ERC Burner
Security Review

Cantina Managed review by:

High Byte, Security Researcher
Chinmay Farkya, Associate Security Researcher

April 30, 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

ERC Burner is a tool to convert small assets to native currency on EVM compatible chains.

From Mar 28th to Apr 4th the Cantina team conducted a review of ercburner-audit on commit hash 67614a43. The team identified a total of **12** issues:

Issues Found

Severity	Count	Fixed	Acknowledged
Critical Risk	0	0	0
High Risk	0	0	0
Medium Risk	2	2	0
Low Risk	5	5	0
Gas Optimizations	1	1	0
Informational	4	4	0
Total	**12**	**12**	**0**

3 Findings

3.1 Medium Risk

3.1.1 Swap execution can completely fail in some cases when tokens like BNB are involved

Severity: Medium Risk

Context: URBurner.sol#L291-L301

Description: `swapExactInputMultiple()` is supposed to handle a batch of swap operations, all leading to WNATIVE ⇒ which can then be unwrapped and bridged as ETH / native token. Current protocol logic has means to handle the whole swap transaction in case any of the constituent swap operation fails, which allows the remaining transaction to go through normally.

This is done by wrapping the swap call to the router in a try-catch block, with the catch block handling token approval reset and refunds. But this catch block does not work correctly for some ERC20 tokens like BNB on Ethereum:

```
catch {
    // If the swap fails, decrease the allowance of the permit2 contract.
    token.safeDecreaseAllowance(address(permit2), amountIn);
    // Return the tokens to the sender.
    token.safeTransfer(msg.sender, amountIn);

    emit BurnerEvents.SwapFailed(msg.sender, param.tokenIn, amountIn, "Router error");

    unchecked { ++i; }
    continue;
}
```

Notably, this `safeDecreaseAllowance()` call is from Openzeppelin's SafeERC20 library. The call flow leads to `forceApprove(0)`, where `token.approve(0)` is called. The `approve(0)` call needs to succeed for this `forceApprove()` call to go through. This is exactly where tokens like BNB misbehave : they revert on an `approve(0)` call.

As a result, the `safeDecreaseAllowance()` call will fail for such tokens, with the revert bubbling up : causing the whole transaction to revert since this operation is inside a catch block.

Recommendation: Consider wrapping this `safeDecreaseAllowance()` call in a try-catch block, or document that certain tokens are not supported and including them in the swaps could lead to the failure of the whole batch-swap transaction.

ERC Burner: Fixed in PR 1.

Cantina Managed: Fix verified.

3.1.2 Bridge refunds could be lost if `URBurner` is not deployed at the same address on all supported chains

Severity: Medium Risk

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

Description: ERCBurner uses relay protocol for bridging ETH from one chain to another. There is no validation of the `bridgeData` in `swapExactInputMultiple()` or `relayBridge()` functions: the data is passed as it is supplied by the user. Utmost care is needed to handle bridge refunds in case bridge operation fails.

According to the relay protocol docs, in some cases {notably, when the `refundTo` and `recipient` addresses are both unspecified in `bridgeData`}, in case of the bridge operation failing ⇒ the refund will be sent to the caller address on the destination chain.

For this protocol's case, this might happen:

- `URBurner` is deployed at address X on Ethereum Mainnet.
- User bridges ETH to Arbitrum.
- The bridge operation fails.

- There was no `refundTo` address specified and no `recipient` address specified, so the refund will go to the caller on destination chain: which means address X on Arbitrum.
- But this address would not be the `URBurner` if `URBurner` is not deployed at the same address on all chains.

This means the refunded ETH will not be recoverable and will lead to loss of funds.

Recommendation: If `URBurner` is the same address on source and destination chains, it could receive those ETH refunds and then the admin could keep track of bridge failures, rescuing and redistributing ETH to original owners. This way there would be no loss of funds.

It is recommended to try to deploy `URBurner` at the same address on all supported chains.

ERC Burner: Fixed in commit c6e7d2fd.

Cantina Managed: Fix verified.

3.2 Low Risk

3.2.1 `relayBridge()` does not check if bridging functionality has been paused

Severity: Low Risk

Context: AVAXBurner.sol#L345-L354, URBurner.sol#L362-L370

Description: There are two functions available to help with bridging funds:

- `swapExactInputMultiple()` if user wants to swap ERC20 tokens to ETH before bridging it.
- `relayBridge()` if user already has ETH and wants to just bridge it.

`swapExactInputMultiple()` has correct checks to not allow bridging when `pauseBridge` boolean has been set to true ⇒ meaning that the bridging functionality has been paused by the admin. But the same checks are missing in `relayBridge()` ⇒ which implies that users could use ERCBurner for bridging even when it should not be allowed as per the contract state.

Recommendation: Add the following check to `relayBridge()` function in URburner and AvaxBurner contracts:

```
if (pauseBridge) revert BurnerErrors.BridgePaused();
```

ERC Burner: Fixed in commit 2a85814c.

Cantina Managed: Fix verified.

3.2.2 Users can lose funds if they swap to tokens other than `WNATIVE` via `ERCBurner`

Severity: Low Risk

Context: URBurner.sol#L279-L290

Description: In `swapExactInputMultiple()`, accounting is done by calculating the amount of WNATIVE tokens received as a result of each swap execution. These amounts are then all added up together to find total WNATIVE funds received from the entire batch of swaps. But the tokenOut is never validated, all this logic assumes that the tokenOut is always encoded as WNATIVE, which may not be true.

The current accounting logic will miss out on any token balances {output from the swaps} that are not of WNATIVE : which means if any user encodes swaps with output token == any token X other than `WNATIVE`, then they will lose all those funds received by `URBurner` as an output from the swap.

Though there is a rescue function which could be used by admin to manually re-distribute these tokens to original owners later, but it will be difficult to track these swaps.

Recommendation: It should be programmatically enforced in `_validateAndDecodeSwapParams()` that the output token is always WNATIVE, to prevent users' losses.

ERC Burner: Fixed in PR 1.

Cantina Managed: Fix verified.

3.2.3 `DEFAULT_ADMIN_ROLE` needs to be re-assigned properly when transferring ownership

Severity: Low Risk

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

Description: According to the ERCBurner specs, the owner of URBurner contract is supposed to have the `DEFAULT_ADMIN_ROLE`. The owner address is correctly assigned with the role when initializing. But in case ownership is transferred, the role setup is not automatically revised.

This can lead to the following problems:

- Ownership gets transferred and the previous admin address gets burned ⇒ leading to loss of access to `DEFAULT_ADMIN_ROLE`.
- Ownership gets transferred, new owner does not get `DEFAULT_ADMIN_ROLE` automatically but the old address retains the role. If the old address was hacked, this will compromise this role.

When ownership is transferred, the role setup should be revised automatically to prevent errors.

Recommendation: Override the `transferOwnership()` function and add these lines:

```
_revokeRole(DEFAULT_ADMIN_ROLE, owner);
_grantRole(DEFAULT_ADMIN_ROLE, newOwner);
```

Also consider using `Ownable2StepUpgradeable` in place of `OwnableUpgradeable` to enable two-step process for modifying ownership. In that case, `acceptOwnership()` should be modified to grant and revoke roles.

Note that this role setup can also be revised manually before transferring ownership, but its better to do it programmatically to prevent human error.

ERC Burner: Fixed in commit af6ed48f.

Cantina Managed: Fix verified.

3.2.4 Users could lose referral registration fees if they register after referrals are paused

Severity: Low Risk

Context: URBurner.sol#L393-L396, URBurner.sol#L424-L427

Description: Whether referral system is active and the referrers will earn any fees is governed by the `pauseReferral` boolean. This is reflected in the `_calculateReferrerFee()` code. If `pauseReferral` is true, then even though the swap/ bridge executions go through, no referrer fees is applied to the transaction.

But in `paidReferrer()` and `upgradeReferrer()`, new users are not prevented from registering if the referrals are already paused, which can make them lose their USDC in return for nothing, especially because they do not know how indefinitely the referrals are going to remain paused.

Suppose they register as a referrer thinking they will get some yield, but it instead causes opportunity cost and in the worst case loss of their USDC if the referrals are never turned back on.

Recommendation: `paidReferrer()` and `upgradeReferrer()` should revert if `pauseReferral` is true, to prevent users from locking their USDC for nothing.Additionally, there should be clear communication from the team about when referrals are going to be paused in future, to provide a clear picture to participants on when not to enter as a referrer.

ERC Burner: Fixed in commit 2a85814c.

Cantina Managed: Fix verified.

3.2.5 Bridging fee is incorrectly charged in `swapExactInputMultiple()` when `bridge` boolean is set to `false`

Severity: Low Risk

Context: URBurner.sol#L322-L327

Description: In `swapExactInputMultiple()`, there are many possible combinations of actions that the user can choose.

One of these actions is ⇒

- Swap tokens to WETH.
- Also send along native currency with the call.
- Bundle the swap output with the native currency sent.
- And transfer it all to the `_to` address.

For this use case, `bridge` boolean is set to false, msg.value > 0, and user provides a recipient `_to` address. The problem with the current logic is that bridge fees is charged on the msg.value, which means any native currency sent with the call will be charged with this fees. Even if the recipient address == msg.sender, bridge fees is still charged. As discussed with the ERCBurner team, this is unintended in the specific set of conditions mentioned above. The bridge Fee has to be charged only when the native currency is supposed to be sent to a different recipient in the use case mentioned.

Recommendation: Do not charge bridge fees while sending native currency via `swapExactInputMultiple()`, when bridge is set to false and `recipient == msg.sender`.

ERC Burner: Fixed in commit 0a191abc.

Cantina Managed: Fixed. Now, the recipient is not allowed to be the same as msg.sender, in the specific set of conditions mentioned in the issue.

3.3 Gas Optimization

3.3.1 Remove `nonReentrant` from functions that have no external calls

Severity: Gas Optimization

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

Description/Recommendation: Remove `nonReentrant` from functions that have no external calls, as they cannot lead to reentrancy.

ERC Burner: Fixed in commit 2a85814c.

Cantina Managed: Fix verified.

3.4 Informational

3.4.1 Prefer explicit calls over arbitrary calldata

Severity: Informational

Context: URBurner.sol#L344

Description: Instead of calling bridge address with arbitrary data, reduce attack surface by being more explicit. For example, restrict to specific selectors:

```
bytes4 selector = bytes4(bridgeData[0:4]);
require(selector == Bridge.func1 || selector == Bridge.func1);
```

or call directly which is much cleaner:

```
IRelayReceiver(bridgeAddress).forward{value: amountAfterFee}(bridgeData)
```

ERC Burner: Fixed in commit 2a85814c.

Cantina Managed: Fix verified.

3.4.2 `paidReferrer()` and `upgradeReferrer()` should have `whenNotPaused` modifier

Severity: Informational

Context: URBurner.sol#L393-L396, URBurner.sol#L424-L427

Description: There are 3 ways to register/ upgrade a referrer:

- `putPartner()` ⇒ called by the admin to grant a referrer status to an address.
- `paidReferrer()` ⇒ allows anyone to register as a referrer to earn referral fees.

- `upgradeReferrer()` ⇒ allows an existing referrer to upgrade their referral fee tier.

While `putPartner()` is guarded by a `whenNotPaused` modifier, the other two functions do not have it.

Recommendation: Use `whenNotPaused` modifier on `paidReferrer()` and `upgradeReferrer()` for consistency.

ERC Burner: Fixed in commit 2a85814c.

Cantina Managed: Fix verified.

3.4.3 Swap deadlines in current logic are not useful

Severity: Informational

Context: URBurner.sol#L237

Description: In `swapExactInputMultiple()`, each of the swap params are used one by one to swap tokens on the router. The router needs a deadline parameter, so ERCBurner logic supplies `block.timestamp + 900` as the deadline timestamp. Such a deadline is not useful, as it will be evaluated according to value of the actual time that this transaction gets included in a block, which can be any time into the future.

Recommendation: Allow the user to supply swap deadline, or set it properly in the frontend.

ERC Burner: Fixed in commit 2a85814c.

Cantina Managed: Fix verified, but I think deadline should be moved to a single param instead of duplicated inside `SwapParams` struct.

3.4.4 Suggestions to improve code readability

Severity: Informational

Context: AVAXBurner.sol#L558, URBurner.sol#L31, URBurner.sol#L192

Description: The following are suggestions to improve code readability:

- `AVAXBurner.sol` :: `setUniversalRouter()` should be renamed to `setLBRouter()` to better reflect functionality.
- At `URBurner.sol # L192` :: Error `ReferrerCannotBeSelf()` is being used, which does not correctly reflect the intention of the checks.
- `URBurner.sol # L31 : A contract that allows users to swap multiple tokens to ETH in a single transaction` should be changed to `A contract that allows users to swap multiple tokens to the native currency in a single transaction` because ERCBurner supports some chains where ETH is not necessarily the native currency.

Recommendation: Apply the changes as recommended.

ERC Burner: Fixed in PR 1.

Cantina Managed: Fix verified.