DFlow Protocol

Security Review

Cantina Managed review by:
Kurt Barry, Lead Security Researcher
RustyRabbit, Security Researcher

December 1, 2023

Contents

1 Introduction

1.1 About Cantina

Cantina is a security services marketplace that connects top security researchers and solutions with clients. Learn more at cantina.xyz

1.2 Disclaimer

Cantina Managed provides a detailed evaluation of the security posture of the code at a particular moment based on the information available at the time of the review. While Cantina Managed endeavors to identify and disclose all potential security issues, it cannot guarantee that every vulnerability will be detected or that the code will be entirely secure against all possible attacks. The assessment is conducted based on the specific commit and version of the code provided. Any subsequent modifications to the code may introduce new vulnerabilities that were absent during the initial review. Therefore, any changes made to the code require a new security review to ensure that the code remains secure. Please be advised that the Cantina Managed security review is not a replacement for continuous security measures such as penetration testing, vulnerability scanning, and regular code reviews.

1.3 Risk assessment

Severity	Description
Critical	*Must* fix as soon as possible (if already deployed).
High	Leads to a loss of a significant portion (>10%) of assets in the protocol, or significant harm to a majority of users.
Medium	Global losses <10% or losses to only a subset of users, but still unacceptable.
Low	Losses will be annoying but bearable. Applies to things like griefing attacks that can be easily repaired or even gas inefficiencies.
Gas Optimization	Suggestions around gas saving practices.
Informational	Suggestions around best practices or readability.

1.3.1 Severity Classification

The severity of security issues found during the security review is categorized based on the above table. Critical findings have a high likelihood of being exploited and must be addressed immediately. High findings are almost certain to occur, easy to perform, or not easy but highly incentivized thus must be fixed as soon as possible.

Medium findings are conditionally possible or incentivized but are still relatively likely to occur and should be addressed. Low findings a rare combination of circumstances to exploit, or offer little to no incentive to exploit but are recommended to be addressed.

Lastly, some findings might represent objective improvements that should be addressed but do not impact the project's overall security (Gas and Informational findings).

2 Security Review Summary

DFlow is a decentralized marketplace that facilitates a market-driven price discovery process for order flow between wallets and a network of institutional market makers. Wallets route order flow to market makers via permissionless order flow auctions (OFAs). In return, wallets receive monetary compensation in USDC and guarantee that all customer orders are executed at the best market prices.

From Oct 30th to Nov 8th the Cantina team conducted a review of the DFlowSwap contract in the evm-contracts repository on commit hash 1d5402fe8abb60912bbe403d496993e15422b1fb.

The SHA-256 checksum of the `DFlowSwap.sol` file on this commit is:
`e6b7fc285d24e8417f6bb6a1a4bf5f3e25bb93a6e5cd9c66edeb14037996f8d3`.

The team identified a total of **4** issues in the following risk categories:

- Critical Risk: 1
- High Risk: 0
- Medium Risk: 0
- Low Risk: 0
- Gas Optimizations: 0
- Informational: 3

3 Findings

3.1 Critical Risk

3.1.1 `callTakerTokenAndFillOrder()` allow arbitrary calls to any contract enabling approved funds to be stolen

Severity: Critical Risk

Context: DFlowSwap.sol#L1875#L123

Description: `callTakerTokenAndFillOrder()` allows anyone to call any function an any contract:

```
function callTakerTokenAndFillOrder(
    IERC20 takerToken,
    bytes calldata takerTokenCalldata,
    bytes calldata fillOrderCalldata
) external {
    (bool success,) = address(takerToken).call(takerTokenCalldata);
    // ...
```

An attacker can use this to steal funds for users that have set an approval on ERC20 tokens for DFlowSwap (e.g. makers).

Recommendation: Limit the allowed contracts and functions to what is essential for the intended use case. For flexibility a contract address/function signature/bool mapping can be used to indicated allowed functions on allowed contracts.

DFlow: Fixed in commit d09c618d.

3.2 Informational

3.2.1 Lack of Support for Account Abstraction

Severity: Informational

Context: DFlowSwap.sol#L235

Description: The implemented permit logic only supports ECDSA-based permits, which assume the current account model of base-layer Ethereum. However, there has been a long push in the direction of account abstraction for Ethereum, which would allow different signing methods to be used, e.g. via custom signature verification smart contracts. Account abstraction is an official part of the Ethereum roadmap, and L2s or other chains may implement it sooner. Supporting a more general `permit` interface may "future-proof" DFlow against shifts in preferred signing methods.

Recommendation: No action is necessary, but consider supporting tokens that implement a

```
function permit(
    address owner,
    address spender,
    uint256 value,
    uint256 deadline,
    bytes memory signature
)
```

signature for `permit` in `_consumePermitForTaker`, as this will be compatible with tokens compliant with EIP-1271, for example the non-mainnet DAI implementation.

DFlow: Fixed in commit 35bd1183.

3.2.2 Tracking signature nonces based on `tx.origin` will be problematical for account abstraction

Severity: Informational

Context: DFlowSwap.sol#L336-L337

Description: The order signature nonces are tracked based on the `tx.origin` of the taker. With account abstraction the `tx.origin`may not be know beforehand when the maker signs the order which includes the nonce.

Recommendation: Consider tracking the nonce based on the maker's address rather than the tx.origin when filling the order which is assumed to be the taker.

DFlow: Fixed in commit 18dcb62e.

3.2.3 Receive function accepts ETH from more sources than strictly necessary

Severity: Informational

Context: DFlowSwap.sol#L1320

Description: The `receive()` function is only needed to receive ETH from the WETH contract during withdrawals. It does however allow any contract or EOA to send ETH to it, which leaves the funds stuck in the contract.

```
receive() external payable {}
```

Recommendation: Although this falls under user error it might by prudent to only allow the `receive()` function to accept ETH when it comes from the WETH contract.

DFlow: Given that this falls under user error, we'll prioritize the gas savings over the additional `receive()` logic here.

4 Additional Comments

The Cantina team reviewed DFlow's changes holistically on commit hash 95db9cb7c17707f0495dcd98b42bf0369af5ab3f and determined that all issues except for **3.2.3 Receive function accepts ETH from more sources than strictly necessary** (informational issue) were resolved and no new issues were introduced.

The SHA-256 checksum of the `DFlowSwap.sol` file on this commit is:
`8a3c5e16127514c6b049e988a36c46b3325ea8c74fd4fe71b64734ea2585d1c0`.