1. ***High-Level properties:*** Solvency: ETH balance of contract >= total supply of shares – **HIGH SEVERITY**
2. ***High-Level properties:*** After withdraw(), the ETH balance of user should go up by the same amount by with the token balance has been reduced.( The burn function in this case isn’t checking for success returned by the msg.sender.call{}() based transfer. Since address.call() doesn’t throw/revert, if we don’t confirm the success, the user could end up losing tokens while not receiving an equal amount in ETH.) - **HIGH SEVERITY**
3. **msg.sender.call**{value**:** amount}("");

<address>.call(bytes memory)*returns (bool, bytes memory): issue low-level CALL with the given payload, returns success condition and return data, forwards all available gas, adjustable*

1. ***High-Level properties:*** After collectFees() the user’s ETH balance should go up by the total payout amount. Since the transfer is made using msg.sender.call() which does not revert on failure, if we don’t check for success in the return data, we’ll potentially end up deducting from the users rewards in the contract while the user doesn’t receive the corresponding ETH amount. - **HIGH SEVERITY**
2. ***Variable transition:*** totalFeesEarnedPerShare can only increase -
3. ***Variable transition:*** feesCollectedPerShare can only increase. The user could end up withdrawing more than what’s rightfully owed to them thereby creating potential solvency issues with the contract. - **HIGH SEVERITY**

In Popsicle Finance the withdraw() and collectFees() functions transfer ETH to user with address.call() low-level call. This call doesn’t revert on failure, instead it sends back success/failure as part of the return data. Since the return value is not being checked in the contract, the user might end up not receiving any ETH while still losing his tokens(shares) on the contract.

**msg.sender.call**{value**:** amount}("");

While this can be easily spotted in a manual audit, is there a way to model these possible scenarios with EVM low-level calls in prover tool. Essentially, we’d want to ensure that the user’s ETH balance goes up by the same amount that the user’s shares or rewards go down.

Are members of address types like <address>.balance usable in some way in CVL?