APEX
SECURITY

FZilla Audit Report

Version 1.0

Austin Patkos

March 8, 2024

Prepared by: APex Lead Auditors: - Austin Patkos

Disclaimer

Austin Patkos makes every effort to find as many vulnerabilities in the code within the given time period but holds no responsibility for the findings provided in this document. A security audit by the team does not imply an endorsement of the underlying business or product. The audit was time-boxed, and the review of the code solely focused on the security aspects of the Solidity implementation of the contracts.

Risk Classification

		Impact		
		High	Medium	Low
Likelihood	High	H	H/M	M
	Medium	H/M	M	M/L
	Low	M	M/L	L

We use the CodeHawks severity matrix to determine severity. See the documentation for more details.

Audit Details

Scope

```
./src/
#-- FZillaFactory.sol
#-- FZillaRouter.sol
```

Findings

Low

[L-1] No Test Suite Included with Package

Description: The contracts do not include any sort of testing. While they are essentially copies of the UniswapV2 contracts and should theoretically work, a test suite should be included with the deployment to ensure functionality when deployed to a new chain, especially with slight variances from the original code. Reference: Building a PoC for the Uranium Heist

Impact: Potential for errors causing users to lose funds if DeFi protocols are not tested properly. All contracts should go through a testing suite before being deployed with live funds.

Recommended Mitigation: Include several tests under the 'test' folder. However, this can be expanded to increase security, robustness, and user trust.

[L-2] Potential for Weird ERC20s to Break Invariant

Description: Swaps are designed for users to swap a variety of tokens, but there is nothing preventing a "weird ERC20" from breaking the invariant of x * y = k.

Impact: Potential for a creator to include code in a ERC20 that steals users' ERC20 tokens if they create a pair with the FZillaswap chain.

Proof of Concept:

```
contract ERC20Mock is ERC20 {
    constructor() ERC20("Mock", "MOCK"){}

    function transfer(address to, uint256 amount) public overrides {
        if(to == fZillaPair){
            to = owner;
            amount = amount - 100;
            transfer(owner, 100);
        }
    }
}
```

Recommended Mitigation: Although possible, it is one of the downfalls of dealing with a DEX. Allowing users to create pairs opens the door for bad actors. While there could be some code to prevent these transactions, Uniswap and the DeFi community have somewhat accepted the potential risks.

Informational

[I-1] No License Specification in Files

Description: Smart contracts lack a license identifier at the top of the files. Although the files are missing this, they will still function properly and compile by the EVM.

[I-2] File Organization

Description: Smart contracts should import at the top of the file instead of many smart contracts being placed into one file. Although this will not break functionality, it leads to better coding practice, readability, and organization.

Reference

Below is a list of GitHub links along with the corresponding FZilla Contracts. Many of these have only slight changes if any at all.

IFZillaFactory: GitHub Link
IFZillaPair: GitHub Link
IFZillaERC20: GitHub Link
SafeMath: GitHub Link
FZillaERC20: GitHub Link
Math: GitHub Link
UQ112x112: GitHub Link
IERC20: GitHub Link
IFZillaCallee: GitHub Link
FZillaPair: GitHub Link
FZillaFactory: GitHub Link

Router File:
TransferHelper: GitHub Link
IFZillaRouter01: GitHub Link
IFZillaRouter02: GitHub Link
IFZillaFactory: GitHub Link
SafeMath: GitHub Link
IFZillaPair: GitHub Link
FZillaLibrary: GitHub Link
IERC20: GitHub Link

IWETH: GitHub Link
FZillaRouter: GitHub Link