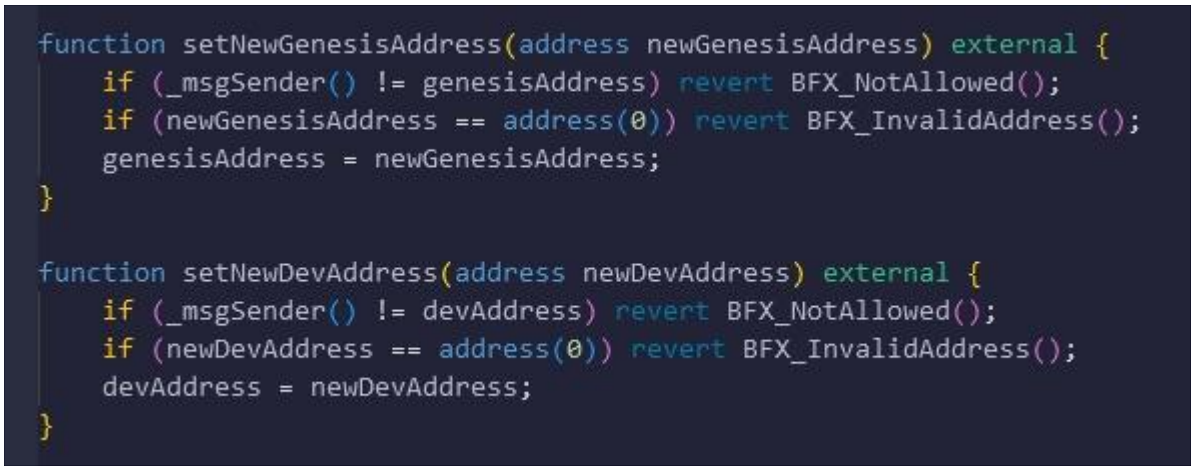
Genesis and devAddress can change percent of tokens for LP allocation up to 7%.
The setLpPercent function allows genesis and devAddress to adjust the percentage of tokens allocated for LP (Liquidity Pool) within specified limits:

- Up to 7% during the first 27 cycles.
- Up to 1% after the 27th cycle.



```
function setLpPercent(uint32 _newLpPercent) external {
    if (msg.sender != address(genesisAddress) || msg.sender != address(devAddress)) revert BFX_NotAllowed();
    LP_BPS = _newLpPercent;
    if (currentCycle <= 27) {
        require(LP_BPS <= 700, "Maximum 7 percent to LP allowed.");
    } else {
        require(LP_BPS <= 100, "Maximum 1 percent to LP allowed.");
    }
}
```

Genesis and devAddress can change percent of tokens for LP allocation up to 7%.



```
function setNewGenesisAddress(address newGenesisAddress) external {
    if (_msgSender() != genesisAddress) revert BFX_NotAllowed();
    if (newGenesisAddress == address(0)) revert BFX_InvalidAddress();
    genesisAddress = newGenesisAddress;
}

function setNewDevAddress(address newDevAddress) external {
    if (_msgSender() != devAddress) revert BFX_NotAllowed();
    if (newDevAddress == address(0)) revert BFX_InvalidAddress();
    devAddress = newDevAddress;
}
```

BonFireBuyAndBurnTest Info

Token Name unavailable	Symbol unavailable
Contract Address 0x9cC5754E64FE0B830ACb8214B269177989293331	
Network ETH - Sepolia testnet	Language Solidity
Deployment Date Dec 18, 2024	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

⚠ Low Risk

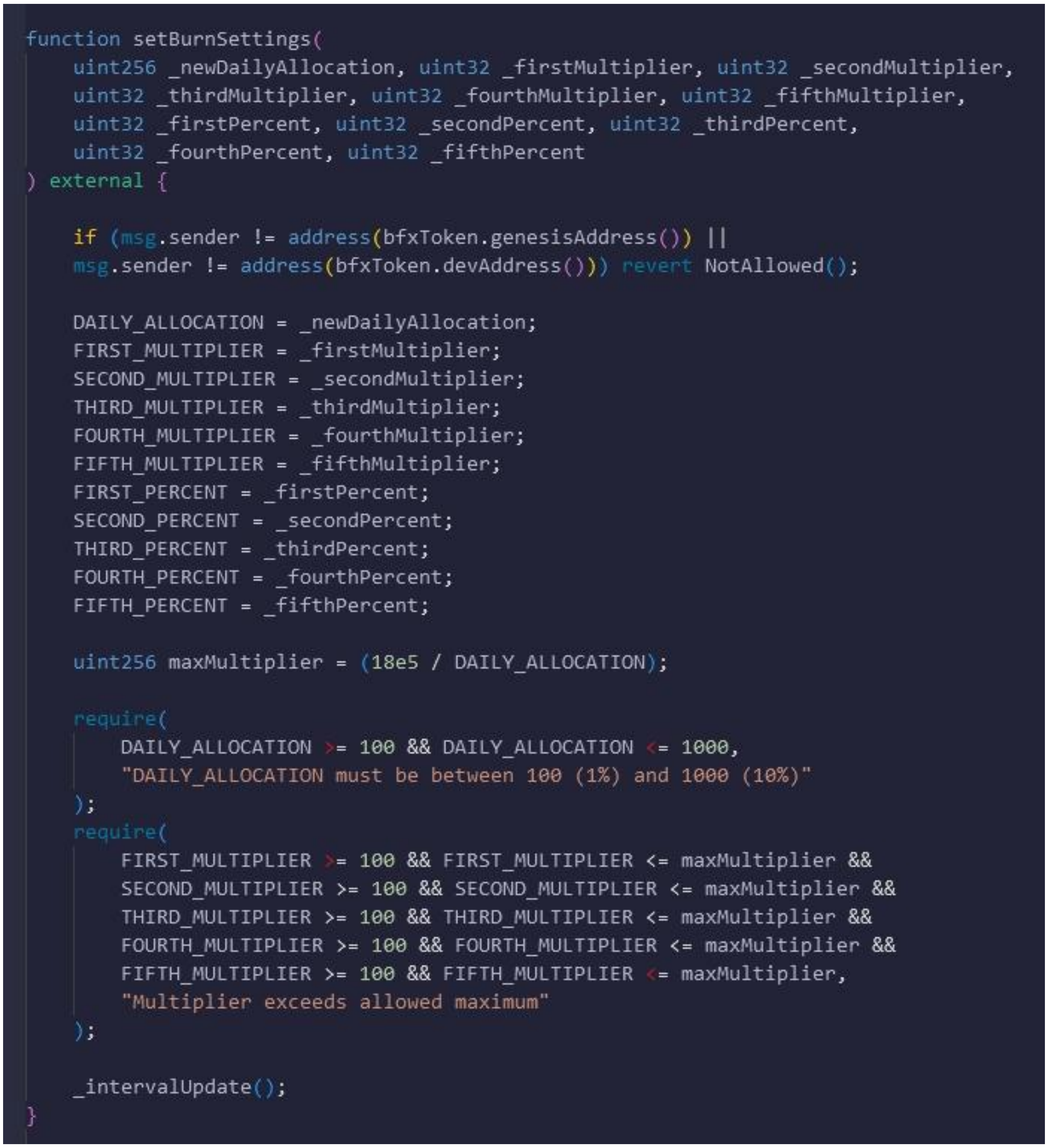
The contract relies on accurate initialization of the startTimestamp to ensure proper time-based calculations in the _intervalUpdate() function. If lastBurnedIntervalStartTimestamp is set to a value higher than the current block.timestamp, it could lead to arithmetic errors or overflows during interval updates. Such discrepancies could disrupt the burn mechanism or other time-sensitive functionalities tied to these calculations, potentially causing unexpected behavior or operational downtime. This risk arises from improper validation of the startTimestamp during contract deployment or updates, which could misalign intervals and impact contract logic.

To mitigate this, it is crucial to validate the startTimestamp during deployment, ensuring it is less than or equal to the current block.timestamp. Adding checks in _intervalUpdate() to confirm that lastBurnedIntervalStartTimestamp is always less than or equal to block.timestamp can prevent overflows and maintain consistent behavior. Additionally, emitting events whenever lastBurnedIntervalStartTimestamp or intervals are updated enhances transparency and allows for easier debugging. While this is a low-risk issue, these preventive measures improve the reliability and auditability of the contract, safeguarding critical functionality.

FOUND THREATS

Informational

Genesis and devAddress can change settings like allocation amounts per day and allocation's multipliers.

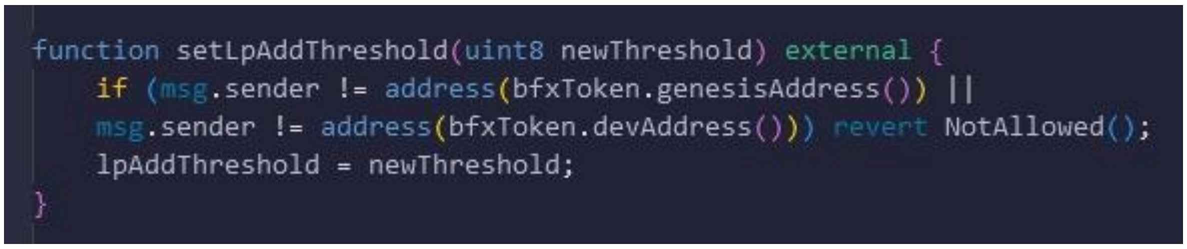


```
function setBurnSettings(
    uint256 _newDailyAllocation, uint32 _firstMultiplier, uint32 _secondMultiplier,
    uint32 _thirdMultiplier, uint32 _fourthMultiplier, uint32 _fifthMultiplier,
    uint32 _firstPercent, uint32 _secondPercent, uint32 _thirdPercent,
    uint32 _fourthPercent, uint32 _fifthPercent
) external {

    if (msg.sender != address(bfxToken.genesisAddress()) ||
    msg.sender != address(bfxToken.devAddress())) revert NotAllowed();

    DAILY_ALLOCATION = _newDailyAllocation;
    FIRST_MULTIPLIER = _firstMultiplier;
    SECOND_MULTIPLIER = _secondMultiplier;
    THIRD_MULTIPLIER = _thirdMultiplier;
    FOURTH_MULTIPLIER = _fourthMultiplier;
    FIFTH_MULTIPLIER = _fifthMultiplier;
    FIRST_PERCENT = _firstPercent;
    SECOND_PERCENT = _secondPercent;
    THIRD_PERCENT = _thirdPercent;
    FOURTH_PERCENT = _fourthPercent;
    FIFTH_PERCENT = _fifthPercent;

    uint256 maxMultiplier = (18e5 / DAILY_ALLOCATION);

    require(
        DAILY_ALLOCATION >= 100 && DAILY_ALLOCATION <= 1000,
        "DAILY_ALLOCATION must be between 100 (1%) and 1000 (10%)"
    );
    require(
        FIRST_MULTIPLIER >= 100 && FIRST_MULTIPLIER <= maxMultiplier &&
        SECOND_MULTIPLIER >= 100 && SECOND_MULTIPLIER <= maxMultiplier &&
        THIRD_MULTIPLIER >= 100 && THIRD_MULTIPLIER <= maxMultiplier &&
        FOURTH_MULTIPLIER >= 100 && FOURTH_MULTIPLIER <= maxMultiplier &&
        FIFTH_MULTIPLIER >= 100 && FIFTH_MULTIPLIER <= maxMultiplier,
        "Multiplier exceeds allowed maximum"
    );

    _intervalUpdate();
}
```

FOUND THREATS

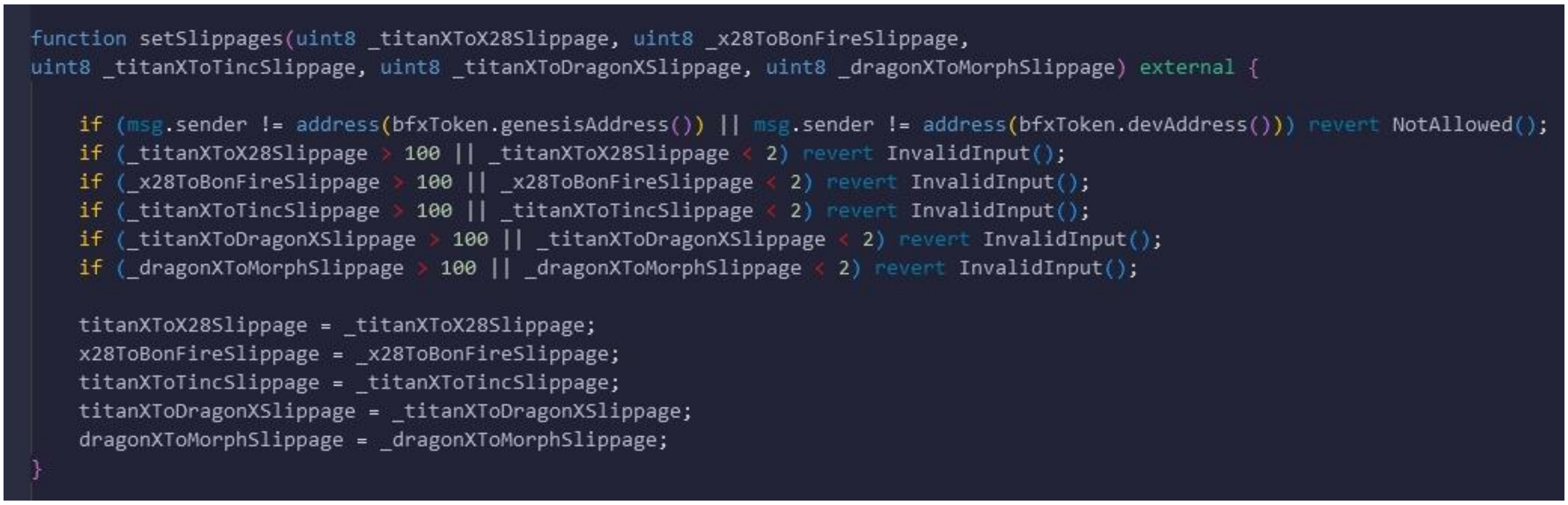
Informational

Genesis and devAddress can change liquidity add tokens threshold.



```
function setLpAddThreshold(uint8 newThreshold) external {
    if (msg.sender != address(bfxToken.genesisAddress()) ||
    msg.sender != address(bfxToken.devAddress())) revert NotAllowed();
    lpAddThreshold = newThreshold;
}
```

Genesis and devAddress can change liquidity add tokens threshold.



```
function setSlippages(uint8 _titanXToX28Slippage, uint8 _x28ToBonFireSlippage,
uint8 _titanXToTincSlippage, uint8 _titanXToDragonXSlippage, uint8 _dragonXToMorphSlippage) external {

    if (msg.sender != address(bfxToken.genesisAddress()) || msg.sender != address(bfxToken.devAddress())) revert NotAllowed();
    if (_titanXToX28Slippage > 100 || _titanXToX28Slippage < 2) revert InvalidInput();
    if (_x28ToBonFireSlippage > 100 || _x28ToBonFireSlippage < 2) revert InvalidInput();
    if (_titanXToTincSlippage > 100 || _titanXToTincSlippage < 2) revert InvalidInput();
    if (_titanXToDragonXSlippage > 100 || _titanXToDragonXSlippage < 2) revert InvalidInput();
    if (_dragonXToMorphSlippage > 100 || _dragonXToMorphSlippage < 2) revert InvalidInput();

    titanXToX28Slippage = _titanXToX28Slippage;
    x28ToBonFireSlippage = _x28ToBonFireSlippage;
    titanXToTincSlippage = _titanXToTincSlippage;
    titanXToDragonXSlippage = _titanXToDragonXSlippage;
    dragonXToMorphSlippage = _dragonXToMorphSlippage;
}
```

BonFireLpAdderTest Info

Token Name
unavailable

Symbol
unavailable

Contract Address
0x2D6BD38707c55d5C503841E5138dB1e3A777bB38

Network
ETH - Sepolia testnet

Language
Solidity

Deployment Date
Dec 18, 2024

Contract Type
Liquidity adder

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.

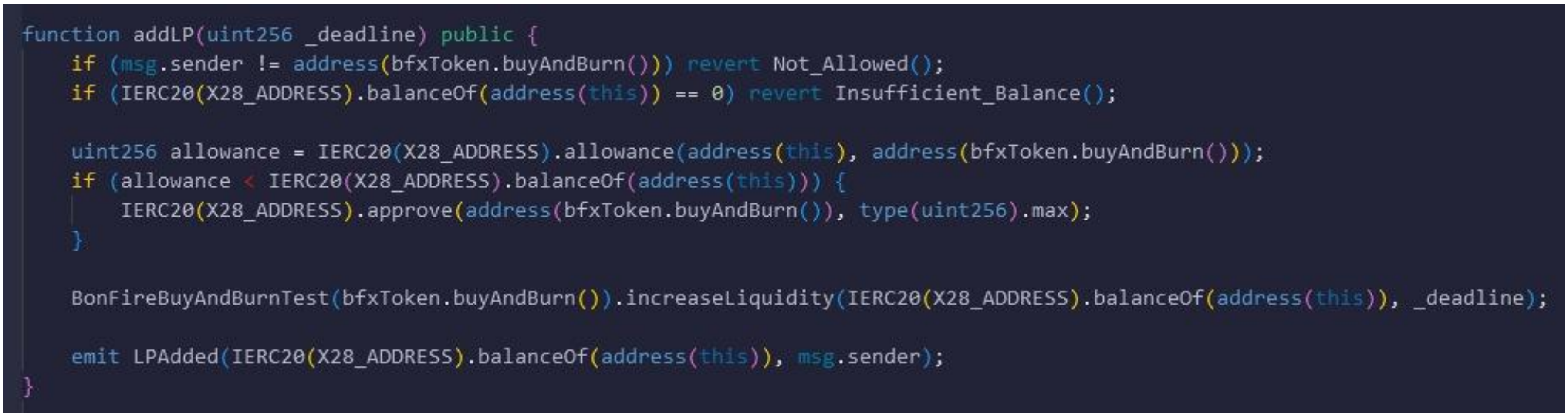
⚠ Low Risk

No low risk-level threats found in this contract.

FOUND THREATS

Informational

Only buyAndBurn address can initiate addLP.



```
function addLP(uint256 _deadline) public {
    if (msg.sender != address(bfxToken.buyAndBurn())) revert Not_Allowed();
    if (IERC20(X28_ADDRESS).balanceOf(address(this)) == 0) revert Insufficient_Balance();

    uint256 allowance = IERC20(X28_ADDRESS).allowance(address(this), address(bfxToken.buyAndBurn()));
    if (allowance < IERC20(X28_ADDRESS).balanceOf(address(this))) {
        IERC20(X28_ADDRESS).approve(address(bfxToken.buyAndBurn()), type(uint256).max);
    }

    BonFireBuyAndBurnTest(bfxToken.buyAndBurn()).increaseLiquidity(IERC20(X28_ADDRESS).balanceOf(address(this)), _deadline);

    emit LPAdded(IERC20(X28_ADDRESS).balanceOf(address(this)), msg.sender);
}
```

BonFireUiHelper Info

Token Name unavailable	Symbol unavailable
Contract Address 0x7BDb8e14048ce1389910001EAda9427471C79ca4	
Network ETH - Sepolia testnet	Language Solidity
Deployment Date unavailable	Contract Type UI Helper
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.

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.