PoolTogether Aave v3 contest Findings & Analysis Report

2022-06-23

Table of contents

Overview

About C4

Code4rena (C4) is an open organization consisting of security researchers, auditors, developers, and individuals with domain expertise in smart contracts.

A C4 audit contest is an event in which community participants, referred to as Wardens, review, audit, or analyze smart contract logic in exchange for a bounty provided by sponsoring projects.

During the audit contest outlined in this document, C4 conducted an analysis of the PoolTogether Aave v3 smart contract system written in Solidity. The audit contest took place between April 29—May 1 2022.

Wardens

44 Wardens contributed reports to the PoolTogether Aave v3 contest:

1. lllllll
2. leastwood
3. unforgiven
4. GimelSec (rayn and sces60107)
5. 0x1f8b
6. MaratCerby
7. CertoraInc
8. berndartmueller
9. gzeon
10. kebabsec (okkothejawa and FlameHorizon)
11. 0xDjango
12. WatchPug (jtp and ming)
13. Tadashi
14. reassor
15. cccz
16. hake
17. 0xf15ers (remora and twojoy)
18. horsefacts
19. z3s
20. Dravee

21. pauliax
22. miguelmtzinf
23. pedroais
24. 0x52
25. delfin454000
26. 0x4non
27. Picodes
28. throttle
29. rotcivegaf
30. joestakey
31. fatherOfBlocks
32. TrungOre
33. tabish
34. 0v3rf10w
35. 0xkatana
36. 242
37. defsec
38. peritoflores
39. simon135
40. slywaters

This contest was judged by Justin Goro.

Final report assembled by liveactionllama.

Summary

The C4 analysis yielded an aggregated total of 6 unique vulnerabilities. Of these vulnerabilities, 1 received a risk rating in the category of HIGH severity and 5 received a risk rating in the category of MEDIUM severity.

Additionally, C4 analysis included 15 reports detailing issues with a risk rating of LOW severity or non-critical. There were also 30 reports recommending gas

optimizations.

All of the issues presented here are linked back to their original finding.

Scope

The code under review can be found within the [C4 PoolTogether Aave v3 contest repository](), and is composed of 1 smart contract written in the Solidity programming language and includes ~200 lines of Solidity code.

Severity Criteria

C4 assesses the severity of disclosed vulnerabilities according to a methodology based on [OWASP standards]().

Vulnerabilities are divided into three primary risk categories: high, medium, and low/non-critical.

High-level considerations for vulnerabilities span the following key areas when conducting assessments:

- Malicious Input Handling
- Escalation of privileges
- Arithmetic
- Gas use

Further information regarding the severity criteria referenced throughout the submission review process, please refer to the documentation provided on [the C4 website]().

High Risk Findings (1)

[H-01] A malicious early user/attacker can manipulate the vault's `pricePerShare` to take an unfair share of future users' deposits

Submitted by WatchPug, also found by 0xDjango, berndartmueller, CertoraInc, gzeon, kebabsec, leastwood, Tadashi, and unforgiven

This is a well-known attack vector for new contracts that utilize pricePerShare for accounting.

[AaveV3YieldSource.sol#L352-L374]

```
    /**
     * @notice Calculates the number of shares that should be mint
     * @param _tokens Amount of asset tokens
     * @return Number of shares.
     */
    function _tokenToShares(uint256 _tokens) internal view returns
      uint256 _supply = totalSupply();

      // shares = (tokens * totalShares) / yieldSourceATokenTotalS
      return _supply == 0 ? _tokens : _tokens.mul(_supply).div(aTo
    }

    /**
     * @notice Calculates the number of asset tokens a user has in
     * @param _shares Amount of shares
     * @return Number of asset tokens.
     */
    function _sharesToToken(uint256 _shares) internal view returns
      uint256 _supply = totalSupply();

      // tokens = (shares * yieldSourceATokenTotalSupply) / totalS
      return _supply == 0 ? _shares : _shares.mul(aToken.balanceOf
    }
```

A malicious early user can `supplyTokenTo()` with `1 wei` of `_underlyingAssetAddress` token as the first depositor of the `AaveV3YieldSource.sol`, and get `1 wei` of shares token.

Then the attacker can send `10000e18 - 1` of `aToken` and inflate the price per share from 1.0000 to an extreme value of 1.0000e22 (from `(1 + 10000e18 - 1) / 1`).

As a result, the future user who deposits `19999e18` will only receive `1 wei` (from `19999e18 * 1 / 10000e18`) of shares token.

They will immediately lose `9999e18` or half of their deposits if they `redeemToken()` right after the `supplyTokenTo()`.

[AaveV3YieldSource.sol#L251-L256]

```
function redeemToken(uint256 _redeemAmount) external override
  address _underlyingAssetAddress = _tokenAddress();
  IERC20 _assetToken = IERC20(_underlyingAssetAddress);

  uint256 _shares = _tokenToShares(_redeemAmount);
  _burn(msg.sender, _shares);
  ...
```

Furthermore, after the PPS has been inflated to an extremely high value (`10000e18`), the attacker can also redeem tokens up to `9999e18` for free, (burn `0` shares) due to the precision loss.

Recommended Mitigation Steps

Consider requiring a minimal amount of share tokens to be minted for the first minter, and send a port of the initial mints as a reserve to the DAO address so that the pricePerShare can be more resistant to manipulation.

Also, consder adding `require(_shares > 0, "AaveV3YS/shares-gt-zero");` before `_burn(msg.sender, _shares);`.

PierrickGT (PoolTogether) confirmed and commented:

> PR: https://github.com/pooltogether/aave-v3-yield-source/pull/15

Medium Risk Findings (5)

[M-01] User fund loss in `supplyTokenTo()` because of rounding

Submitted by unforgiven, also found by 0x1f8b

[AaveV3YieldSource.sol#L231-L242]
[AaveV3YieldSource.sol#L357-L362]

When user use `supplyTokenTo()` to deposit his tokens and get `share` in `FeildSource` because of rounding in division user gets lower amount of `share`. for example if token's `_decimal` was `1` and `totalSupply()` was `1000` and `aToken.balanceOf(FieldSource.address)` was `2100` (becasue of profits in `Aave Pool balance` is higher than `supply`), then if user deposit `4` token to the contract with `supplyTokenTo()`, contract is going to `mint` only `1` share for that user and if user calls `YeildToken.balanceOf(user)` the return value is going to be `2` and user already lost half of his deposit.
Of course if `_precision` was high this loss is going to be low enough to ignore but in case of low `_precision` and high price `token` and high `balance / supply` ratio this loss is going to be noticeable.

Proof of Concept

This is the code of `supplyTokenTo()`:

```
function supplyTokenTo(uint256 _depositAmount, address _to) ex
  uint256 _shares = _tokenToShares(_depositAmount);
  require(_shares > 0, "AaveV3YS/shares-gt-zero");

  address _underlyingAssetAddress = _tokenAddress();
  IERC20(_underlyingAssetAddress).safeTransferFrom(msg.sender,
  _pool().supply(_underlyingAssetAddress, _depositAmount, addr

  _mint(_to, _shares);

  emit SuppliedTokenTo(msg.sender, _shares, _depositAmount, _t
}
```

which in line: `_shares = _tokenToShares(_depositAmount)` trying to calculated `shares` corresponding to the number of tokens supplied. and then transfer

`_depositAmount` from user and `mint` shares amount for user. the problem is that if user convert `_shares` to token, he is going to receive lower amount because in most cases:

```
_depositAmount > _sharesToToken(_tokenToShares(_depositAmount))
```

and that's because of rounding in division. Value of `_shares` is less than _depositAmount. so `YeildSource` should only take part of `_depositAmount` that equals to `_sharesToToken(_tokenToShares(_depositAmount))` and mint `_share` for user.

Of course if `_precision` was high and `aToken.balanceOf(FieldSource.address) / totalSupply()` was low, then this amount will be insignificant, but for some cases it can be harmful for users. for example this conditions:

- `_perecision` is low like 1 or 2.
- `token` value is very high like BTC.
- `aToken.balanceOf(FieldSource.address) / totalSupply()` is high due to manipulation or profit in `Aave pool`.

Tools Used

VIM

Recommended Mitigation Steps

To resolve this issue this can be done:

```
function supplyTokenTo(uint256 _depositAmount, address _to) ex
  uint256 _shares = _tokenToShares(_depositAmount);
  require(_shares > 0, "AaveV3YS/shares-gt-zero");

  _depositAmount = _sharesToToken(_shares); // added hero to c
  address _underlyingAssetAddress = _tokenAddress();
  IERC20(_underlyingAssetAddress).safeTransferFrom(msg.sender,
  _pool().supply(_underlyingAssetAddress, _depositAmount, addr
```

```
        _mint(_to, _shares);

        emit SuppliedTokenTo(msg.sender, _shares, _depositAmount, _t
    }
```

PierrickGT (PoolTogether) confirmed and commented:

> PR: https://github.com/pooltogether/aave-v3-yield-source/pull/16

Justin Goro (judge) decreased severity from High to Medium

[M-02] `_depositAmount` requires to be updated to contract balance increase

Submitted by MaratCerby, also found by berndartmueller, cccz, CertoraInc, IIIIIII, and reassor

AaveV3YieldSource.sol#L231-L242

Every time transferFrom or transfer function in ERC20 standard is called there is a possibility that underlying smart contract did not transfer the exact amount entered. It is required to find out contract balance increase/decrease after the transfer.
This pattern also prevents from re-entrancy attack vector.

Recommended Mitigation Steps

Recommended code:
function supplyTokenTo(uint256 _depositAmount, address _to) external override nonReentrant {
uint256 _shares = _tokenToShares(_depositAmount);
require(_shares > 0, “AaveV3YS/shares-gt-zero”);

```
    address _underlyingAssetAddress = _tokenAddress();

    uint256 balanceBefore = IERC20(_underlyingAssetAddress).balanceO
    IERC20(_underlyingAssetAddress).safeTransferFrom(msg.sender, ad
    _depositAmount = IERC20(_underlyingAssetAddress).balanceOf(addre
```

```
    _pool().supply(_underlyingAssetAddress, _depositAmount, address(

    _mint(_to, _shares);

    emit SuppliedTokenTo(msg.sender, _shares, _depositAmount, _to);


}
```

PierrickGT (PoolTogether) acknowledged and commented:

> This scenario would only happen if we use fee on transfer tokens.
> We don't plan to support fee on transfer token, so for this reason, I have acknowledged the issue but we won't implement the code suggestion.

Justin Goro (judge) decreased severity to Medium and commented:

> This could lead to protocol leakage rather than risk of total funds loss.
> Downgrading to Medium Risk.
> Reentrancy isn't an issue since the function is marked nonReentrant.

[M-03] Owner or Managers can rug Aave rewards

Submitted by lllllll, also found by GimelSec

A malicious owner or manager can steal all Aave rewards that are meant for PoolTogether users.

Even if the user is benevolent the fact that there is a rug vector available may negatively impact the protocol's reputation.

Proof of Concept

```
File: contracts/AaveV3YieldSource.sol   #X

275     function claimRewards(address _to) external onlyManager(
276       require(_to != address(0), "AaveV3YS/payee-not-zero-ac
277
```

```
278         address[] memory _assets = new address[](1);
279         _assets[0] = address(aToken);
280 
281         (address[] memory _rewardsList, uint256[] memory _clai
282           .claimAllRewards(_assets, _to);
```

[AaveV3YieldSource.sol#L275-L282]

The `claimRewards()` function allows the caller to send the rewards to an arbitrary address.

Recommended Mitigation Steps

Use a `poolAddressesProviderRegistry`-like contract to determine where the rewards should go, instead of letting an address be passed in

[PierrickGT (PoolTogether) acknowledged and commented]:

> Governance will own this contract with a multisig that will need 7 signatures out of 11 members before a transaction can be sent, so it limits the possibilities of a rug pull. These rewards are not tokens deposited by the users, so it's also less of a concern.

[M-04] `RewardsController` Emission Manager Can Authorize Users to Claim on Behalf of the `AaveV3YieldSource` Contract and Siphon Yield

Submitted by leastwood

[aave/RewardsController.sol#L190-L193]
[aave/RewardsController.sol#L39-L42]
[aave/RewardsController.sol#L133-L143]
[AaveV3YieldSource.sol#L275-L286]

The `AaveV3YieldSource` contract allows the manager or owner of the contract to claim rewards from Aave's rewards controller. However, there is an external

dependency on this periphery Aave contract such that the emission manager of the `RewardsController` contract may allows other users to be authorized claimers.

Authorized claimers can claim rewards on behalf of the `AaveV3YieldSource` contract, effectively bypassing any restrictions put in place by this proprietary contract and its `claimRewards()` function. A malicious emissions manager can effectively siphon yield away from the `AaveV3YieldSource` contract and redirect it to them-self.

Recommended Mitigation Steps

Ensure this is understood and enforce that the `RewardsController` contract is owner by PoolTogether's multisig.

PierrickGT (PoolTogether) acknowledged and commented:

> Exactly, we will need to whitelist an address by calling the `setClaimer` function: aave/RewardsController.sol#L190
> We will probably setup a contract that can claim for various yield sources.

[M-05] Yield source does not correctly calculate share conversions

Submitted by lllllll

> The aTokens' value is pegged to the value of the corresponding supplied asset at a 1:1 ratio and can be safely stored, transferred or traded. All yield collected by the aTokens' reserves are distributed to aToken holders directly by continuously increasing their wallet balance.

https://docs.aave.com/developers/tokens/atoken

Impact

Incorrect share conversions lead to incorrect pricing of assets and loss of principal. aTokens are rebasing tokens, which means that holders of the token have their `balanceof()` increase over time, but each token is still redeemable for exactly one underlying asset. Any formula that does not return one out for one in is incorrect.

Proof of Concept

File: contracts/AaveV3YieldSource.sol #X

```
352     /**
353      * @notice Calculates the number of shares that should b
354      * @param _tokens Amount of asset tokens
355      * @return Number of shares.
356      */
357     function _tokenToShares(uint256 _tokens) internal view r
358       uint256 _supply = totalSupply();
359
360       // shares = (tokens * totalShares) / yieldSourceAToker
361       return _supply == 0 ? _tokens : _tokens.mul(_supply).d
362     }
363
364     /**
365      * @notice Calculates the number of asset tokens a user
366      * @param _shares Amount of shares
367      * @return Number of asset tokens.
368      */
369     function _sharesToToken(uint256 _shares) internal view r
370       uint256 _supply = totalSupply();
371
372       // tokens = (shares * yieldSourceATokenTotalSupply) /
373       return _supply == 0 ? _shares : _shares.mul(aToken.bal
374     }
```

[AaveV3YieldSource.sol#L352-L374]

The above code is used for both `supplyTokenTo()` and `redeemToken()` and does not return one for one. Consider the following chain of events:

1. There are no deposits yet
2. Alice deposits one wBTC, getting back a AaveV3YieldSource share, while the yield source gets the aToken
3. Some time later a total of one extra wBTC worth of aToken is generated as yield and is in the `balanceOf(this)`

4. Alice attempts to withdraw her one share but gets zero wBTC, because `(tokens{1} * totalSupply(){1}) / aToken.balanceOf(this){2}` is zero

Recommended Mitigation Steps

There does not need to be a conversion function - one share must always equal one token.

PierrickGT (PoolTogether) acknowledged and commented:

> This is a plausible but unlikely scenario since it assumes that only Alice has deposited into the yield source and that the interest rate is so high that 1 more wBTC is now in the yield source.
> Also, the calculation is wrong since 1 wBTC would represent 1 ether or `1000000000000000000` wei. So the actual calculation would return `(1000000000000000000 * 1000000000000000000) / 2000000000000000000 = 500000000000000000` wei or 0.5 wBTC.

> We also periodically capture the interest accumulated in the yield source by calling the PrizeFlush contract: pooltogether/PrizeFlush.sol#L105.
> The capture of the interest happens here: pooltogether/PrizeSplitStrategy.sol#L51.

> For these reasons, I've acknowledged the issue but we won't change how shares are calculated.

Justin Goro (judge) decreased severity to Medium and commented:

> Downgraded severity as this is more of a value leakage situation, especially given the unlikely edge case of an ERC20 token that sets decimals to 0 and uses low base values.

Low Risk and Non-Critical Issues

For this contest, 15 reports were submitted by wardens detailing low risk and non-critical issues. The report highlighted below by **MaratCerby** received the top score from the judge.

The following wardens also submitted reports: GimelSec, hake, 0xf15ers, gzeon, kebabsec, z3s, Dravee, pauliax, 0x52, 0xDjango, lllllll, delfin454000, reassor, *and* Picodes.

[N-01]

uint256 is assigned to zero by default, additional reassignment to zero is unnecessary Affected code: AaveV3YieldSource.sol#L142

Proof of Concept

https://docs.soliditylang.org/en/v0.8.13/control-structures.html#default-value

Recommended Mitigation Steps

Recommended code: uint256 private constant ADDRESSES*PROVIDER*ID;

[N-02]

Since solidity v0.8.0 SafeMath library is used by default in arithmetic operations. Using external SafeMath libraries is unnecessary.

Affected code: AaveV3YieldSource.sol#L361

Proof of Concept

https://blog.soliditylang.org/2020/12/16/solidity-v0.8.0-release-announcement/#:~:text=the%20near%20future.-,Checked,-arithmetic%2C%20i.e

Recommended Mitigation Steps

Recommended code: return _supply == 0 ? _tokens : (_tokens * _supply) / aToken.balanceOf(address(this));

[N-03]

Since solidity v0.8.0 SafeMath library is used by default in arithmetic operations. Using external SafeMath libraries is unnecessary.

Affected code: AaveV3YieldSource.sol#L373

Proof of Concept

https://blog.soliditylang.org/2020/12/16/solidity-v0.8.0-release-announcement/#:~:text=the%20near%20future.-,Checked,-arithmetic%2C%20i.e

Recommended Mitigation Steps

Recommended code: return _supply == 0 ? _shares : (_shares * aToken.balanceOf(address(this))) / _supply;

PierrickGT (PoolTogether) commented:

> **[N-01]:** Constant variables need to be initialized, so it is not possible to declare the variable without assigning a value.

> **[N-02] & [N-03]:** SafeMath has been removed in the following PR: https://github.com/pooltogether/aave-v3-yield-source/pull/5

> Great report by this warden that should get extra point for their recommendation to remove SafeMath.

Justin Goro (judge) commented:

> **Severity:** There were two issues reported. Both were non-critical.

Gas Optimizations

For this contest, 30 reports were submitted by wardens detailing gas optimizations. The report highlighted below by IIIIIII received the top score from the judge.

The following wardens also submitted reports: horsefacts, miguelmtzinf, pedroais, 0x4non, WatchPug, Dravee, gzeon, throttle, 0xDjango, pauliax, rotcivegaf,

joestakey, GimelSec, fatherOfBlocks, TrungOre, tabish, 0v3rf10w, 0xf15ers, 0xkatana, 242, Tadashi, defsec, hake, peritoflores, z3s, 0x1f8b, MaratCerby, simon135, *and* slywaters.

[G-01] State variables only set in the constructor should be declared `immutable`

Avoids a Gsset (20000 gas) in the constructor, and replaces each Gwarmacces (100 gas) with a `PUSH32` (3 gas)

```
File: contracts/AaveV3YieldSource.sol   #1

127     IAToken public aToken;
```

https://github.com/pooltogether/aave-v3-yield-source/blob/e63d1b0e396a5bce89f093630c282ca1c6627e44/contracts/AaveV3YieldSource.sol#L127

```
File: contracts/AaveV3YieldSource.sol   #2

130     IRewardsController public rewardsController;
```

https://github.com/pooltogether/aave-v3-yield-source/blob/e63d1b0e396a5bce89f093630c282ca1c6627e44/contracts/AaveV3YieldSource.sol#L130

```
File: contracts/AaveV3YieldSource.sol   #3

133     IPoolAddressesProviderRegistry public poolAddressesProvi
```

https://github.com/pooltogether/aave-v3-yield-source/blob/e63d1b0e396a5bce89f093630c282ca1c6627e44/contracts/AaveV3YieldSource.sol#L133

```
diff --git a/AaveV3YieldSource.sol.orig b/AaveV3YieldSource.sol.
```

```
index 3975311..1d229ff 100644
--- a/AaveV3YieldSource.sol.orig
+++ b/AaveV3YieldSource.sol.new
@@ -124,13 +124,13 @@ contract AaveV3YieldSource is ERC20, IYiel
    /* ============ Variables ============ */

    /// @notice Yield-bearing Aave aToken address.
-   IAToken public aToken;
+   IAToken public immutable aToken;

    /// @notice Aave RewardsController address.
-   IRewardsController public rewardsController;
+   IRewardsController public immutable rewardsController;

    /// @notice Aave poolAddressesProviderRegistry address.
-   IPoolAddressesProviderRegistry public poolAddressesProviderRe
+   IPoolAddressesProviderRegistry public immutable poolAddresses

    /// @notice ERC20 token decimals.
    uint8 private immutable _decimals;



diff --git a/gas.orig b/gas.new
index d87edc2..a6cd51d 100644
--- a/gas.orig
+++ b/gas.new
@@ -5,21 +5,21 @@
 ·····························|·····························|·····
 |  Contract                  ·  Method                    ·  Min
 ·····························|·····························|·····
-|  AaveV3YieldSourceHarness  ·  claimRewards              ·
+|  AaveV3YieldSourceHarness  ·  claimRewards              ·
 ·····························|·····························|·····
-|  AaveV3YieldSourceHarness  ·  decreaseERC20Allowance    ·
+|  AaveV3YieldSourceHarness  ·  decreaseERC20Allowance    ·
 ·····························|·····························|·····
-|  AaveV3YieldSourceHarness  ·  increaseERC20Allowance    ·
+|  AaveV3YieldSourceHarness  ·  increaseERC20Allowance    ·
 ·····························|·····························|·····
 |  AaveV3YieldSourceHarness  ·  mint                       ·
 ·····························|·····························|·····
-|  AaveV3YieldSourceHarness  ·  redeemToken                ·
+|  AaveV3YieldSourceHarness  ·  redeemToken                ·
 ·····························|·····························|·····
 |  AaveV3YieldSourceHarness  ·  setManager                 ·
```

```
 ·····························|·····························|·····
-|  AaveV3YieldSourceHarness   ·  supplyTokenTo             ·     1
+|  AaveV3YieldSourceHarness   ·  supplyTokenTo             ·     1
 ·····························|·····························|·····
-|  AaveV3YieldSourceHarness   ·  transferERC20             ·
+|  AaveV3YieldSourceHarness   ·  transferERC20             ·
 ·····························|·····························|·····
 |  ERC20Mintable              ·  approve                   ·
 ·····························|·····························|·····
@@ -29,7 +29,7 @@
 ·····························|·····························|·····
 |  Deployments                                             ·
 ···········································································|·····
-|  AaveV3YieldSourceHarness                                ·    25
+|  AaveV3YieldSourceHarness                                ·    24
 ···········································································|·····
 |  ERC20Mintable                                           ·
 ·-----------------------------------------------------------|-----
```

[G-02] `internal` functions only called once can be inlined to save gas

Not inlining costs 20 to 40 gas because of two extra `JUMP` instructions and additional stack operations needed for function calls.

```
File: contracts/AaveV3YieldSource.sol   #1

369     function _sharesToToken(uint256 _shares) internal view r
```

https://github.com/pooltogether/aave-v3-yield-source/blob/e63d1b0e396a5bce89f093630c282ca1c6627e44/contracts/AaveV3YieldSource.sol#L369

```
File: contracts/AaveV3YieldSource.sol   #2

388     function _poolProvider() internal view returns (IPoolAdc
```

https://github.com/pooltogether/aave-v3-yield-source/blob/e63d1b0e396a5bce89f093630c282ca1c6627e44/contracts/AaveV3

YieldSource.sol#L388

```
diff --git a/AaveV3YieldSource.sol.orig b/AaveV3YieldSource.sol.
index 3975311..d7ec7ec 100644
--- a/AaveV3YieldSource.sol.orig
+++ b/AaveV3YieldSource.sol.new
@@ -201,7 +201,12 @@ contract AaveV3YieldSource is ERC20, IYield
     * @return The underlying balance of asset tokens.
     */
    function balanceOfToken(address _user) external override retu
-    return _sharesToToken(balanceOf(_user));
+    uint256 _shares = balanceOf(_user);
+    uint256 _supply = totalSupply();
+
+    // tokens = (shares * yieldSourceATokenTotalSupply) / total
+    return _supply == 0 ? _shares : _shares.mul(aToken.balanceO
+
    }

    /**
@@ -361,18 +366,6 @@ contract AaveV3YieldSource is ERC20, IYield
     return _supply == 0 ? _tokens : _tokens.mul(_supply).div(aT
    }

-  /**
-   * @notice Calculates the number of asset tokens a user has i
-   * @param _shares Amount of shares
-   * @return Number of asset tokens.
-   */
-  function _sharesToToken(uint256 _shares) internal view returr
-    uint256 _supply = totalSupply();
-
-    // tokens = (shares * yieldSourceATokenTotalSupply) / total
-    return _supply == 0 ? _shares : _shares.mul(aToken.balanceO
-  }
-
    /**
     * @notice Returns the underlying asset token address.
     * @return Underlying asset token address.
@@ -381,22 +374,13 @@ contract AaveV3YieldSource is ERC20, IYiel
     return aToken.UNDERLYING_ASSET_ADDRESS();
    }

-  /**
-   * @notice Retrieves Aave PoolAddressesProvider address.
```

```
-   * @return A reference to PoolAddressesProvider interface.
-   */
-  function _poolProvider() internal view returns (IPoolAddresse
-    return
-      IPoolAddressesProvider(
-        poolAddressesProviderRegistry.getAddressesProvidersList
-      );
-  }
-
   /**
    * @notice Retrieves Aave Pool address.
    * @return A reference to Pool interface.
    */
   function _pool() internal view returns (IPool) {
-    return IPool(_poolProvider().getPool());
+    return IPool(IPoolAddressesProvider(
+        poolAddressesProviderRegistry.getAddressesProvidersList
+      ).getPool());
   }
 }
```

```
diff --git a/gas.orig b/gas.new
index d87edc2..6948266 100644
--- a/gas.orig
+++ b/gas.new
@@ -13,11 +13,11 @@
 ·····························|·····························|·····
 |  AaveV3YieldSourceHarness  ·  mint                       ·
 ·····························|·····························|·····
-|  AaveV3YieldSourceHarness  ·  redeemToken                ·
+|  AaveV3YieldSourceHarness  ·  redeemToken                ·
 ·····························|·····························|·····
 |  AaveV3YieldSourceHarness  ·  setManager                 ·
 ·····························|·····························|·····
-|  AaveV3YieldSourceHarness  ·  supplyTokenTo              ·     1
+|  AaveV3YieldSourceHarness  ·  supplyTokenTo              ·     1
 ·····························|·····························|·····
 |  AaveV3YieldSourceHarness  ·  transferERC20              ·
 ·····························|·····························|·····
@@ -29,7 +29,7 @@
 |  Deployments                                             ·
 ···········································|·····
-|  AaveV3YieldSourceHarness                                ·    25
```

```
+|  AaveV3YieldSourceHarness                                  ·    25
 ·-------------------------------------------------------------|-----
  |  ERC20Mintable                                             ·
  ·-------------------------------------------------------------|-----
```

[G-03] Using `> 0` costs more gas than `!= 0` when used on a `uint` in a `require()` statement

This change saves 6 gas per instance

```
File: contracts/AaveV3YieldSource.sol   #1

179       require(decimals_ > 0, "AaveV3YS/decimals-gt-zero");
```

https://github.com/pooltogether/aave-v3-yield-source/blob/e63d1b0e396a5bce89f093630c282ca1c6627e44/contracts/AaveV3YieldSource.sol#L179

```
File: contracts/AaveV3YieldSource.sol   #2

233       require(_shares > 0, "AaveV3YS/shares-gt-zero");
```

https://github.com/pooltogether/aave-v3-yield-source/blob/e63d1b0e396a5bce89f093630c282ca1c6627e44/contracts/AaveV3YieldSource.sol#L233

```
diff --git a/AaveV3YieldSource.sol.orig b/AaveV3YieldSource.sol.
index 3975311..e00ad47 100644
--- a/AaveV3YieldSource.sol.orig
+++ b/AaveV3YieldSource.sol.new
@@ -176,7 +176,7 @@ contract AaveV3YieldSource is ERC20, IYield

     require(_owner != address(0), "AaveV3YS/owner-not-zero-addr

-    require(decimals_ > 0, "AaveV3YS/decimals-gt-zero");
+    require(decimals_ != 0, "AaveV3YS/decimals-gt-zero");
     _decimals = decimals_;
```

```
     // Approve once for max amount
@@ -230,7 +230,7 @@ contract AaveV3YieldSource is ERC20, IYield
     */
    function supplyTokenTo(uint256 _depositAmount, address _to) e
      uint256 _shares = _tokenToShares(_depositAmount);
-     require(_shares > 0, "AaveV3YS/shares-gt-zero");
+     require(_shares != 0, "AaveV3YS/shares-gt-zero");

      address _underlyingAssetAddress = _tokenAddress();
      IERC20(_underlyingAssetAddress).safeTransferFrom(msg.sender
```

```
diff --git a/gas.orig b/gas.new
index d87edc2..9a90ad3 100644
--- a/gas.orig
+++ b/gas.new
@@ -17,7 +17,7 @@
 ·····························|·····························|·····
 |  AaveV3YieldSourceHarness  ·  setManager                 ·
 ·····························|·····························|·····
-|  AaveV3YieldSourceHarness  ·  supplyTokenTo              ·     1
+|  AaveV3YieldSourceHarness  ·  supplyTokenTo              ·     1
 ·····························|·····························|·····
 |  AaveV3YieldSourceHarness  ·  transferERC20              ·
 ·····························|·····························|·····
@@ -29,7 +29,7 @@
 ·····························|·····························|·····
 |  Deployments                                             ·
 ·····························|·····························|·····
-|  AaveV3YieldSourceHarness                                ·    25
+|  AaveV3YieldSourceHarness                                ·    25
 ·····························|·····························|·····
 |  ERC20Mintable                                           ·
 ·-----------------------------|---------------------------|------
```

[G-04] Usage of `uints` / `ints` smaller than 32 bytes (256 bits) incurs overhead

> When using elements that are smaller than 32 bytes, your contract's gas usage may be higher. This is because the EVM operates on 32 bytes at a time. Therefore, if the element is smaller than that, the EVM must use more operations in order to reduce the size of the element from 32 bytes to the desired size.

https://docs.soliditylang.org/en/v0.8.11/internals/layout_in_storage.html Use a larger size then downcast where needed

```
File: contracts/AaveV3YieldSource.sol   #1

47        uint8 decimals,
```

https://github.com/pooltogether/aave-v3-yield-source/blob/e63d1b0e396a5bce89f093630c282ca1c6627e44/contracts/AaveV3YieldSource.sol#L47

```
File: contracts/AaveV3YieldSource.sol   #2

136     uint8 private immutable _decimals;
```

https://github.com/pooltogether/aave-v3-yield-source/blob/e63d1b0e396a5bce89f093630c282ca1c6627e44/contracts/AaveV3YieldSource.sol#L136

```
File: contracts/AaveV3YieldSource.sol   #3

145     uint16 private constant REFERRAL_CODE = uint16(188);
```

https://github.com/pooltogether/aave-v3-yield-source/blob/e63d1b0e396a5bce89f093630c282ca1c6627e44/contracts/AaveV3YieldSource.sol#L145

```
File: contracts/AaveV3YieldSource.sol   #4

165       uint8 decimals_,
```

https://github.com/pooltogether/aave-v3-yield-source/blob/e63d1b0e396a5bce89f093630c282ca1c6627e44/contracts/AaveV3YieldSource.sol#L165

```
diff --git a/AaveV3YieldSource.sol.orig b/AaveV3YieldSource.sol.
index 3975311..0dfa477 100644
--- a/AaveV3YieldSource.sol.orig
+++ b/AaveV3YieldSource.sol.new
@@ -44,7 +44,7 @@ contract AaveV3YieldSource is ERC20, IYieldSou
     IPoolAddressesProviderRegistry poolAddressesProviderRegistr
     string name,
     string symbol,
-    uint8 decimals,
+    uint256 decimals,
     address owner
   );

@@ -133,7 +133,7 @@ contract AaveV3YieldSource is ERC20, IYieldS
   IPoolAddressesProviderRegistry public poolAddressesProviderRe

   /// @notice ERC20 token decimals.
-  uint8 private immutable _decimals;
+  uint256 private immutable _decimals;

   /**
    * @dev Aave genesis market PoolAddressesProvider's ID.
@@ -142,7 +142,7 @@ contract AaveV3YieldSource is ERC20, IYieldS
   uint256 private constant ADDRESSES_PROVIDER_ID = uint256(0);

   /// @dev PoolTogether's Aave Referral Code
-  uint16 private constant REFERRAL_CODE = uint16(188);
+  uint256 private constant REFERRAL_CODE = 188;

   /* ============ Constructor ============ */

@@ -162,7 +162,7 @@ contract AaveV3YieldSource is ERC20, IYieldS
     IPoolAddressesProviderRegistry _poolAddressesProviderRegist
     string memory _name,
     string memory _symbol,
-    uint8 decimals_,
+    uint256 decimals_,
     address _owner
   ) Ownable(_owner) ERC20(_name, _symbol) ReentrancyGuard() {
     require(address(_aToken) != address(0), "AaveV3YS/aToken-nc
@@ -218,7 +218,7 @@ contract AaveV3YieldSource is ERC20, IYieldS
    * @return The number of decimals.
    */
   function decimals() public view virtual override returns (uir
-    return _decimals;
```

```
+     return uint8(_decimals);
    }

    /**
@@ -234,7 +234,7 @@ contract AaveV3YieldSource is ERC20, IYieldS

      address _underlyingAssetAddress = _tokenAddress();
      IERC20(_underlyingAssetAddress).safeTransferFrom(msg.sender
-     _pool().supply(_underlyingAssetAddress, _depositAmount, adc
+     _pool().supply(_underlyingAssetAddress, _depositAmount, adc

      _mint(_to, _shares);
```

```
diff --git a/gas.orig b/gas.new
index d87edc2..367daee 100644
--- a/gas.orig
+++ b/gas.new
@@ -29,7 +29,7 @@
 ·····························|·····························|·····
 |  Deployments                                              ·
 ··········································|·····|·····|·····
-|  AaveV3YieldSourceHarness                 ·     25
+|  AaveV3YieldSourceHarness                 ·     25
 ··········································|·····|·····|·····
 |  ERC20Mintable                            ·
 ·------------------------------------------|------
```

[G-05] Don't use `SafeMath` once the solidity version is 0.8.0 or greater

Version 0.8.0 introduces internal overflow checks, so using `SafeMath` is redundant and adds overhead

```
File: contracts/AaveV3YieldSource.sol   #1

262   uint256 _balanceDiff = _afterBalance.sub(_beforeBalance);
```

https://github.com/pooltogether/aave-v3-yield-source/blob/e63d1b0e396a5bce89f093630c282ca1c6627e44/contracts/AaveV3

YieldSource.sol#L262

```
File: contracts/AaveV3YieldSource.sol   #2

361   return _supply == 0 ? _tokens : _tokens.mul(_supply).div(a
```

https://github.com/pooltogether/aave-v3-yield-source/blob/e63d1b0e396a5bce89f093630c282ca1c6627e44/contracts/AaveV3YieldSource.sol#L361

```
File: contracts/AaveV3YieldSource.sol   #3

373   return _supply == 0 ? _shares : _shares.mul(aToken.balance
```

https://github.com/pooltogether/aave-v3-yield-source/blob/e63d1b0e396a5bce89f093630c282ca1c6627e44/contracts/AaveV3YieldSource.sol#L373

```
diff --git a/AaveV3YieldSource.sol.orig b/AaveV3YieldSource.sol.
index 3975311..6346344 100644
--- a/AaveV3YieldSource.sol.orig
+++ b/AaveV3YieldSource.sol.new
@@ -11,7 +11,6 @@ import { IRewardsController } from "@aave/peri
 import { ERC20 } from "@openzeppelin/contracts/token/ERC20/ERC2
 import { IERC20 } from "@openzeppelin/contracts/token/ERC20/IER
 import { SafeERC20 } from "@openzeppelin/contracts/token/ERC20/
-import { SafeMath } from "@openzeppelin/contracts/utils/math/Sa
 import { ReentrancyGuard } from "@openzeppelin/contracts/securi

 import { Manageable, Ownable } from "@pooltogether/owner-manage
@@ -23,7 +22,6 @@ import { IYieldSource } from "@pooltogether/yi
  * @notice Yield Source for a PoolTogether prize pool that gene
  */
 contract AaveV3YieldSource is ERC20, IYieldSource, Manageable,
-  using SafeMath for uint256;
   using SafeERC20 for IERC20;

   /* ============ Events ============ */
@@ -259,7 +257,7 @@ contract AaveV3YieldSource is ERC20, IYieldS
     _pool().withdraw(_underlyingAssetAddress, _redeemAmount, ac
```

```
     uint256 _afterBalance = _assetToken.balanceOf(address(this)

-    uint256 _balanceDiff = _afterBalance.sub(_beforeBalance);
+    uint256 _balanceDiff = _afterBalance - _beforeBalance;
     _assetToken.safeTransfer(msg.sender, _balanceDiff);

     emit RedeemedToken(msg.sender, _shares, _redeemAmount);
@@ -358,7 +356,7 @@ contract AaveV3YieldSource is ERC20, IYield
     uint256 _supply = totalSupply();

     // shares = (tokens * totalShares) / yieldSourceATokenTotal
-    return _supply == 0 ? _tokens : _tokens.mul(_supply).div(aT
+    return _supply == 0 ? _tokens : _tokens * _supply / aToken.
   }

   /**
@@ -370,7 +368,7 @@ contract AaveV3YieldSource is ERC20, IYield
     uint256 _supply = totalSupply();

     // tokens = (shares * yieldSourceATokenTotalSupply) / total
-    return _supply == 0 ? _shares : _shares.mul(aToken.balanceO
+    return _supply == 0 ? _shares : _shares * aToken.balanceOf(
   }

   /**



diff --git a/gas.orig b/gas.new
index d87edc2..1c91b72 100644
--- a/gas.orig
+++ b/gas.new
@@ -13,11 +13,11 @@
 ·····························|·····························|·····
 |  AaveV3YieldSourceHarness  ·  mint                       ·
 ·····························|·····························|·····
-|  AaveV3YieldSourceHarness  ·  redeemToken                ·
+|  AaveV3YieldSourceHarness  ·  redeemToken                ·
 ·····························|·····························|·····
 |  AaveV3YieldSourceHarness  ·  setManager                 ·
 ·····························|·····························|·····
-|  AaveV3YieldSourceHarness  ·  supplyTokenTo              ·     1
+|  AaveV3YieldSourceHarness  ·  supplyTokenTo              ·     1
 ·····························|·····························|·····
 |  AaveV3YieldSourceHarness  ·  transferERC20              ·
 ·····························|·····························|·····
```

```
@@ -29,7 +29,7 @@
 ·····························|·····························|·····
 |  Deployments                                              ·
 ·····························|·····························|·····
-|  AaveV3YieldSourceHarness                                 ·    25
+|  AaveV3YieldSourceHarness                                 ·    24
 ·····························|·····························|·····
 |  ERC20Mintable                                            ·
 ·-----------------------------|-----------------------------|-----
```

[G-06] `require()` or `revert()` statements that check input arguments should be at the top of the function

Checks that involve constants should come before checks that involve state variables

```
File: contracts/AaveV3YieldSource.sol   #1

177       require(_owner != address(0), "AaveV3YS/owner-not-zero
```

https://github.com/pooltogether/aave-v3-yield-source/blob/e63d1b0e396a5bce89f093630c282ca1c6627e44/contracts/AaveV3YieldSource.sol#L177

```
File: contracts/AaveV3YieldSource.sol   #2

179       require(decimals_ > 0, "AaveV3YS/decimals-gt-zero");
```

https://github.com/pooltogether/aave-v3-yield-source/blob/e63d1b0e396a5bce89f093630c282ca1c6627e44/contracts/AaveV3YieldSource.sol#L179

```
diff --git a/AaveV3YieldSource.sol.orig b/AaveV3YieldSource.sol.
index 3975311..5d7d9fe 100644
--- a/AaveV3YieldSource.sol.orig
+++ b/AaveV3YieldSource.sol.new
@@ -166,17 +166,16 @@ contract AaveV3YieldSource is ERC20, IYiel
     address _owner
```

```
    ) Ownable(_owner) ERC20(_name, _symbol) ReentrancyGuard() {
      require(address(_aToken) != address(0), "AaveV3YS/aToken-nc
+     require(_owner != address(0), "AaveV3YS/owner-not-zero-addr
+     require(decimals_ > 0, "AaveV3YS/decimals-gt-zero");
+     require(address(_rewardsController) != address(0), "AaveV3Y
+     require(address(_poolAddressesProviderRegistry) != address(
      aToken = _aToken;

-     require(address(_rewardsController) != address(0), "AaveV3Y
      rewardsController = _rewardsController;

-     require(address(_poolAddressesProviderRegistry) != address(
      poolAddressesProviderRegistry = _poolAddressesProviderRegis

-     require(_owner != address(0), "AaveV3YS/owner-not-zero-addr
-
-     require(decimals_ > 0, "AaveV3YS/decimals-gt-zero");
      _decimals = decimals_;

      // Approve once for max amount
```

```
diff --git a/gas.orig b/gas.new
index d87edc2..e76a03d 100644
--- a/gas.orig
+++ b/gas.new
@@ -29,7 +29,7 @@
 ·····························|·····························|·····
 |  Deployments                                              ·
 ··········································|·····|·····|·····
-|  AaveV3YieldSourceHarness                 ·     25
+|  AaveV3YieldSourceHarness                 ·     25
 ··········································|·····|·····|·····
 |  ERC20Mintable                            ·
 ·-----------------------------------------|------|
```

[G-07] Use custom errors rather than `revert()` / `require()` strings to save gas

File: contracts/AaveV3YieldSource.sol #1

```
168         require(address(_aToken) != address(0), "AaveV3YS/aTok
```

https://github.com/pooltogether/aave-v3-yield-source/blob/e63d1b0e396a5bce89f093630c282ca1c6627e44/contracts/AaveV3YieldSource.sol#L168

```
File: contracts/AaveV3YieldSource.sol    #2

171         require(address(_rewardsController) != address(0), "Aa
```

https://github.com/pooltogether/aave-v3-yield-source/blob/e63d1b0e396a5bce89f093630c282ca1c6627e44/contracts/AaveV3YieldSource.sol#L171

```
File: contracts/AaveV3YieldSource.sol    #3

174         require(address(_poolAddressesProviderRegistry) != add
```

https://github.com/pooltogether/aave-v3-yield-source/blob/e63d1b0e396a5bce89f093630c282ca1c6627e44/contracts/AaveV3YieldSource.sol#L174

```
File: contracts/AaveV3YieldSource.sol    #4

177         require(_owner != address(0), "AaveV3YS/owner-not-zero
```

https://github.com/pooltogether/aave-v3-yield-source/blob/e63d1b0e396a5bce89f093630c282ca1c6627e44/contracts/AaveV3YieldSource.sol#L177

```
File: contracts/AaveV3YieldSource.sol    #5

179         require(decimals_ > 0, "AaveV3YS/decimals-gt-zero");
```

https://github.com/pooltogether/aave-v3-yield-source/blob/e63d1b0e396a5bce89f093630c282ca1c6627e44/contracts/AaveV3YieldSource.sol#L179

```
File: contracts/AaveV3YieldSource.sol   #6

233         require(_shares > 0, "AaveV3YS/shares-gt-zero");
```

https://github.com/pooltogether/aave-v3-yield-source/blob/e63d1b0e396a5bce89f093630c282ca1c6627e44/contracts/AaveV3YieldSource.sol#L233

```
File: contracts/AaveV3YieldSource.sol   #7

276         require(_to != address(0), "AaveV3YS/payee-not-zero-ac
```

https://github.com/pooltogether/aave-v3-yield-source/blob/e63d1b0e396a5bce89f093630c282ca1c6627e44/contracts/AaveV3YieldSource.sol#L276

```
File: contracts/AaveV3YieldSource.sol   #8

337         require(_token != address(aToken), "AaveV3YS/forbid-aT
```

https://github.com/pooltogether/aave-v3-yield-source/blob/e63d1b0e396a5bce89f093630c282ca1c6627e44/contracts/AaveV3YieldSource.sol#L337

```
File: contracts/AaveV3YieldSource.sol   #9

349         require(_token != address(aToken), "AaveV3YS/forbid-aT
```

https://github.com/pooltogether/aave-v3-yield-source/blob/e63d1b0e396a5bce89f093630c282ca1c6627e44/contracts/AaveV3YieldSource.sol#L349

```
diff --git a/AaveV3YieldSource.sol.orig b/AaveV3YieldSource.sol.
index 3975311..0a01635 100644
--- a/AaveV3YieldSource.sol.orig
+++ b/AaveV3YieldSource.sol.new
@@ -26,6 +26,17 @@ contract AaveV3YieldSource is ERC20, IYieldSc
   using SafeMath for uint256;
   using SafeERC20 for IERC20;

+  error ATokenNotZeroAddress();
+  error RCNotZeroAddress();
+  error PRNotZeroAddress();
+  error OwnerNotZeroAddress();
+  error DecimalsGtZero();
+  error SharesGtZero();
+  error PayeeNotZeroAddress();
+  error ForbidATokenTransfer();
+  error ForbidATokenAllowance();
+
+
   /* ============ Events ============ */

   /**
@@ -165,18 +176,18 @@ contract AaveV3YieldSource is ERC20, IYiel
     uint8 decimals_,
     address _owner
   ) Ownable(_owner) ERC20(_name, _symbol) ReentrancyGuard() {
-    require(address(_aToken) != address(0), "AaveV3YS/aToken-nc
+    if (address(_aToken) == address(0)) revert ATokenNotZeroAdc
     aToken = _aToken;

-    require(address(_rewardsController) != address(0), "AaveV3Y
+    if (address(_rewardsController) == address(0)) revert RCNot
     rewardsController = _rewardsController;

-    require(address(_poolAddressesProviderRegistry) != address(
+    if (address(_poolAddressesProviderRegistry) == address(0))
     poolAddressesProviderRegistry = _poolAddressesProviderRegis

-    require(_owner != address(0), "AaveV3YS/owner-not-zero-addr
+    if (_owner == address(0)) revert OwnerNotZeroAddress();

-    require(decimals_ > 0, "AaveV3YS/decimals-gt-zero");
+    if (decimals_ == 0) revert DecimalsGtZero();
     _decimals = decimals_;
```

```
     // Approve once for max amount
@@ -230,7 +241,7 @@ contract AaveV3YieldSource is ERC20, IYieldS
     */
    function supplyTokenTo(uint256 _depositAmount, address _to) e
      uint256 _shares = _tokenToShares(_depositAmount);
-     require(_shares > 0, "AaveV3YS/shares-gt-zero");
+     if (_shares == 0) revert SharesGtZero();

      address _underlyingAssetAddress = _tokenAddress();
      IERC20(_underlyingAssetAddress).safeTransferFrom(msg.sender
@@ -273,7 +284,7 @@ contract AaveV3YieldSource is ERC20, IYieldS
     * @return True if operation was successful.
     */
    function claimRewards(address _to) external onlyManagerOrOwne
-     require(_to != address(0), "AaveV3YS/payee-not-zero-address
+     if (_to == address(0)) revert PayeeNotZeroAddress();

      address[] memory _assets = new address[](1);
      _assets[0] = address(aToken);
@@ -334,7 +345,7 @@ contract AaveV3YieldSource is ERC20, IYieldS
      address _to,
      uint256 _amount
    ) external onlyManagerOrOwner {
-     require(address(_token) != address(aToken), "AaveV3YS/forbi
+     if (address(_token) == address(aToken)) revert ForbidAToker
      _token.safeTransfer(_to, _amount);
      emit TransferredERC20(msg.sender, _to, _amount, _token);
    }
@@ -346,7 +357,7 @@ contract AaveV3YieldSource is ERC20, IYieldS
     * @param _token Address of the ERC20 token to check
     */
    function _requireNotAToken(address _token) internal view {
-     require(_token != address(aToken), "AaveV3YS/forbid-aToken-
+     if (_token == address(aToken)) revert ForbidATokenTransfer(
    }

    /**



diff --git a/gas.orig b/gas.new
index d87edc2..3aa8ff3 100644
--- a/gas.orig
+++ b/gas.new
@@ -17,7 +17,7 @@
 ·····························|·····························|·····
```

```
|  AaveV3YieldSourceHarness  ·  setManager                ·
·····························|·····························|·····
-|  AaveV3YieldSourceHarness  ·  supplyTokenTo             ·     1
+|  AaveV3YieldSourceHarness  ·  supplyTokenTo             ·     1
·····························|·····························|·····
|  AaveV3YieldSourceHarness  ·  transferERC20             ·
·····························|·····························|·····
@@ -29,7 +29,7 @@
·····························|·····························|·····
|  Deployments                                            ·
·····························|·····························|·····
-|  AaveV3YieldSourceHarness                              ·    25
+|  AaveV3YieldSourceHarness                              ·    24
·····························|·····························|·····
|  ERC20Mintable                                          ·
·----------------------------|-----------------------------|------
```

[G-08] Functions guaranteed to revert when called by normal users can be marked `payable`

If a function modifier such as `onlyOwner` is used, the function will revert if a normal user tries to pay the function. Marking the function as `payable` will lower the gas cost for legitimate callers because the compiler will not include checks for whether a payment was provided. The extra opcodes avoided are `CALLVALUE` (2), `DUP1` (3), `ISZERO` (3), `PUSH2` (3), `JUMPI` (10), `PUSH1` (3), `DUP1` (3), `REVERT` (0), `JUMPDEST` (1), `POP` (2), which costs an average of about 21 gas per call to the function, in addition to the extra deployment cost

```
File: contracts/AaveV3YieldSource.sol   #1

275     function claimRewards(address _to) external onlyManagerC
```

https://github.com/pooltogether/aave-v3-yield-source/blob/e63d1b0e396a5bce89f093630c282ca1c6627e44/contracts/AaveV3YieldSource.sol#L275

```
File: contracts/AaveV3YieldSource.sol   #2

296     function decreaseERC20Allowance(
```

```
297       IERC20 _token,
298       address _spender,
299       uint256 _amount
300     ) external onlyManagerOrOwner {
```

https://github.com/pooltogether/aave-v3-yield-source/blob/e63d1b0e396a5bce89f093630c282ca1c6627e44/contracts/AaveV3YieldSource.sol#L296-L300

```
File: contracts/AaveV3YieldSource.sol   #3

315     function increaseERC20Allowance(
316       IERC20 _token,
317       address _spender,
318       uint256 _amount
319     ) external onlyManagerOrOwner {
```

https://github.com/pooltogether/aave-v3-yield-source/blob/e63d1b0e396a5bce89f093630c282ca1c6627e44/contracts/AaveV3YieldSource.sol#L315-L319

```
File: contracts/AaveV3YieldSource.sol   #4

332     function transferERC20(
333       IERC20 _token,
334       address _to,
335       uint256 _amount
336     ) external onlyManagerOrOwner {
```

https://github.com/pooltogether/aave-v3-yield-source/blob/e63d1b0e396a5bce89f093630c282ca1c6627e44/contracts/AaveV3YieldSource.sol#L332-L336

```
diff --git a/AaveV3YieldSource.sol.orig b/AaveV3YieldSource.sol.
index 3975311..b16b82a 100644
--- a/AaveV3YieldSource.sol.orig
+++ b/AaveV3YieldSource.sol.new
@@ -272,7 +272,7 @@ contract AaveV3YieldSource is ERC20, IYield
```

```
     * @param _to Address where the claimed rewards will be sent
     * @return True if operation was successful.
     */
-  function claimRewards(address _to) external onlyManagerOrOwne
+  function claimRewards(address _to) payable external onlyManag
     require(_to != address(0), "AaveV3YS/payee-not-zero-address

     address[] memory _assets = new address[](1);
@@ -297,7 +297,7 @@ contract AaveV3YieldSource is ERC20, IYieldS
     IERC20 _token,
     address _spender,
     uint256 _amount
-  ) external onlyManagerOrOwner {
+  ) payable external onlyManagerOrOwner {
     _requireNotAToken(address(_token));
     _token.safeDecreaseAllowance(_spender, _amount);
     emit DecreasedERC20Allowance(msg.sender, _spender, _amount,
@@ -316,7 +316,7 @@ contract AaveV3YieldSource is ERC20, IYieldS
     IERC20 _token,
     address _spender,
     uint256 _amount
-  ) external onlyManagerOrOwner {
+  ) payable external onlyManagerOrOwner {
     _requireNotAToken(address(_token));
     _token.safeIncreaseAllowance(_spender, _amount);
     emit IncreasedERC20Allowance(msg.sender, _spender, _amount,
@@ -333,7 +333,7 @@ contract AaveV3YieldSource is ERC20, IYieldS
     IERC20 _token,
     address _to,
     uint256 _amount
-  ) external onlyManagerOrOwner {
+  ) payable external onlyManagerOrOwner {
     require(address(_token) != address(aToken), "AaveV3YS/forbi
     _token.safeTransfer(_to, _amount);
     emit TransferredERC20(msg.sender, _to, _amount, _token);



diff --git a/gas.orig b/gas.new
index d87edc2..b10b8b3 100644
--- a/gas.orig
+++ b/gas.new
@@ -5,11 +5,11 @@
 ·····························|·····························|·····
 |  Contract                   ·  Method                     ·  Min
 ·····························|·····························|·····
```

```
-|  AaveV3YieldSourceHarness  ·  claimRewards              ·
+|  AaveV3YieldSourceHarness  ·  claimRewards              ·
 ·····························|·····························|·····
-|  AaveV3YieldSourceHarness  ·  decreaseERC20Allowance    ·
+|  AaveV3YieldSourceHarness  ·  decreaseERC20Allowance    ·
 ·····························|·····························|·····
-|  AaveV3YieldSourceHarness  ·  increaseERC20Allowance    ·
+|  AaveV3YieldSourceHarness  ·  increaseERC20Allowance    ·
 ·····························|·····························|·····
 |  AaveV3YieldSourceHarness  ·  mint                      ·
 ·····························|·····························|·····
@@ -19,7 +19,7 @@
 ·····························|·····························|·····
 |  AaveV3YieldSourceHarness  ·  supplyTokenTo             ·     1
 ·····························|·····························|·····
-|  AaveV3YieldSourceHarness  ·  transferERC20             ·
+|  AaveV3YieldSourceHarness  ·  transferERC20             ·
 ·····························|·····························|·····
 |  ERC20Mintable             ·  approve                   ·
 ·····························|·····························|·····
@@ -29,7 +29,7 @@
 |  Deployments                                            ·
 ·····························|·····························|·····
-|  AaveV3YieldSourceHarness                               ·    25
+|  AaveV3YieldSourceHarness                               ·    25
 ·····························|·····························|·····
 |  ERC20Mintable                                          ·
 ·-----------------------------|-----------------------------·------
```

[G-09] Method IDs can be fiddled with to reduce gas costs

See this page for details

```
File: contracts/AaveV3YieldSource.sol (various lines)   #1
```

https://github.com/pooltogether/aave-v3-yield-source/blob/e63d1b0e396a5bce89f093630c282ca1c6627e44/contracts/AaveV3YieldSource.sol

```
diff --git a/AaveV3YieldSource.sol.orig b/AaveV3YieldSource.sol.
```

```
index 3975311..dc04818 100644
--- a/AaveV3YieldSource.sol.orig
+++ b/AaveV3YieldSource.sol.new
@@ -180,7 +180,7 @@ contract AaveV3YieldSource is ERC20, IYieldS
     _decimals = decimals_;

     // Approve once for max amount
-    IERC20(_tokenAddress()).safeApprove(address(_pool()), type(
+    IERC20(_tokenAddress()).safeApprove(address(_pool_Jo$()), t

     emit AaveV3YieldSourceInitialized(
       _aToken,
@@ -234,7 +234,7 @@ contract AaveV3YieldSource is ERC20, IYieldS

     address _underlyingAssetAddress = _tokenAddress();
     IERC20(_underlyingAssetAddress).safeTransferFrom(msg.sender
-    _pool().supply(_underlyingAssetAddress, _depositAmount, add
+    _pool_Jo$().supply(_underlyingAssetAddress, _depositAmount,

     _mint(_to, _shares);

@@ -256,7 +256,7 @@ contract AaveV3YieldSource is ERC20, IYieldS
     _burn(msg.sender, _shares);

     uint256 _beforeBalance = _assetToken.balanceOf(address(this
-    _pool().withdraw(_underlyingAssetAddress, _redeemAmount, ad
+    _pool_Jo$().withdraw(_underlyingAssetAddress, _redeemAmount
     uint256 _afterBalance = _assetToken.balanceOf(address(this)

     uint256 _balanceDiff = _afterBalance.sub(_beforeBalance);
@@ -385,7 +385,7 @@ contract AaveV3YieldSource is ERC20, IYieldS
    * @notice Retrieves Aave PoolAddressesProvider address.
    * @return A reference to PoolAddressesProvider interface.
    */
-  function _poolProvider() internal view returns (IPoolAddresse
+  function _poolProvider_uaF() internal view returns (IPoolAddr
     return
       IPoolAddressesProvider(
         poolAddressesProviderRegistry.getAddressesProvidersList
@@ -396,7 +396,7 @@ contract AaveV3YieldSource is ERC20, IYieldS
    * @notice Retrieves Aave Pool address.
    * @return A reference to Pool interface.
    */
-  function _pool() internal view returns (IPool) {
-    return IPool(_poolProvider().getPool());
+  function _pool_Jo$() internal view returns (IPool) {
```

```
+     return IPool(_poolProvider_uaF().getPool());
    }
  }


diff --git a/gas.orig b/gas.new
index d87edc2..72bf242 100644
--- a/gas.orig
+++ b/gas.new
@@ -29,7 +29,7 @@
  ·····························|·····························|·····
  |  Deployments                                              ·
  ·····························|·····························|·····
-|  AaveV3YieldSourceHarness                                  ·    25
+|  AaveV3YieldSourceHarness                                  ·    25
  ·····························|·····························|·····
  |  ERC20Mintable                                            ·
  ·----------------------------|-----------------------------|------
```

[PierrickGT (PoolTogether) commented]:

> Great report by this warden, they should get extra points.
>
> **[G-01] State variables only set in the constructor should be declared immutable**
>
> PR: https://github.com/pooltogether/aave-v3-yield-source/pull/2
>
> **[G-02] internal functions only called once can be inlined to save gas**
>
> We prefer to keep the internal function `_sharesToToken` since it's a pretty important one with several lines of code in it.
>
> **[G-03] Using > 0 costs more gas than != 0 when used on a uint in a require() statement**
> **[G-05] Don't use SafeMath once the solidity version is 0.8.0 or greater**
>
> See comments on issue #11
>
> **[G-04] Usage of uints/ints smaller than 32 bytes (256 bits) incurs overhead**

> We prefer to cast them first instead to have to downcast them in each functions.

> **[G-06] require() or revert() statements that check input arguments should be at the top of the function**

> PR: https://github.com/pooltogether/aave-v3-yield-source/pull/12

> **[G-07] Use custom errors rather than revert()/require() strings to save gas**

> See comments on issue #13

> **[G-08] Functions guaranteed to revert when called by normal users can be marked payable**

> Interesting but since these functions don't accept native assets, we shouldn't mark them as payable

> **[G-09] Method IDs can be fiddled with to reduce gas costs**

> Seems overly complicate to save 12 gas.

Disclosures

C4 is an open organization governed by participants in the community.

C4 Contests incentivize the discovery of exploits, vulnerabilities, and bugs in smart contracts. Security researchers are rewarded at an increasing rate for finding higher-risk issues. Contest submissions are judged by a knowledgeable security researcher and solidity developer and disclosed to sponsoring developers. C4 does not conduct formal verification regarding the provided code but instead provides final verification.

C4 does not provide any guarantee or warranty regarding the security of this project. All smart contract software should be used at the sole risk and responsibility of users.

Top