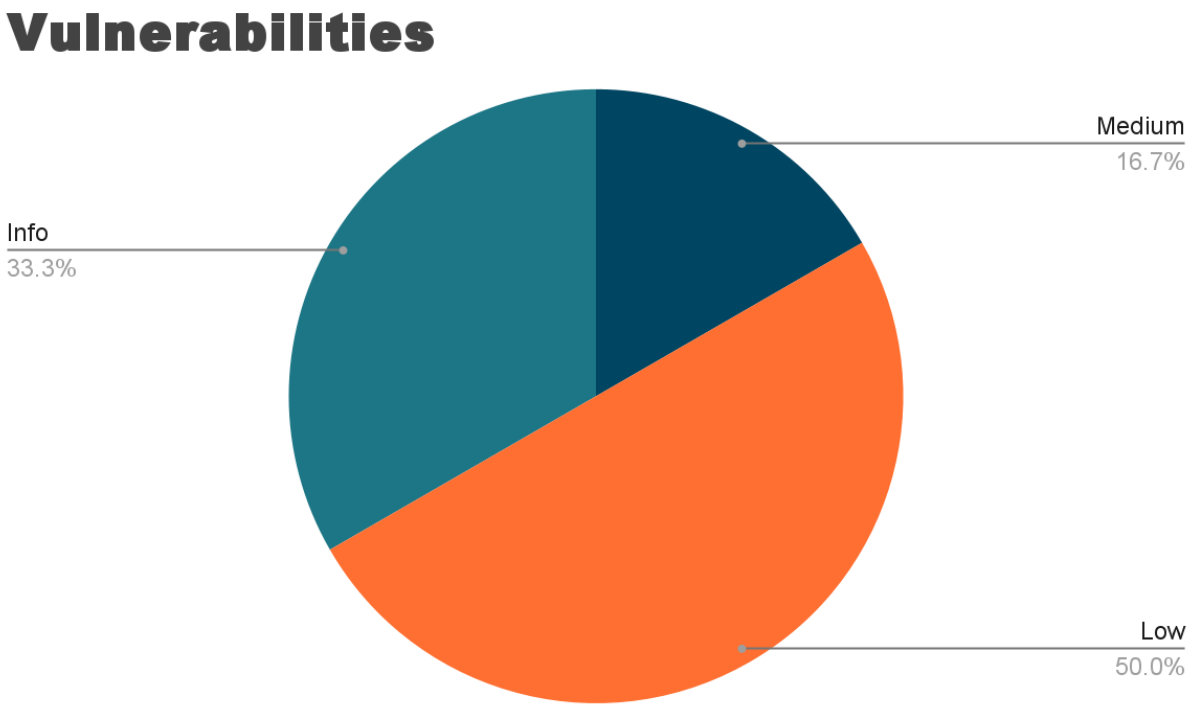
True Yield Audit

By John Nguyen (jooohn.eth)

General Info

Resources:
Github repo which consists of the project's core smart-contracts, tests, user interface and documentation.
Project author:
Umair Mirza (dreamygeek)
Audit author:
John Nguyen (jooohn.eth)



Summary

TrueYield is a Decentralized application built on Ethereum Blockchain that lets users stake their ETH and earn passive income yield on their ETH. This is a full-stack Web3 application with complete Frontend and Backend functionality. The dapp also integrates with Aave Lending pool so that the staked funds can be further lended to the Aave Lending Pool to generate yield.

The main branch of True Yield was reviewed.
Covered:

- TrueYield.sol and Interfaces - main contracts that are used to interact with the project.
- TrueYield.t.sol and Mocks - contract's unit tests and mocks.

The project was reviewed manually and with the help of tools.

Scope:

Github Repo
Commit

The commit reviewed was 2b4b581e939bfe3ed6a32014abf109d7fe7436ea. The review covered the repository at the specific commit and focused on the contracts directory.

Code Evaluation Matrix

Category	Mark	Description
Access Control	Okay	No access control was used. Access control not needed at this stage but highly recommended.
Libraries	Good	Only Openzeppelin's IERC20 was used. Less external dependency = good for security.

Documentation	Good	All comments were provided where needed.
Monitoring	Good	Events exist for all important functions that modify state variables.
Testing	Good	All tests passed with a good percentage of code coverage.
Decentralization	Good	No external party access provided.

Findings Explanation

Findings are broken down into sections by their respective impact:

- Critical, High, Medium, Low impact
    - These are findings that range from attacks that may cause loss of funds, impact control/ownership of the contracts, or cause any unintended consequences/actions that are outside the scope of the requirements.
- Gas Savings
    - Findings that can improve the gas efficiency of the contracts
- Informational
    - Findings including recommendations and best practices

No Critical and High Findings

Medium Findings

1. **Return value from IERC20 approve function ignored.**
    - Location: TrueYield.closePosition()
    - Description: the return value of an external call is not stored in a local or state variable.

```
IERC20(aWethAddress).approve(address(iWethGateway), type(uint256).max);
```

- Recommendation: Store return value and ensure expected value.

```
  bool approved = IERC20(aWethAddress).approve(
    address(iWethGateway),
    type(uint256).max
  );
  require(approved, "");
```

Low Findings

2. **Variable potentially used before declaration.**
    - Location: TrueYield.closePosition()

```
    (bool success, ) = payable(msg.sender).call{
      value: positions[positionId].weiStaked
    }("");
```

- Recommendation: Ensure that reaching a variable declaration does not depend on some condition.

```
bool success;

  //If the user is un-staking before the Unlock period, they won't gain any interest
  if (block.timestamp > positions[positionId].unlockDate) {
    uint256 amount = positions[positionId].weiStaked +
      positions[positionId].weiInterest;
    (success, ) = payable(msg.sender).call{ value: amount }("");
    require(success, "Transaction failed");
  } else {
    (success, ) = payable(msg.sender).call{
      value: positions[positionId].weiStaked
    }("");
    require(success, "Transaction failed");
  }
```

3. **Unused global variable.**
    - Location: TrueYield

```
Position private position;
```

    - Recommendation: remove unused variables.

4. **Unused function parameter.**
    - Location: TrueYield.calculateInterest()

```
 function calculateInterest(
  uint256 basisPoints,
  uint256 numDays,
  uint256 weiAmount
) public pure returns (uint256) {
  return (basisPoints * weiAmount) / 10000;
}
```

    - Description: parameter numDays never used.
    - Recommendation: remove unused parameters.

Informational Findings

5. **Comparison to a boolean constant.**
    - Location: TrueYield.closePosition() line#

```
  require(positions[positionId].open == true, "Position is closed");
```

    - Recommendation: remove equality to boolean constant.

```
  require(positions[positionId].open, "Position is closed");
```

6. **Naming convention not followed.**
    - Description: constants lendingPoolAddress and aWethAddress are not UPPER_CASE_WITH_UNDERSCORES.

```
address public constant lendingPoolAddress =
  0x4bd5643ac6f66a5237E18bfA7d47cF22f1c9F210;
```

```
address public constant aWethAddress =
  0x22404B0e2a7067068AcdaDd8f9D586F834cCe2c5;
```

- Recommendation: rename variables according to [Solidity naming conventions.]

```
address public constant LENDING_POOL_ADDRESS =
  0x4bd5643ac6f66a5237E18bfA7d47cF22f1c9F210;

address public constant A_WETH_ADDRESS =
  0x22404B0e2a7067068AcdaDd8f9D586F834cCe2c5;
```

Final Remarks

After reviewing the core smart contracts, no critical and high vulnerabilities were found, mostly low-level or informational issues occurred and one medium level vulnerability. Unit tests were reviewed - no anomalies found, the tests were accurate.