



# SPADE SOLIDITY AUDITS

Lil Floki Audit

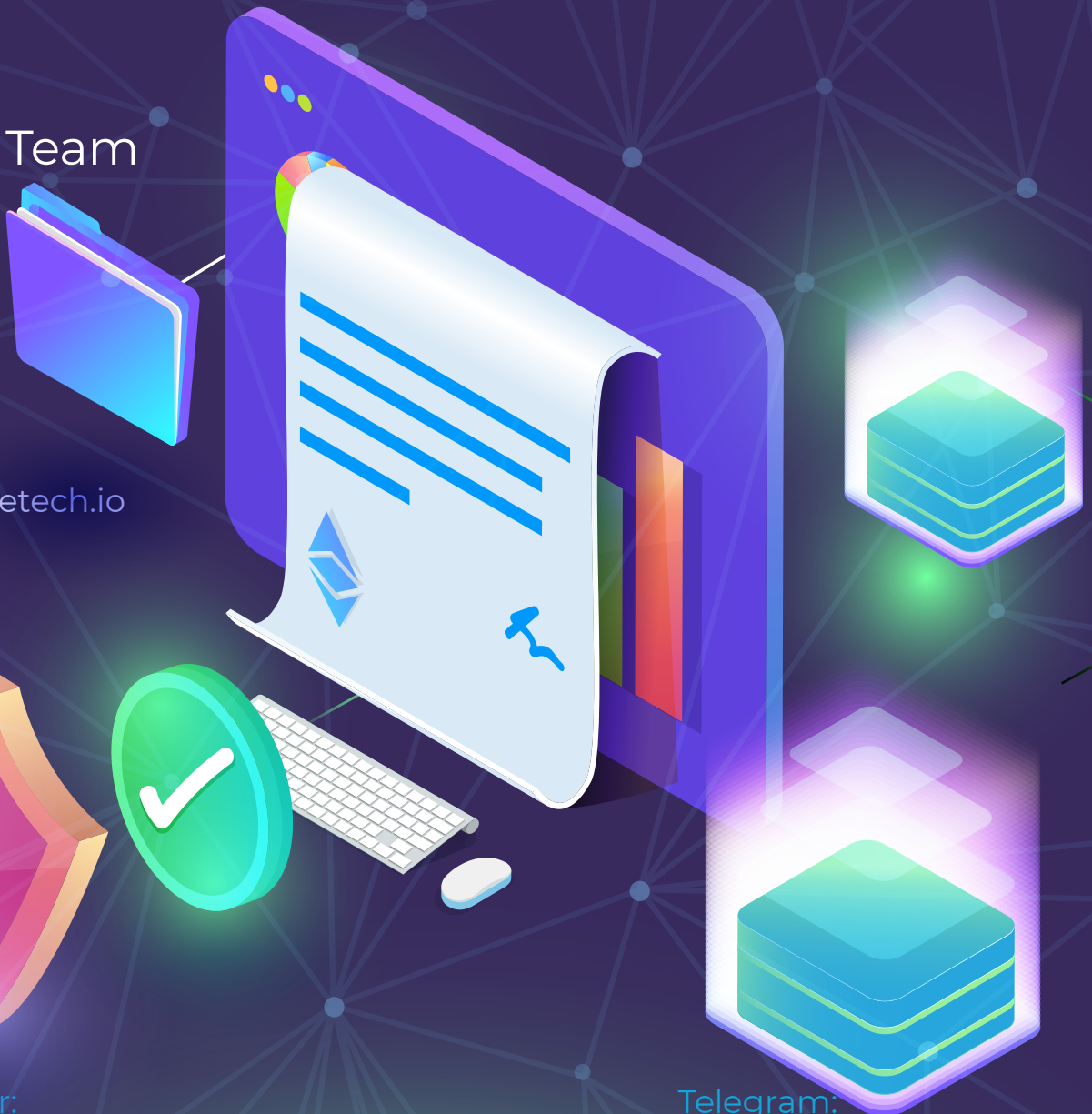
October 29, 2021

For :  
Lil Floki Team

Website:  
<https://spadetech.io>

Twitter:  
[@SpadeAudits](https://twitter.com/SpadeAudits)

Telegram:  
[t.me/spadeaudits](https://t.me/spadeaudits)





# Disclaimer

Spade Solidity reports are not, nor should be considered, an “endorsement” or “disapproval” of any particular project or team. These reports are not, nor should be considered, an indication of the economics or value of any “product” or “asset” created by any team or project that contracts Spade to perform a security review.

Spade Solidity Reports do not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors, business, business model or legal compliance.

Spade Solidity Reports should not be used in any way to make decisions around investment or involvement with any particular project. These reports in no way provide investment advice, nor should be leveraged as investment advice of any sort.

Spade Solidity Reports represent an extensive auditing process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.

Blockchain technology and cryptographic assets present a high level of ongoing risk. Spade Solidity’s position is that each company and individual are responsible for their own due diligence and continuous security. Spade Solidity’s goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies, and in no way claims any guarantee of security or functionality of the technology we agree to analyze.

What is a Spade Solidity report?

- A document describing in detail an in depth analysis of a particular piece(s) of source code provided to Spade Solidity by a Client.
- An organized collection of testing results, analysis and inferences made about the structure, implementation and overall best practices of a particular piece of source code.
- Representation that a Client of Spade Solidity has indeed completed a round of auditing with the intention to increase the quality of the company/ product’s IT infrastructure and or source code.

# OVERVIEW

## Project Summary

Project Name	Lil Floki Audit
Description	Meme Token
Platform	Binance Smart Chain
Codebase	Received file
Commit	NA

## Audit Summary

Delivery Date	October 29, 2021
Method of Audit	Static Analysis, Manual Review
Timeline	Story Points - 32

## Vulnerability Summary

Total Issues	5
Total Critical	0
Total High	0
Total Medium	1
Total Low	2
Total Informational	2

# Executive Summary

Lil Floki is the latest token to envelope the BSC space which combines Elon Musk's influential Shiba Inu Puppy called Floki while also rewarding Lil Floki holders with \$BNB Reward pay-outs.

This strategic decision to call the project Lil Floki was made as Elon Musk, being the trendsetter that he is, every time he would tweet about this dog it would create more hype around the Lil Floki token for our investors.

Tokenomics: 13% Buy/Sell Tax: 1% BNB Rewards to all holders, 5% to Liquidity & 7% to Marketing/Charity Wallet


Our detailed audit methodology was as follows:

Step 1
A manual line-by-line code review to ensure the logic behind each function is sound and safe from common attack vectors.
Step 2
Simulation of hundreds of thousands of Smart Contract Interactions on a test blockchain using a combination of automated test tools and manual testing to determine if any security vulnerabilities exist.
Step 3
Consultation with the project team on the audit report pre-publication to implement recommendations and resolve any outstanding issues.



# Grading

The following grading structure was used to assess the level of vulnerability found within Padswap Smart Contracts:



Threat Level	Definition
Critical	Severe vulnerabilities which compromise the entire protocol and could result in immediate data manipulation or asset loss.
High	Significant vulnerabilities which compromise the functioning of the smart contracts leading to possible data manipulation or asset loss.
Medium	Vulnerabilities which if not fixed within in a set timescale could compromise the functioning of the smart contracts leading to possible data manipulation or asset loss.
Low	Low level vulnerabilities which may or may not have an impact on the optimal performance of the Smart contract.
Informational	Issues related to coding best practice which do not have any impact on the functionality of the Smart Contracts.



# Lil Floki Contract on BSC

<https://bscscan.com/address/0x3271d12d5ba36b6582fafa029598fe0f5f6db35#code>

ID	TITLE	SERVERITY
FLO-001	Volatile code	Informational
FLO-002	Volatile code	Informational
FLO-003	Volatile code	Low
FLO-004	Volatile code	Medium
FLO-005	Volatile code	Low



FLO-001

TYPE	SERVERITY	LOCATION
Volatile Code	Informational	LilFloki.sol

### Description:

line 164 -170 function should emit an event. (informational),

```
function setFee(uint256 _BNBRewardsFee, uint256 _liquidityFee,
uint256 _marketingFee) public onlyOwner {
    BNBRewardsFee = _BNBRewardsFee;
    liquidityFee = _liquidityFee;
    marketingFee = _marketingFee;

    totalFees = BNBRewardsFee.add(liquidityFee).add
(marketingFee); // total fee transfer and buy
}
```

LilFloki.sol

### Suggestion:

function setFee() should emit an event to let the users knows whenever fees is updated,

---

### Resolution status:

Issue Acknowledged by Lil Floki Team.



FLO-002

TYPE	SERVERITY	LOCATION
Volatile code	Informational	LilFloki.sol

### Description:

line 137 : needs to re-exclude AMMs when updating DividendTracker.  
(informational)

```
function updateDividendTracker(address newAddress) public only  
Owner {
```

LilFloki.sol

### Suggestion:

function updateDividendTracker() does not exclude already excluded automated market maker pair for getting dividend. owner have to exclude those automated market maker pairs again.

---

### Resolution status:

Issue Acknowledged by Lil Floki Team.



TYPE	SERVERITY	LOCATION
Volatile code	Low	LilFloki.sol

**Description:**

line 349 -369 : Mutliple times swapTokensForEth() called in a transaction.(low)

```
if(
    tradingIsEnabled &&
    canSwap &&
    !swapping &&
    !automatedMarketMakerPairs[from] &&
    from != liquidityWallet &&
    to != liquidityWallet
) {
    swapping = true;
    uint256 swapTokens = contractTokenBalance.mul(liquidityFee).
div(totalFees);
    swapAndLiquify(swapTokens);
    uint256 marketingTokens = contractTokenBalance.mul
(marketingFee).div(totalFees);
    sendBNBToMarketing(marketingTokens);

    uint256 sellTokens = balanceOf(address(this));
    swapAndSendDividends(sellTokens);
    swapping = false;
}
```

LilFloki.sol

**Suggestion:**

swapAndLiquify(), sendBNBToMarketing(), swapAndSendDividends(), calls swapTokensForEth() independently resulting in high transaction gas fees.

---

**Resolution status:**

Issue Acknowledged by Lil Floki Team.



FLO-004

TYPE	SERVERITY	LOCATION
Volatile Code	Medium	LilFloki.sol

### Description:

line 453-468: Auto LP not locked.(informational)

```
function addLiquidity(uint256 tokenAmount, uint256 ethAmount)
private {

    // approve token transfer to cover all possible scenarios
    _approve(address(this), address(uniswapV2Router),
tokenAmount);

    // add the liquidity
    uniswapV2Router.addLiquidityETH{value: ethAmount}(
        address(this),
        tokenAmount,
        0, // slippage is unavoidable
        0, // slippage is unavoidable
        liquidityWallet,
        block.timestamp
    );
}
```

LilFloki.sol

### Suggestion:

auto LP tokens are sent to liquidity wallet, hence liquidity wallet owner can drain liquidity whenever he wishes.

---

### Resolution status:

Issue Acknowledged by Lil Floki Team.



TYPE	SERVERITY	LOCATION
Volatile code	Low	LilFloki.sol

### Description:

line 334,341 : other DEX pair contract should also be exempt from max wallet amount and maxSellTransactionAmount (remove liquidity condition).(medium/low)

```
        from != address(uniswapV2Router) && //router -> pair is removing
liquidity which shouldn't have max
        !_isExcludedFromFees[to] //no max for those excluded from fees
    ) {
        require(amount <= maxSellTransactionAmount, "Sell transfer
amount exceeds the maxSellTransactionAmount.");
    }

    if (from != address(this) && to != address(this)) {
        if (to != uniswapV2Pair
    )
```

LilFloki.sol

### Suggestion:

condition should be modified to support others decentralized exchanges also.

---

### Resolution status:

Issue Acknowledged by Lil Floki Team.



## Conclusion

The token contract developed by the Lil Floki team is well written and the team appear to have carefully considered the security and functionality of their code. The Spade Audit found no critical issues . All other issues have been acknowledged by the Lil Floki team. Investors should also be aware that the Lil Floki team have privileged access to all auto liquidity generated by the contract but they have committed to locking this manually for 1 year as it accumulates.

The Spade Audit team can never give absolute guarantees about the security and functionality of any system or protocol, but our detailed review of the Lil Floki Bep 20 token contract found that it functions as intended. No serious security issues were identified.

# Appendix

## Finding Categories

### Gas Optimization

Gas Optimization findings refer to exhibits that do not affect the functionality of the code but generate different, more optimal EVM opcodes resulting in a reduction on the total gas cost of a transaction.

### Mathematical Operations

Mathematical Operation exhibits entail findings that relate to mishandling of math formulas, such as overflows, incorrect operations etc.

### Logical Issue

Logical Issue findings are exhibits that detail a fault in the logic of the linked code, such as an incorrect notion on how `block.timestamp` works.

### Control Flow

Control Flow findings concern the access control imposed on functions, such as owner-only functions being invoke-able by anyone under certain circumstances.

### Volatile Code

Volatile Code findings refer to segments of code that behave unexpectedly on certain edge cases that may result in avulnerability.

### Data Flow

Data Flow findings describe faults in the way data is handled at rest and in memory, such as the result of a `structassignment` operation affecting an in-memory struct rather than an `instorage` one.

### Language Specific

Language Specific findings are issues that would only arise within Solidity, i.e. incorrect usage of `private` or `delete` .

### Coding Style

Coding Style findings usually do not affect the generated byte-code and comment on how to make the codebase more legible and as a result easily maintainable.

### Inconsistency

Inconsistency findings refer to functions that should seemingly behave similarly yet contain different code, such as a constructor assignment imposing different require statements on the input variables than a setter function.

### Magic Numbers

Magic Number findings refer to numeric literals that are expressed in the codebase in their raw format and should otherwise be specified as constant contract variables aiding in their legibility and maintainability.

### Compiler Error

Compiler Error findings refer to an error in the structure of the code that renders it impossible to compile using the specified version of the project.

### Dead Code

Code that otherwise does not affect the functionality of the codebase and can be safely omitted.

Twitter:  
[@SpadeAudits](https://twitter.com/SpadeAudits)

Website:  
<https://spadetech.io>

Telegram:  
[t.me/spadeaudits](https://t.me/spadeaudits)