Paintswap Brush Token
Security Review

Cantina Managed review by:

Sujith Somraaj, Security Researcher
Windhustler, Security Researcher

April 3, 2025

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

PaintSwap is an open NFT marketplace and NFT launchpad on Sonic. They are also the developers of the Web3 game Estfor Kingdom.

From Apr 1st to Apr 2nd the Cantina team conducted a review of paintswap-brush-token on commit hash 0e3f7b31. The team identified a total of **5** issues:

Issues Found

Severity	Count	Fixed	Acknowledged
Critical Risk	0	0	0
High Risk	0	0	0
Medium Risk	1	0	1
Low Risk	2	0	2
Gas Optimizations	0	0	0
Informational	2	0	2
Total	**5**	**0**	**5**

3 Findings

3.1 Medium Risk

3.1.1 Increasing outbound max message size on sonic can lead to permanent loss of user funds

Severity: Medium Risk

Context: *(No context files were provided by the reviewer)*

Description: Since phase 2 has already started in the `BrushOFTAdapter` contract, the `_credit` function being called as a part of the `lzReceive` permanently reverts. Hence, any incoming messages from the `Brush.sol` contract on Sonic to `BrushOFTAdapter.sol` on Fantom will always revert and lead to permanent funds being locked.

```
function _credit(address to, uint256 amount, uint32 /* srcEid */) internal override returns (uint256) {
    // Phase 2: Reject unlocks
    require(block.timestamp < _burnStartTimestamp, InvalidState());Fantom will revert/fail as the outbound
    ↪  message size is limited to `1` bytes, and hence, it
    // ...
}
```

By taking a look at the currently deployed Brush contract on Sonic, the outbound message to Fantom will revert/fail as the outbound message size is limited to `1` bytes. Hence, it becomes impossible to send the cross-chain transfer payload.

However, if the delegate (which is an EOA) increases this limit, the outbound message becomes active and will lead to permanent losses for the users.

Proof of Concept:

```
// SPDX-License-Identifier: UNLICENSED
pragma solidity ^0.8.13;

import {Test, console} from "forge-std/Test.sol";

interface IUln {
    struct ExecutorConfig {
        uint32 maxMessageSize;
        address executor;
    }

    function getExecutorConfig(address _oapp, uint32 _remoteEid) external view returns (ExecutorConfig memory
    ↪  rtnConfig);
}

interface IEndpoint {
    struct SetConfigParam {
        uint32 eid;
        uint32 configType;
        bytes config;
    }

    function setConfig(address _oapp, address _lib, SetConfigParam[] calldata _params) external;
}

interface IBrushAdapter {
    function burnRemainingTokens() external;

    error BurnCompleted();
}

interface IBrush {
    error NoPeer(uint32 eid);
    error LZ_MessageLib_InvalidMessageSize(uint256 actual, uint256 max);

    struct SendParam {
        uint32 dstEid; // Destination endpoint ID.
        bytes32 to; // Recipient address.
        uint256 amountLD; // Amount to send in local decimals.
        uint256 minAmountLD; // Minimum amount to send in local decimals.
        bytes extraOptions; // Additional options supplied by the caller to be used in the LayerZero message.
        bytes composeMsg; // The composed message for the send() operation.
        bytes oftCmd; // The OFT command to be executed, unused in default OFT implementations.
    }

    struct MessagingFee {
        uint256 nativeFee;
        uint256 lzTokenFee;
    }

    struct MessagingReceipt {
        bytes32 guid;
        uint64 nonce;
        MessagingFee fee;
    }

    struct OFTReceipt {
        uint256 amountSentLD; // Amount of tokens ACTUALLY debited from the sender in local decimals.
        // @dev In non-default implementations, the amountReceivedLD COULD differ from this value.
        uint256 amountReceivedLD; // Amount of tokens to be received on the remote side.
    }

    function send(
        SendParam calldata _sendParam,
        MessagingFee calldata _fee,
        address _refundAddress
    ) external payable returns (MessagingReceipt memory, OFTReceipt memory);
}

contract ForkTest is Test {
    uint256 ftmFork;
    uint256 sonicFork;

    IBrushAdapter public brushAdapter;
    IBrush public brush;
    IUln public uln;
    IEndpoint public endpoint;

    function setUp() public {
        sonicFork = vm.createSelectFork("https://rpc.soniclabs.com");
        brush = IBrush(0xE51EE9868C1f0d6cd968A8B8C8376Dc2991BFE44);
        uln = IUln(0xC39161c743D0307EB9BCc9FEF03eeb9Dc4802de7);
        endpoint = IEndpoint(0x6F475642a6e85809B1c36Fa62763669b1b48DD5B);


        ftmFork = vm.createSelectFork("https://rpc.fantom.network");
        brushAdapter = IBrushAdapter(0x9D92cD1A5Cea3147e5Bf47EfF2D2C632C9839267);
    }

    function test_get_ulnconfig() public {
        vm.selectFork(sonicFork);
        uln.getExecutorConfig(address(brush), 30112); // 30112 is fantom

        /// increase it by current delegate
        vm.startPrank(0x3F64e0E81853d8913f7e646c13Dd7B28411DCB11);
        IEndpoint.SetConfigParam[] memory param = new IEndpoint.SetConfigParam[](1);
        param[0] = IEndpoint.SetConfigParam(30112, 1, abi.encode(IUln.ExecutorConfig(10000,
        ↪  0x4208D6E27538189bB48E603D6123A94b8Abe0A0b)));

        endpoint.setConfig(address(brush), address(uln), param);
        uln.getExecutorConfig(address(brush), 30112); // 30112 is fantom

        deal(0xa801864d0D24686B15682261aa05D4e1e6e5BD94, 100 ether);
        deal(address(brush),0xa801864d0D24686B15682261aa05D4e1e6e5BD94, 1000000000000000000);

        IBrush.SendParam memory sendParam = IBrush.SendParam(
            uint32(30112),
            0x000000000000000000000000a801864d0d24686b15682261aa05d4e1e6e5bd94,
            1000000000000000000,
            1000000000000000000,
            hex"00030100110100000000000000000000000000030d40",
            "",
            ""
        );

        IBrush.MessagingFee memory feeParam = IBrush.MessagingFee(918636848344l2189, 0);

        vm.startPrank(0xa801864d0D24686B15682261aa05D4e1e6e5BD94);
        brush.send{value: feeParam.nativeFee}(
            sendParam,
            feeParam,
            0xa801864d0D24686B15682261aa05D4e1e6e5BD94
        );
    }
}
```

Likelihood explanation: The likelihood of this issue occurring is `LOW` since the delegate or app is not motivated to act in this manner unless it has been hacked or compromised.

Impact explanation: If exploited, the issue will permanently lock the bridged funds, which has a very high impact.

Recommendation: Move the delegate to a timelock contract, ensuring all changes to `peer` addresses, executor and library configurations are safe and made after a timelock.

Paintswap: While true, we'd have to re-enable it and users would have to actively send BRUSH to Fantom themselves for those funds to be lost. This issue is more wide-spread though with any bridging contracts which allows sending on a chain but not receiving on the destination, those funds could also be lost. A timelock wouldn't stop this in the long-term just delay it by a day or so. Changing the delegate to a multi-sig should provide enough enhanced security vs an EOA for this current case I think. In the future if we were to add more chains we would look into a timelock contract on each chain to give users more heads up about proposed bridge changes.

Changing delegate: 0x5279f7a62adc6bf8536f8d61091051ea2a3e6d8d.

Cantina Managed: The delegate is now a multi-sig, which is more safe than the previous approach. Still the security can be improved further by locking it behind a timelock (to give users enough time to react to the incoming update).

3.2 Low Risk

3.2.1 Permanent loss of funds if outbound message size is increased

Severity: Low Risk

Context: *(No context files were provided by the reviewer)*

Description: Since phase 2 has already started in the `BrushOFTAdapter` contract, the `_credit` function being called as a part of the `lzReceive` permanently reverts. Hence, any incoming messages from the `Brush.sol` contract on Sonic to `BrushOFTAdapter.sol` on Fantom will always revert and lead to permanent funds being locked.

- BrushOFTAdapter.sol:

```
function _credit(address to, uint256 amount, uint32 /* srcEid */) internal override returns (uint256) {
    // Phase 2: Reject unlocks
    require(block.timestamp < _burnStartTimestamp, InvalidState()); // <<<

    emit BridgeIn(to, amount);

    // Phase 1: Unlock tokens
    require(innerToken.transfer(to, amount), TokenTransferFailed());

    return amount;
}
```

By taking a look at the currently deployed Brush contract on Sonic and BrushOFTAdapter on Fantom, this issue is not present at the moment because:

- Brush contract on Sonic can't send any more messages to Fantom since the maximum message size the Executor can process was set to 1, disabling communication in that direction.
- There are no in-flight messages in both directions per Layerzero explorer. link-1 and link-2.
- `burnRemainingTokens` was executed burning all the tokens in the `BrushOFTAdapter`..

Recommendation: Make sure to highlight these limitations of the system in case these contracts are used for other deployments.

Paintswap: The adapter won't be used on other chains as Brush on Sonic has Layer Zero functionality built-in, so can use their send/receive functions directly.

Cantina Managed: Acknowledged.

3.2.2 Inconsistent zero address handling between `OFT.sol` and `Brush.sol`

Severity: Low Risk

Context: Brush.sol#L37

Description: In the base `OFT.sol` contract, the `_credit` function redirects zero address credits to `0xdead`:

```
function _credit(address _to, uint256 _amountLD, uint32 /*_srcEid*/) internal virtual override returns
↪  (uint256 amountReceivedLD) {
    if (_to == address(0x0)) _to = address(0xdead); // _mint(...) does not support address(0x0)
    _mint(_to, _amountLD);
    return _amountLD;
}
```

However, implementations that override this function do not include this zero address redirection, leading to inconsistency.

Recommendation: Consider directing all zero address credits to `0xdead` in the implementation to ensure consistency with the base implementation.

Paintswap: Probably should have been done to be consistent, but I don't fully agree with minting to the dead address either. It is still burnt on the other side of the bridge, so permanent failure to deliver this message while annoying does not affect our token directly and would interfere with our deflationary model for BRUSH, which burns the tokens out of the `totalSupply()`.

Cantina Managed: Acknowledged.

3.3 Informational

3.3.1 Missing indexed parameters in events reduces filtering efficiency

Severity: Informational

Context: BrushOFTAdapter.sol#L25, BrushOFTAdapter.sol#L32

Description: The events `BridgeIn` and `BridgeOut` in the `BrushOFTAdapter.sol` contract lack the `indexed` keyword for their parameters. This makes it difficult and inefficient for off-chain services to filter for specific events based on address or amount values.

Recommendation: Add the `indexed` keyword to address parameters and optionally to significant `uint256` parameters. This will store these values as topics in the event logs, enabling efficient filtering:

```
event BridgeIn(address indexed to, uint256 amount);
event BridgeOut(address indexed from, uint256 amount);
```

Paintswap: The intention here is to index all events, and then filtering can be done off-chain using an indexer like a subgraph. We prefer to be more gas efficient on-chain and using indexed parameters costs more gas to emit the event.

Cantina Managed: Acknowledged.

3.3.2 Incorrect visibility specifier for `burnFrom` function

Severity: Informational

Context: Brush.sol#L74

Description: The `burnFrom()` function in `Brush.sol` contract is declared as `public` but is never called internally within the contract.

```
function burnFrom(address account, uint256 amount) public {
    // ...
}
```

Recommendation: Change the visibility modifier from public to external:

```diff
- function burnFrom(address account, uint256 amount) public {
+ function burnFrom(address account, uint256 amount) external {
```

Paintswap: Yes it should have been. But the deployed bytecode looks to be the same with both.

Cantina Managed: Acknowledged.