



DefySwap

Smart Contract Security Audit

Prepared by ShellBoxes

October 20th, 2021 - October 28th, 2021

[Shellboxes.com](https://shellboxes.com)

contact@shellboxes.com

Document Properties

| | |
|----------------|--------------------------|
| Client | DefySwap |
| Target | DefySwap Smart Contracts |
| Version | 1.0 |
| Classification | Public |

Scope

| Repo | Commit Hash |
|---|--|
| https://github.com/DefyFarm/Defyswap | 352d45af101c25bf5815486db48b479706362d39 |

Contacts

| COMPANY | EMAIL |
|------------|------------------------|
| ShellBoxes | contact@shellboxes.com |

Contents

- 1 Introduction 5
 - 1.1 About DefySwap 5
 - 1.2 Approach & Methodology 5
 - 1.2.1 Risk Methodology 6
- 2 Findings Overview 7
 - 2.1 Summary 7
 - 2.2 Key Findings 7
- 3 Finding Details 9
 - A DFYToken.sol 9
 - A.1 Approve Race Condition [MEDIUM] 9
 - A.2 Race Condition [MEDIUM] 10
 - A.3 Old Dev/BurnVault/IlpVault are not included in fee [MEDIUM] 11
 - A.4 Missing Address Verification [LOW] 14
 - A.5 Integer Overflow [LOW] 16
 - A.6 Floating Pragma [LOW] 17
 - B DFYMaster.sol 18
 - B.1 Usage Of transfer Instead Of safeTransfer [HIGH] 18
 - B.2 Race Condition [MEDIUM] 20
 - B.3 Owner Can Create Duplicate Pools [MEDIUM] 21
 - B.4 Reward Miscalculation [MEDIUM] 22
 - B.5 Usage of Block.TimeStamp [LOW] 25
 - B.6 Owner Can Renounce Ownership [LOW] 26
 - B.7 Missing Address Verification [LOW] 27
 - B.8 Missing Value Verification [LOW] 29
 - B.9 Floating Pragma [LOW] 31
 - C BurnVault.sol 32
 - C.1 Missing Address Verification [LOW] 32
 - C.2 Owner Can Renounce Ownership [LOW] 34
 - D Factory.sol 35
 - D.1 Approve Race [MEDIUM] 35
 - D.2 Missing Address Verification [LOW] 36

| | | |
|-----|--|-----|
| E | ILP.sol | 39 |
| E.1 | Usage Of transfer Instead Of safeTransfer [HIGH] | 39 |
| E.2 | Owner Can Renounce Ownership [LOW] | 40 |
| E.3 | Missing Address Verification [LOW] | 41 |
| E.4 | Integer Overflow [LOW] | 43 |
| F | Router.sol | 44 |
| F.1 | For Loop Over Dynamic Array [MEDIUM] | 44 |
| G | STUB.sol | 47 |
| G.1 | Missing Address Verification [LOW] | 47 |
| G.2 | Integer Overflow [LOW] | 48 |
| G.3 | Floating Pragma [LOW] | 49 |
| H | subDefyMaster.sol | 50 |
| H.1 | Owner Can Create Duplicate Pools [MEDIUM] | 50 |
| H.2 | Reward Miscalculation [MEDIUM] | 51 |
| H.3 | For Loop Over Dynamic Array [MEDIUM] | 52 |
| H.4 | Missing Value Verification [LOW] | 53 |
| H.5 | Missing Address Verification [LOW] | 55 |
| I | Zapper.sol | 56 |
| I.1 | Usage Of transfer Instead Of safeTransfer [HIGH] | 56 |
| I.2 | Owner Can Renounce Ownership [LOW] | 57 |
| I.3 | Missing Address Verification [LOW] | 58 |
| 4 | Static Analysis (Slither) | 61 |
| 5 | Conclusion | 135 |

1 Introduction

DefySwap engaged ShellBoxes to conduct a security assessment on the DefySwap beginning on October 20th, 2021 and ending October 28th, 2021. In this report, we detail our methodical approach to evaluate potential security issues associated with the implementation of smart contracts, by exposing possible semantic discrepancies between the smart contract code and design document, and by recommending additional ideas to optimize the existing code. Our findings indicate that the current version of smart contracts can still be enhanced further due to the presence of many security and performance concerns.

This document summarizes the findings of our audit.

1.1 About DefySwap

The project is being handled and launched by defy.farm team which is an established project on the BSC chain. DEFYSWAP is one of the most advanced trading platforms on the Fantom Opera chain which also offers multiple decisionmaking tools for the users and offers a user-friendly and detailed full-options trading experience for all.

| | |
|--------------|---|
| Issuer | DefySwap |
| Website | https://defyswap.finance |
| Type | Solidity Smart Contract |
| Audit Method | Whitebox |

1.2 Approach & Methodology

ShellBoxes used a combination of manual and automated security testing to achieve a balance between efficiency, timeliness, practicability, and correctness within the audit's scope. While manual testing is advised for identifying problems in logic, procedure, and implementation, automated testing techniques help to expand the coverage of smart contracts and can quickly detect code that does not comply with security best practices.

1.2.1 Risk Methodology

Vulnerabilities or bugs identified by ShellBoxes are ranked using a risk assessment technique that considers both the LIKELIHOOD and IMPACT of a security incident. This framework is effective at conveying the features and consequences of technological vulnerabilities.

Its quantitative paradigm enables repeatable and precise measurement, while also revealing the underlying susceptibility characteristics that were used to calculate the Risk scores. A risk level will be assigned to each vulnerability on a scale of 5 to 1, with 5 indicating the greatest possibility or impact.

- Likelihood quantifies the probability of a certain vulnerability being discovered and exploited in the untamed.
- Impact quantifies the technical and economic costs of a successful attack.
- Severity indicates the risk's overall criticality.

Probability and impact are classified into three categories: H, M, and L, which correspond to high, medium, and low, respectively. Severity is determined by probability and impact and is categorized into four levels, namely Critical, High, Medium, and Low.

| | | | | |
|--------|--------|------------|--------|--------|
| Impact | High | Critical | High | Medium |
| | Medium | High | Medium | Low |
| | Low | Medium | Low | Low |
| | | High | Medium | Low |
| | | Likelihood | | |

2 Findings Overview

2.1 Summary

The following is a synopsis of our conclusions from our analysis of the DefySwap implementation. During the first part of our audit, we examine the smart contract source code and run the codebase via a static code analyzer. The objective here is to find known coding problems statically and then manually check (reject or confirm) issues highlighted by the tool. Additionally, we check business logics, system processes, and DeFi-related components manually to identify potential hazards and/or defects.

2.2 Key Findings

In general, these smart contracts are well-designed and constructed, but their implementation might be improved by addressing the discovered flaws, which include , **3** high-severity, **11** medium-severity, **20** low-severity vulnerabilities.

| Vulnerabilities | Severity | Status |
|--|----------|--------------|
| Usage Of transfer Instead Of safeTransfer | HIGH | Fixed |
| Usage Of transfer Instead Of safeTransfer | HIGH | Fixed |
| Usage Of transfer Instead Of safeTransfer | HIGH | Fixed |
| Approve Race Condition | MEDIUM | Acknowledged |
| Race Condition | MEDIUM | Acknowledged |
| Old Dev/BurnVault/IlpVault are not included in fee | MEDIUM | Fixed |
| Race Condition | MEDIUM | Acknowledged |
| Owner Can Create Duplicate Pools | MEDIUM | Acknowledged |
| Reward Miscalculation | MEDIUM | Fixed |
| Approve Race | MEDIUM | Acknowledged |
| For Loop Over Dynamic Array | MEDIUM | Acknowledged |
| Owner Can Create Duplicate Pools | MEDIUM | Acknowledged |
| Reward Miscalculation | MEDIUM | Fixed |
| For Loop Over Dynamic Array | MEDIUM | Acknowledged |

| | | |
|------------------------------|-----|--------------|
| Missing Address Verification | LOW | Fixed |
| Integer Overflow | LOW | Acknowledged |
| Floating Pragma | LOW | Fixed |
| Usage of Block.TimeStamp | LOW | Acknowledged |
| Owner Can Renounce Ownership | LOW | Acknowledged |
| Missing Address Verification | LOW | fixed |
| Missing Value Verification | LOW | Fixed |
| Floating Pragma | LOW | Fixed |
| Missing Address Verification | LOW | Fixed |
| Owner Can Renounce Ownership | LOW | Acknowledged |
| Missing Address Verification | LOW | Fixed |
| Owner Can Renounce Ownership | LOW | Acknowledged |
| Missing Address Verification | LOW | Fixed |
| Integer Overflow | LOW | Acknowledged |
| Missing Address Verification | LOW | Fixed |
| Integer Overflow | LOW | Acknowledged |
| Floating Pragma | LOW | Fixed |
| Missing Value Verification | LOW | Fixed |
| Missing Address Verification | LOW | Fixed |
| Owner Can Renounce Ownership | LOW | Acknowledged |

3 Finding Details

A DFYToken.sol

A.1 Approve Race Condition [MEDIUM]

Description:

The standard [ERC20](#) implementation contains a widely-known racing condition in its approve function, wherein a spender is able to witness the token owner broadcast a transaction altering their approval and quickly sign and broadcast a transaction using [transferFrom](#) to move the current approved amount from the owner's balance to the spender. If the spender's transaction is validated before the owner's, the spender will be able to get both approval amounts of both transactions.

Code:

Listing 1: DFYtoken.sol

```
669     function _approve(address owner, address spender, uint256 amount)
        ↪ internal {
670     require(owner != address(0), 'ERC20: approve from the zero address');
671     require(spender != address(0), 'ERC20: approve to the zero address');
672     _allowances[owner][spender] = amount;
673     emit Approval(owner, spender, amount);
674 }
```

Risk Level:

Likelihood - 2

Impact - 5

Recommendation:

Use `increaseAllowance` and `decreaseAllowance` functions to modify the approval amount instead of using the `approve` function to modify it.

Status – Acknowledged

The DefySwap team has acknowledged the risk.

A.2 Race Condition [MEDIUM]

Description:

The `_burnFee`, the `_ilpFee` and the `_devFee` variables have setters. If the user checks the value of one of these variables, then performs a transfer, then the owner updates the fees, the order of the transaction might overturn and the user's transaction in this case will be executed with the new fees without him knowing about it.

Code:

Listing 2: DFYtoken.sol

```
1033 // A percentage of every transfer goes to Burn Vault ,ILP Vault & Dev
1034 uint256 burnAmount = amount.mul(_burnFee).div(1000);
1035 uint256 ilpAmount = amount.mul(_ilpFee).div(1000);
1036 uint256 devAmount = amount.mul(_devFee).div(1000);
```

Risk Level:

Likelihood – 1

Impact – 4

Recommendation:

Add the fees in the arguments of the `transfer` function, then add a `require` statements that verifies that the values that are provided in the arguments are the same as the ones that are stored in the smart contract.

Code:

Listing 3: DFYtoken.sol

```
1025 function _transfer(address sender, address recipient, uint256 amount,
    ↪ uint256 burnFee, uint256 ilpFee uint256 devFee) internal override
    ↪ {
1026     require(burnFee == _burnFee, "DFY: Invalid burnFree");
1027     require(ilpFee == _ilpFee, "DFY: Invalid ilpFree");
1028     require(devFee == _devFee, "DFY: Invalid devFree");
```

Status - Acknowledged

The DefySwap team has acknowledged the risk.

A.3 Old Dev/BurnVault/IlpVault are not included in fee [MEDIUM]

Description:

When setting the [Dev](#), the [BurnVault](#) and the [IlpVault](#) addresses, the old addresses are not included back in the fee. Thus, all the old addresses will be excluded from the fee transactions.

Code:

Listing 4: DFYtoken.sol

```
944 function setDev(address _dev) external onlyDev {
945     require(dev != address(0), 'DEFY: dev cannot be the zero address');
946     dev = _dev ;
947     _isExcludedFromFee[_dev] = true;
948     emit NewDeveloper(_dev);
949 }
950 function setBurnVault(address _burnVault) external onlyMaster {
951     BURN_VAULT = _burnVault ;
```

```

952  _isExcludedFromFee[_burnVault] = true;
953  emit SetBurnVault(_burnVault);
954  }
955  function setIlpVault(address _ilpVault) external onlyOwner {
956  ILP_VAULT = _ilpVault;
957  _isExcludedFromFee[_ilpVault] = true;
958  emit SetIlpVault(_ilpVault);
959  }

```

Risk Level:

Likelihood - 3

Impact - 2

Recommendation:

Include the old value of the address to the fee before updating it to the new value using the mapping `_isExcludedFromFee` and set it back to `false`.

Listing 5: DFYtoken.sol

```

944  function setDev(address _dev) external onlyDev {
945  require(dev != address(0), 'DEFY: dev cannot be the zero address');
946  _isExcludedFromFee[dev] = false;
947  dev = _dev ;
948  _isExcludedFromFee[_dev] = true;
949  emit NewDeveloper(_dev);
950  }
951  function setBurnVault(address _burnVault) external onlyMaster {
952  _isExcludedFromFee[BURN_VAULT] = false;
953  BURN_VAULT = _burnVault ;
954  _isExcludedFromFee[_burnVault] = true;
955  emit SetBurnVault(_burnVault);
956  }
957  function setIlpVault(address _ilpVault) external onlyOwner {
958  _isExcludedFromFee[ILP_VAULT] = false;

```

```

959 ILP_VAULT = _ilpVault;
960 _isExcludedFromFee[_ilpVault] = true;
961 emit SetIlpVault(_ilpVault);
962 }

```

Status - Fixed

The DefySwap team has fixed the issue by including the old value of the address to the fee before updating it.

Listing 6: DFYtoken.sol

```

971 function setDev(address _dev) external onlyDev {
972     require(dev != address(0), 'DEFY: dev cannot be the zero address');
973     _isExcludedFromFee[dev] = false;
974     dev = _dev ;
975     _isExcludedFromFee[_dev] = true;
976     emit NewDeveloper(_dev);
977 }
978 function setBurnVault(address _burnVault) external onlyMaster {
979     _isExcludedFromFee[BURN_VAULT] = false;
980     BURN_VAULT = _burnVault ;
981     _isExcludedFromFee[_burnVault] = true;
982     emit SetBurnVault(_burnVault);
983 }
984 function setIlpVault(address _ilpVault) external onlyOwner {
985     _isExcludedFromFee[ILP_VAULT] = false;
986     ILP_VAULT = _ilpVault;
987     _isExcludedFromFee[_ilpVault] = true;
988     emit SetIlpVault(_ilpVault);
989 }

```

A.4 Missing Address Verification [LOW]

Description:

Certain functions lack a safety check in the address, the address-type argument should include a zero-address test, otherwise, the contract's functionality may become inaccessible.

Code:

Listing 7: DFYtoken.sol

```
897 constructor(address _dev, address _bunVault, uint256 _initAmount) public
    ↪ { //address _ilpVault,
898 dev = _dev;
899 BURN_VAULT = _bunVault;
900 defyMaster = msg.sender;
901 mint(msg.sender, _initAmount);
902 _isExcludedFromFee[msg.sender] = true;
903 _isExcludedFromFee[_bunVault] = true;
904 }
```

Listing 8: DFYtoken.sol

```
932 function setRouter(address _router) external onlyOwner {
933 router = _router;
934 emit SetRouter(_router);
935 }
```

Listing 9: DFYtoken.sol

```
987 function setMaster(address master) public onlyMaster {
988 defyMaster = master;
989 emit SetDefyMaster(master);
990 }
```

Risk Level:

Likelihood - 1

Impact - 3

Recommendation:

It is recommended to undertake further validation prior to user-supplied data. The concerns can be resolved by utilizing a whitelist technique or a modifier.

Listing 10: DFYtoken.sol

```
987 function setMaster(address master) public onlyMaster {
988     require(master != address(0), 'DEFY: master cannot be the zero address')
           ↪ ;
989     defyMaster = master;
990     emit SetDefyMaster(master);
991 }
```

Status - Fixed

The DefySwap team has fixed the issue as recommended by adding `require` statements that verify the addresses provided in the arguments.

Listing 11: DFYtoken.sol

```
897 constructor(address _dev, address _burnVault, uint256 _initAmount) public
           ↪ {
898     require(_dev != address(0), 'DEFY: dev cannot be the zero address');
899     require(_burnVault != address(0), 'DEFY: burn vault cannot be the zero
           ↪ address');
900     dev = _dev;
901     BURN_VAULT = _burnVault;
902     defyMaster = msg.sender;
903     mint(msg.sender, _initAmount);
904     _isExcludedFromFee[msg.sender] = true;
905     _isExcludedFromFee[_burnVault] = true;
906     _isExcludedFromFee[_dev] = true;
```

```
907 }
```

Listing 12: DFYtoken.sol

```
934 function setRouter(address _router) external onlyOwner {
935     require(_router != address(0), 'DEFY: Router cannot be the zero address
        ↪ ');
936     router = _router;
937     emit SetRouter(_router);
938 }
```

Listing 13: DFYtoken.sol

```
995 function setMaster(address master) public onlyMaster {
996     require(master != address(0), 'DEFY: DefyMaster cannot be the zero
        ↪ address');
997     defyMaster = master;
998     emit SetDefyMaster(master);
999 }
```

A.5 Integer Overflow [LOW]

Description:

The `nonce` mapping was implemented and integrated in the signature process in order to prevent the spender from claiming the reward multiple times (replay attack). The problem here is long term, when the `nonces[target]` reaches $2^{256}-1$, the next increment will cause an integer overflow and the `nonces[signatory]` value will change to 0.

Code:

Listing 14: DFYToken.sol

```
1153 require(nonce == nonces[signatory]++, "DEFY::delegateBySig: invalid
        ↪ nonce");
```


Risk Level:

Likelihood – 1

Impact – 4

Recommendation:

Use the `add` function from the [SafeMath](#) library. Also, returning an error message would help to explain why the transaction failed.

Status – Acknowledged

The DefySwap team has acknowledged the risk.

A.6 Floating Pragma [LOW]

Description:

The contract makes use of the floating-point pragma 0.6.12. Contracts should be deployed using the same compiler version and flags that were used during the testing process. Locking the pragma helps ensure that contracts are not unintentionally deployed using another pragma, such as an obsolete version, that may introduce issues in the contract system.

Code:

Listing 15: DFYToken.sol

```
3 pragma solidity ^0.6.12;
```

Recommendation:

Consider locking the pragma version. It is advised that floating pragma not be used in production. Both `truffle-config.js` and `hardhat.config.js` support locking the pragma version.

Status - Fixed

The DefySwap team has fixed the issue by locking the pragma version to 0.6.12.

Listing 16: DFYToken.sol

```
3 pragma solidity 0.6.12;
```

B DFYMaster.sol

B.1 Usage Of transfer Instead Of safeTransfer [HIGH]

Description:

The [ERC20](#) standard token implementation functions also return the transaction status as a boolean. It is a good practice to check for the return status of the function call to ensure that the transaction was successful. It is the developer's responsibility to enclose these function calls with [require\(\)](#) to ensure that, when the intended [ERC20](#) function call returns false, the caller transaction also fails. However, it is mostly missed by developers when they carry out checks; in effect, the transaction would always succeed, even if the token transfer did not.

Code:

Listing 17: DFYMaster.sol

```
2101 function safeDefyTransfer(address _to, uint256 _amount) internal {
2102     uint256 defyBal = defy.balanceOf(address(this));
2103     if (_amount > defyBal) {
2104         defy.transfer(_to, defyBal);
2105     } else {
2106         defy.transfer(_to, _amount);
2107     }
2108 }
```

Listing 18: DFYMaster.sol

```
2111 function safeSecondRTransfer(address _to, uint256 _amount) internal {
```

```

2112 uint256 secondRBal = secondR.balanceOf(address(this));
2113 if (_amount > secondRBal) {
2114     secondR.transfer(_to, secondRBal);
2115 } else {
2116     secondR.transfer(_to, _amount);
2117 }
2118 }

```

Risk Level:

Likelihood - 3

Impact - 5

Recommendation:

Use the `safeTransfer` function from the `safeERC20` Implementation, or put the transfer call inside an `assert` or `require` to verify that it returned `true`.

Status - Fixed

The DefySwap team has fixed the issue by adding a `require` statement that verifies if the transfer has passed correctly.

Listing 19: DFYMaster.sol

```

2108 function safeDefyTransfer(address _to, uint256 _amount) internal {
2109     uint256 defyBal = defy.balanceOf(address(this));
2110     bool successfulTansfer = false;
2111     if (_amount > defyBal) {
2112         successfulTansfer = defy.transfer(_to, defyBal);
2113     } else {
2114         successfulTansfer = defy.transfer(_to, _amount);
2115     }
2116     require(successfulTansfer, "safeDefyTransfer: transfer failed");
2117 }

```

Listing 20: DFYMaster.sol

```
2120 function safeSecondRTransfer(address _to, uint256 _amount) internal {
2121     uint256 secondRBal = secondR.balanceOf(address(this));
2122     bool successfulTansfer = false;
2123     if (_amount > secondRBal) {
2124         successfulTansfer = secondR.transfer(_to, secondRBal);
2125     } else {
2126         successfulTansfer = secondR.transfer(_to, _amount);
2127     }
2128     require(successfulTansfer, "safeSecondRTransfer: transfer failed");
2129 }
```

B.2 Race Condition [MEDIUM]

Description:

The `depositFee` variable have a setter. If the user checks the value of this variable, then calls the deposit function, and the owner updates the `depositFee`, the order of the transaction might overturn and the user's transaction in this case will be executed with the new fee without him knowing about it.

Code:

Listing 21: DFYMaster.sol

```
2020 if (pool.depositFee > 0) {
2021     uint256 depositFee = amount_.mul(pool.depositFee).div(10000);
2022     pool.lpToken.safeTransfer(feeAddress, depositFee);
2023     user.amount = user.amount.add(amount_).sub(depositFee);
2024     pool.lpSupply = pool.lpSupply.add(amount_).sub(depositFee);
}
```

Risk Level:

Likelihood - 1

Impact - 4

Recommendation:

Add the `depositFee` in the arguments of the `deposit` function, then add a `require` statements that verifies that the value provided in the arguments is the same as the one that is stored in the smart contract.

Listing 22: DFYMaster.sol

```
2020 function deposit(uint256 _pid, uint256 _amount, uint256 _depositFee)
    ↪ public {
2021     require(depositFee == _depositFee, "DFY: Invalid depositFee");
```

Status - Acknowledged

The DefySwap team has acknowledged the risk.

B.3 Owner Can Create Duplicate Pools [MEDIUM]

Description:

The `add()` function is used to add a new pool, it turns out that it did not complete essential sanity checks to prohibit the creation of a new pool with duplicate LP tokens. If a new pool with a duplicate LP token is introduced, it is likely that an error in the reward distribution to the pools and staking will occur.

Code:

Listing 23: DFYMaster.sol

```
1762 function add(uint256 _allocPoint, uint256 _allocPointDR, IERC20 _lpToken
    ↪ , DefySTUB _stub, IERC20IERC20 _token1, uint256 _depositFee,
    ↪ uint256 _withdrawalFee, bool _offerILP, bool _issueSTUB,
1763 uint256 _rewardEndTimestamp, bool _withUpdate ) public onlyDev {
1764     require(_depositFee <= 600, "Add : Max Deposit Fee is 6%");
1765     require(_withdrawalFee <= 600, "Add : Max Deposit Fee is 6%");
1766     require(_rewardEndTimestamp > block.timestamp , "Add: invalid
    ↪ rewardEndTimestamp");
```

Risk Level:

Likelihood – 3

Impact – 3

Recommendation:

This might be avoided by defining a mapping from addresses to booleans, such that once added, LP tokens are mapped to `true`. A `require`-statement might then be added to the method to prevent the same LP token from being added again.

Status – Acknowledged

The DefySwap team has acknowledged the risk, saying that they need this as they may have to add a duplicate pool in an emergency situation while keeping 0 `allocpoints` to not used pools).

B.4 Reward Miscalculation [MEDIUM]

Description:

The `totalAllocPoint` variable is used to determine the portion of total rewards minted that each pool will get, making it a critical part in the rewards calculation. As a result, if the `totalAllocPoint` variable is changed without first updating the pending awards, the payout for each pool is calculated improperly. The following `add()` and `set()` functions modify the `totalAllocPoint` variable without updating the awards.

Code:

Listing 24: DFYMaster.sol

```
1762 function add(uint256 _allocPoint, uint256 _allocPointDR, IERC20 _lpToken
    ↪ , DefySTUB _stub, IERC20 _token0,
1763 IERC20 _token1, uint256 _depositFee, uint256 _withdrawalFee, bool
    ↪ _offerILP, bool _issueSTUB,
1764 uint256 _rewardEndTimestamp, bool _withUpdate ) public onlyDev {
```

```

1765 require(_depositFee <= 600, "Add : Max Deposit Fee is 6%");
1766 require(_withdrawalFee <= 600, "Add : Max Deposit Fee is 6%");
1767 require(_rewardEndTimestamp > block.timestamp , "Add: invalid
    ↪ rewardEndTimestamp");
1768 if (_withUpdate) {
1769 massUpdatePools();
1770 }

```

Listing 25: DFYMaster.sol

```

1762 function set(uint256 _pid, uint256 _allocPoint, uint256 _allocPointDR,
    ↪ IERC20 _token0,
1763 IERC20 _token1, uint256 _depositFee, uint256 _withdrawalFee, _offerILP,
1764 bool _issueSTUB, uint256 _rewardEndTimestamp, bool _withUpdate ) public
    ↪ onlyOwner {
1765 require(_depositFee <= 600, "Add : Max Deposit Fee is 6%");
1766 require(_withdrawalFee <= 600, "Add : Max Deposit Fee is 6%");
1767 require(_rewardEndTimestamp > block.timestamp, "Add: invalid
    ↪ rewardEndTimestamp");
1768 if (_withUpdate) {
1769 massUpdatePools();
1770 }

```

Risk Level:

Likelihood - 2

Impact - 4

Recommendation:

The Team should remove `_withUpdate` variable in the `set()` and `add()` functions and always calling the `massUpdatePools()` function before updating `totalAllocPoint` variable.

Status - Fixed

The DefySwap team have fixed the issue by removing `_withUpdate` variable in the `set()` and `add()` functions and calling the `massUpdatePools()` function before updating `totalAllocPoint` variable.

Listing 26: DFYMaster.sol

```
1762 function add(uint256 _allocPoint, uint256 _allocPointDR, IERC20 _lpToken
    ↪ , DefySTUB _stub, IERC20 _token0,
1763 IERC20 _token1, uint256 _depositFee, uint256 _withdrawalFee, bool
    ↪ _offerILP, bool _issueSTUB,
1764 uint256 _rewardEndTimestamp) public onlyDev {
1765     require(_depositFee <= 600, "Add : Max Deposit Fee is 6%");
1766     require(_withdrawalFee <= 600, "Add : Max Deposit Fee is 6%");
1767     require(_rewardEndTimestamp > block.timestamp , "Add: invalid
    ↪ rewardEndTimestamp");
1768     massUpdatePools();
1769 }
```

Listing 27: DFYMaster.sol

```
1762 function set(uint256 _pid, uint256 _allocPoint, uint256 _allocPointDR,
    ↪ IERC20 _token0, IERC20 _token1,
1763 uint256 _depositFee, uint256 _withdrawalFee, _offerILP, bool _issueSTUB
    ↪ ,
1764 uint256 _rewardEndTimestamp) public onlyOwner {
1765     require(_depositFee <= 600, "Add : Max Deposit Fee is 6%");
1766     require(_withdrawalFee <= 600, "Add : Max Deposit Fee is 6%");
1767     require(_rewardEndTimestamp > block.timestamp, "Add: invalid
    ↪ rewardEndTimestamp");
1768     massUpdatePools();
1769 }
```


B.5 Usage of Block.TimeStamp [LOW]

Description:

`block.timestamp` is used in the contract. The variable `block` is a set of variables. The timestamp does not always reflect the current time and may be inaccurate. The value of a block can be influenced by miners. Maximal Extractable Value attacks require a timestamp of up to 900 seconds. There is no guarantee that the value is right, all what is guaranteed is that it is higher than the timestamp of the previous block.

Code:

Listing 28: DFYMaster.sol

```
1787 uint256 lastRewardTimestamp = block.timestamp > startTimestamp ? block.  
    ↪ timestamp : startTimestamp;
```

Listing 29: DFYMaster.sol

```
1829 require(_rewardEndTimeStamp > block.timestamp , "Add: invalid  
    ↪ rewardEndTimeStamp");
```

Risk Level:

Likelihood - 3

Impact - 3

Recommendation:

You can use an Oracle to get the exact time or verify if a delay of 900 seconds will not destroy the logic of the staking contract.

Status - Acknowledged

The DefySwap team has acknowledged the risk.

B.6 Owner Can Renounce Ownership [LOW]

Description:

Typically, the contract's owner is the account that deploys the contract. As a result, the owner is able to perform certain privileged activities on his behalf. The `renounceOwnership` function is used in smart contracts to renounce ownership. Otherwise, if the contract's ownership has not been transferred previously, it will never have an Owner, which is risky.

Code:

Listing 30: DFYMaster.sol

```
1477 contract DefyMaster is Ownable , ReentrancyGuard {  
1478     using SafeMath for uint256;
```

Risk Level:

Likelihood - 1

Impact - 3

Recommendation:

It is advised that the Owner cannot call `renounceOwnership` without first transferring ownership to a different address. Additionally, if a multi-signature wallet is utilized, executing the `renounceOwnership` method for two or more users should be confirmed. Alternatively, the `RenounceOwnership` functionality can be disabled by overriding it.

Status - Acknowledged

The DefySwap team has acknowledged the risk.

B.7 Missing Address Verification [LOW]

Description:

Certain functions lack a safety check in the address, the address-type argument should include a zero-address test, otherwise, the contract's functionality may become inaccessible.

Code:

Listing 31: DFYMaster.sol

```
1598 constructor(DfyToken _defy, DefySTUB _stub, BurnVault _burnvault,
    ↪ ImpermanentLossProtection _ilp,
1599 address _devaddr, address _feeAddress, uint256 _startTimestamp, uint256
    ↪ _initMint) public {
1600 defy = _defy;
1601 burn_vault = _burnvault;
1602 ilp = _ilp;
1603 devaddr = _devaddr;
1604 feeAddress = _feeAddress;
1605 startTimestamp = _startTimestamp;
```

Listing 32: DFYMaster.sol

```
1640 function setImpermanentLossProtection(address _ilp)public onlyDev
    ↪ returns (bool){
1641 ilp = ImpermanentLossProtection(_ilp);
1642 }
1643 function setFeeAddress(address _feeAddress)public onlyDev returns (bool)
    ↪ {
1644 feeAddress = _feeAddress;
1645 emit SetFeeAddress(_feeAddress);
1646 return true;
1647 }
1648 function setDFY(DfyToken _dfy)public onlyDev returns (bool){
1649 defy = _dfy;
```

```

1650 emit SetDFY(address(_dfy));
1651 return true;
1652 }
1653 function setSecondaryReward(IERC20 _rewardToken)public onlyDev returns (
    ↪ bool){
1654 secondR = _rewardToken ;
1655 emit SetSecondaryReward(address(_rewardToken));
1656 return true;
1657 }

```

Listing 33: DFYMaster.sol

```

2121 function dev(address _devaddr) public {
2122 require(msg.sender == devaddr, "dev: wut?");
2123 devaddr = _devaddr;
2124 }

```

Risk Level:

Likelihood - 1

Impact - 3

Recommendation:

It's recommended to undertake further validation prior to user-supplied data. The concerns can be resolved by utilizing a whitelist technique or a modifier.

Listing 34: DFYMaster.sol

```

2121 function dev(address _devaddr) public {
2122 require(msg.sender == devaddr, "dev: wut?");
2123 require(_devaddr != address(0), 'DEFY: dev cannot be the zero address')
    ↪ ;
2124 devaddr = _devaddr;
2125 }

```

Status - fixed

The DefySwap team has fixed the issue by adding `require` statements that verify the addresses provided in the arguments.

Listing 35: DFYMaster.sol

```
2132 function dev(address _devaddr) public {
2133     require(_devaddr != address(0), 'DEFY: dev cannot be the zero address');
2134     require(msg.sender == devaddr, "dev: wut?");
2135     devaddr = _devaddr;
2136 }
```

B.8 Missing Value Verification [LOW]

Description:

Certain functions lack a safety check in the values, the values of the arguments should include some safety checks test, otherwise, the contract's functionality may get hurt.

Code:

Listing 36: DFYMaster.sol

```
1715 function setStartTimestamp(uint256 sTimestamp) public onlyDev{
1716     startTimestamp = sTimestamp;
1717     emit UpdateStartTimestamp(sTimestamp);
1718 }
```

Listing 37: DFYMaster.sol

```
1720 function updateMultiplier(uint256 multiplierNumber) public onlyDev {
1721     BONUS_MULTIPLIER = multiplierNumber;
1722 }
```

Listing 38: DFYMaster.sol

```

1762 function add(uint256 _allocPoint, uint256 _allocPointDR, IERC20 _lpToken
    ↪ , DefySTUB _stub, IERC20 _token0, IERC20 _token1, uint256
    ↪ _depositFee, uint256 _withdrawalFee, _offerILP, bool _issueSTUB,
    ↪ uint256 _rewardEndTimestamp, bool _withUpdate ) public onlyDev {
1763 require(_depositFee <= 600, "Add : Max Deposit Fee is 6%");
1764 require(_withdrawalFee <= 600, "Add : Max Deposit Fee is 6%");
1765 require(_rewardEndTimestamp > block.timestamp , "Add: invalid
    ↪ rewardEndTimestamp");

```

Risk Level:

Likelihood - 1

Impact - 3

Recommendation:

It's recommended to undertake further validation prior to user-supplied data. The concerns can be resolved by utilizing require statements.

Listing 39: DFYMaster.sol

```

1715 function setStartTimestamp(uint256 sTimestamp) public onlyDev{
1716     require(sTimestamp > block.timestamp, "Invalid Timestamp");
1717     startTimestamp = sTimestamp;
1718     emit UpdateStartTimestamp(sTimestamp);
1719 }

```

Listing 40: DFYMaster.sol

```

1720 function updateMultiplier(uint256 multiplierNumber) public onlyDev {
1721     require(multiplierNumber != 0, " multiplierNumber should not be null");
1722     BONUS_MULTIPLIER = multiplierNumber;
1723 }

```

Listing 41: DFYMaster.sol

```

1762 function add(uint256 _allocPoint, uint256 _allocPointDR, IERC20 _lpToken
    ↪ , DefySTUB _stub, IERC20 _token0,

```

```

1763 IERC20 _token1, uint256 _depositFee, uint256 _withdrawalFee, _offerILP,
    ↪ bool _issueSTUB,
1764 uint256 _rewardEndTimestamp, bool _withUpdate ) public onlyDev {
1765     require(_allocPoint != 0 && _allocPointDR != 0, "AllocPoint and
        ↪ allocPointDR should not be null");
1766     require(_depositFee <= 600, "Add : Max Deposit Fee is 6%");
1767     require(_withdrawalFee <= 600, "Add : Max Deposit Fee is 6%");
1768     require(_rewardEndTimestamp > block.timestamp , "Add: invalid
        ↪ rewardEndTimestamp");

```

Status - Fixed

The DefySwap team has fixed the issue by adding `require` statements in order to verify the values that are provided in the arguments.

Listing 42: DFYMaster.sol

```

1724 function setStartTimestamp(uint256 sTimestamp) public onlyDev{
1725     require(sTimestamp > block.timestamp, "Invalid Timestamp");
1726     startTimestamp = sTimestamp;
1727     emit UpdateStartTimestamp(sTimestamp);
1728 }

```

Listing 43: DFYMaster.sol

```

1730 function updateMultiplier(uint256 multiplierNumber) public onlyDev {
1731     require(multiplierNumber != 0, " multiplierNumber should not be null");
1732     BONUS_MULTIPLIER = multiplierNumber;
1733 }

```

B.9 Floating Pragma [LOW]

Description:

The contract makes use of the floating-point pragma [0.6.12](#). Contracts should be deployed using the same compiler version and flags that were used during the testing process. Lock-

ing the pragma helps ensure that contracts are not unintentionally deployed using another pragma, such as an obsolete version, that may introduce issues in the contract system.

Code:

Listing 44: DFYMaster.sol

```
3 pragma solidity ^0.6.12;
```

Risk Level:

Likelihood - 1

Impact - 3

Recommendation:

Consider locking the pragma version. It is advised that floating pragma not be used in production. Both `truffle-config.js` and `hardhat.config.js` support locking the pragma version.

Status - Fixed

The DefySwap team has fixed the issue by locking the pragma version to [0.6.12](#).

Listing 45: DFYMaster.sol

```
3 pragma solidity 0.6.12;
```

C BurnVault.sol

C.1 Missing Address Verification [LOW]

Description:

Certain functions lack a safety check in the address, the address-type argument should include a zero-address test, otherwise, the contract's functionality may become inaccessible.

Code:

Listing 46: BurnVault.sol

```
876 function setDefyMaster (address master) external onlyDefy{
877     defyMaster = master ;
878     emit SetDefyMaster(master);
879 }
880 function setDefy (address _defy) external onlyDefy{
881     defy = DfyToken(_defy);
882 }
```

Risk Level:

Likelihood - 1

Impact - 3

Recommendation:

It is recommended to undertake further validation prior to user-supplied data. The concerns can be resolved by utilizing a whitelist technique or a modifier.

Listing 47: BurnVault.sol

```
876 function setDefyMaster (address master) external onlyDefy{
877     require(_devaddr != address(0), 'DEFY: DefyMaster cannot be the zero
      ↪ address');
878     defyMaster = master ;
879     emit SetDefyMaster(master);
880 }
881 function setDefy (address _defy) external onlyDefy{
882     require(_defy!= address(0), 'DEFY: Defy cannot be the zero address');
```

Status - Fixed

The DefySwap team has fixed the issue as recommended by adding a [require](#) statement that verifies the addresses provided in the arguments.

Listing 48: BurnVault.sol

```
876 function setDefyMaster (address master) external onlyDefy{
877     require(master != address(0), 'DEFY: DefyMaster cannot be the zero
        ↪ address');
878     defyMaster = master ;
879     emit SetDefyMaster(master);
880 }
```

Listing 49: BurnVault.sol

```
882 function setDefy (address _defy) external onlyDefy{
883     require(_defy!= address(0), 'DEFY: DFY cannot be the zero address');
884     defy = DfyToken(_defy);
885 }
```

C.2 Owner Can Renounce Ownership [LOW]

Description:

Typically, the contract's owner is the account that deploys the contract. As a result, the owner is able to perform certain privileged activities on his behalf. The `renounceOwnership` function is used in smart contracts to renounce ownership. Otherwise, if the contract's ownership has not been transferred previously, it will never have an Owner, which is risky.

Code:

Listing 50: BurnVault.sol

```
863 contract BurnVault is Ownable {
864     DfyToken public defy;
865     address public defyMaster;
```

Risk Level:

Likelihood – 1

Impact – 3

Recommendation:

It is advised that the Owner cannot call `renounceOwnership` without first transferring ownership to a different address. Additionally, if a multi-signature wallet is utilized, executing the `renounceOwnership` method for two or more users should be confirmed. Alternatively, the `renounceOwnership` functionality can be disabled by overriding it.

Status – Acknowledged

The DefySwap team has acknowledged the risk.

D Factory.sol

D.1 Approve Race [MEDIUM]

Description:

The standard `ERC20` implementation contains a widely-known racing condition in its `approve` function, wherein a spender is able to witness the token owner broadcast a transaction altering their approval and quickly sign and broadcast a transaction using `transferFrom` to move the current approved amount from the owner's balance to the spender. If the spender's transaction is validated before the owner's, the spender will be able to get both approval amounts of both transactions.

Code:

Listing 51: DefySwapERC20.sol

```
158 function _approve(address owner, address spender, uint256 value) private
    ↪ {
159 allowance[owner][spender] = value;
```

```
160 emit Approval(owner, spender, value);
161 }
```

Risk Level:

Likelihood - 2

Impact - 5

Recommendation:

Use `increaseAllowance` and `decreaseAllowance` functions to modify the approval amount instead of using the `approve` function to modify it.

Status - Acknowledged

The DefySwap team has acknowledged the risk.

D.2 Missing Address Verification [LOW]

Description:

Certain functions lack a safety check in the address, the address-type argument should include a zero-address test, otherwise, the contract's functionality may become inaccessible.

Code:

Listing 52: DefySwapERC20.sol

```
174 function transfer(address to, uint256 value) external returns (bool) {
175     _transfer(msg.sender, to, value);
176     return true;
177 }
```

Listing 53: DefySwapPair.sol

```
347 function initialize(address _token0, address _token1) external {
```

```

348 require(msg.sender == factory, 'DefySwap: FORBIDDEN'); // sufficient
    ↪ check
349 token0 = _token0;
350 token1 = _token1;
351 }

```

Listing 54: DefySwapFactory.sol

```

507 constructor(address _feeToSetter) public {
508 feeToSetter = _feeToSetter;
509 }

```

Listing 55: DefySwapFactory.sol

```

532 function setFeeTo(address _feeTo) external {
533 require(msg.sender == feeToSetter, "DefySwap: FORBIDDEN");
534 feeTo = _feeTo;
535 }
536 function setFeeToSetter(address _feeToSetter) external {
537 require(msg.sender == feeToSetter, "DefySwap: FORBIDDEN");
538 feeToSetter = _feeToSetter;
539 }

```

Risk Level:

Likelihood - 1

Impact - 3

Recommendation:

It's recommended to undertake further validation prior to user-supplied data. The concerns can be resolved by utilizing a whitelist technique or a modifier.

Listing 56: DefySwapERC20

```

174 function transfer(address to, uint256 value) external returns (bool) {
175 require(to != address(0), 'DEFY: to cannot be the zero address');
176 _transfer(msg.sender, to, value);

```

```
177 return true;
178 }
```

Listing 57: DefySwapPair.sol

```
374 function initialize(address _token0, address _token1) external {
375     require(token0 != address(0) && token1 != address(0), 'DEFY: tokens
        ↪ cannot be the zero address');
376     require(msg.sender == factory, 'DefySwap: FORBIDDEN'); // sufficient
        ↪ check
377     token0 = _token0;
378     token1 = _token1;
379 }
```

Listing 58: DefySwapPair.sol

```
507 constructor(address _feeToSetter) public {
508     require(_feeToSetter != address(0), 'DEFY: feeToSetter cannot be the
        ↪ zero address');
509     feeToSetter = _feeToSetter;
510 }
```

Listing 59: DefySwapFactory.sol

```
532 function setFeeTo(address _feeTo) external {
533     require(_feeTo != address(0), 'DEFY: feeTo cannot be the zero address')
        ↪ ;
534     require(msg.sender == feeToSetter, "DefySwap: FORBIDDEN");
535     feeTo = _feeTo;
536 }
537 function setFeeToSetter(address _feeToSetter) external {
538     require(_feeToSetter != address(0), 'DEFY: feeToSetter cannot be the
        ↪ zero address');
539     require(msg.sender == feeToSetter, "DefySwap: FORBIDDEN");
540     feeToSetter = _feeToSetter;
```

Status - Fixed

The DefySwap team has fixed the issue as recommended by adding `require` statements in order to verify the values that are provided in the arguments.

E ILP.sol

E.1 Usage Of transfer Instead Of safeTransfer [HIGH]

Description:

The `ERC20` standard token implementation functions also return the transaction status as a Boolean. It is a good practice to check for the return status of the function call to ensure that the transaction was successful. It is the developer's responsibility to enclose these function calls with `require()` to ensure that, when the intended `ERC20` function call returns `false`, the caller transaction also fails. However, it is mostly missed by developers when they carry out checks; in effect, the transaction would always succeed, even if the token transfer did not.

Code:

Listing 60: ImpermanentLossProtection.sol

```
761 ting 32 : ImpermanentLossProtection (Line 761)
762 function defyTransfer(address _to, uint256 _amount) externalonlyFarm {
763     uint256 defyBal = IERC20(defy).balanceOf(address(this));
764     uint256 xfAmt = _amount;
765     if( xfAmt > defyBal )
766     xfAmt = defyBal;
767     if(xfAmt > 0)
768     IERC20(defy).transfer(_to, xfAmt);
769 }
```

Risk Level:

Likelihood – 3

Impact – 5

Recommendation:

Use the `safeTransfer` function from the `safeERC20` Implementation, or put the transfer call inside an `assert` or `require` to verify that it returned `true`.

Status – Fixed

The DefySwap team has fixed the issue by adding a `require` statement that verifies if the transfer has passed correctly.

E.2 Owner Can Renounce Ownership [LOW]

Description:

Typically, the contract's owner is the account that deploys the contract. As a result, the owner is able to perform certain privileged activities on his behalf. The `renounceOwnership` function is used in smart contracts to renounce ownership. Otherwise, if the contract's ownership has not been transferred previously, it will never have an Owner.

Code:

Listing 61: ImpermanentLossProtection.sol

```
691 contract ImpermanentLossProtection is Ownable {
692     using SafeMath for uint256;
693     using SafeERC20 for IERC20;
```

Risk Level:

Likelihood – 1

Impact – 3

Recommendation:

It is advised that the Owner cannot call `renounceOwnership` without first transferring ownership to a different address. Additionally, if a multi-signature wallet is utilized, executing the `renounceOwnership` method for two or more users should be confirmed. Alternatively, the renounce ownership functionality can be disabled by overriding it.

Status - Acknowledged

The DefySwap team has acknowledged the risk.

E.3 Missing Address Verification [LOW]

Description:

Certain functions lack a safety check in the address, the address-type argument should include a zero-address test, otherwise, the contract's functionality may become inaccessible.

Code:

Listing 62: ImpermanentLossProtection.sol

```
730 constructor( address _defy, address _defyMaster)
731 public
732 {
733     defy = _defy;
734     defyMaster = _defyMaster;
735     devAddr = msg.sender;
```

Listing 63: ImpermanentLossProtection.sol

```
742 function setAddresses( address _defy, address _defyMaster ) public
    ↪ onlyFarm {
743     defy = _defy;
744     defyMaster = _defyMaster;
745     set(0,IERC20(0),IERC20(0),false);
```

```
746 poolInfo[0].lpToken = _defy ;
747 }
```

Listing 64: ImpermanentLossProtection.sol

```
774 function dev(address _devAddr) public {
775     require(msg.sender == devAddr, "dev: wut?");
776     devAddr = _devAddr;
777 }
778 function add(address _lpToken, IERC20 _token0, IERC20 _token1, bool
    ↪ _offerILP)
779     public onlyDev {
780     poolInfo.push(
781     PoolInfo({
782     lpToken: _lpToken,
783     token0: _token0,
784     token1: _token1,
785     token0_decimal: _token0.decimals(),
786     token1_decimal: _token1.decimals(),
787     impermanentLossProtection: _offerILP
788     })
789     );
790 }
```

Risk Level:

Likelihood - 1

Impact - 3

Recommendation:

It's recommended to undertake further validation prior to user-supplied data. The concerns can be resolved by utilizing a whitelist technique or a modifier.

Status - Fixed

The DefySwap team has fixed the issue as recommended by adding `require` statements that verify the addresses provided in the arguments.

E.4 Integer Overflow [LOW]

Description:

The `pow` and `defyPrice` are vulnerable to integer overflow since they are the result of an addition, the problem here is in the long term, when the variable reaches 2^{256-1} , the next increment will cause an integer overflow, so the value will change to `0`.

Code:

Listing 65: ImpermanentLossProtection.sol

```
873 if(_dec0 > _dec1){
874   pow = 18 + _dec0 - _dec1;
875 }
876 if (_dec0 < _dec1){
877   pow = 18 + _dec1 - _dec0;
878 }
879 defyPrice = ( (10 ** pow) * (r0) ) / (r1);
```

Risk Level:

Likelihood - 1

Impact - 3

Recommendation:

Use the `add` function from the `SafeMath` library. Also, returning an error message like would help explain why the transaction failed.

Status - Acknowledged

The DefySwap team has acknowledged the risk.

F Router.sol

F.1 For Loop Over Dynamic Array [MEDIUM]

Description:

When smart contracts are deployed or their associated functions are invoked, the execution of these operations always consumes a certain quantity of gas, according to the amount of computation required to accomplish them. Modifying an unknown-size array that grows in size over time can result in a Denial-of-Service. Simply by having an excessively huge array, users can exceed the gas limit, therefore preventing the transaction from ever succeeding.

Code:

Listing 66: DefySwapLibrary.sol

```
321 function getAmountsOut(address factory, uint amountIn, address[] memory
    ↪ path) internal view
322 returns (uint[] memory amounts) {
323 require(path.length >= 2, 'DefySwapLibrary: INVALID_PATH');
324 amounts = new uint[](path.length);
325 amounts[0] = amountIn;
326 for (uint i; i < path.length - 1; i++) {
327 (uint reserveIn, uint reserveOut) = getReserves(factory, path[i], path[i
    ↪ + 1]);
328 amounts[i + 1] = getAmountOut(amounts[i], reserveIn, reserveOut);
329 }
330 }
```

Listing 67: DefySwapLibrary.sol

```

332 function getAmountsIn(address factory, uint256 amountOut, address[]
    ↪ memory path) internal
333 view returns (uint256[] memory amounts) {
334 require(path.length >= 2, "DefySwapLibrary: INVALID_PATH");
335 amounts = new uint256[](path.length);
336 amounts[amounts.length - 1] = amountOut;
337 for (uint256 i = path.length - 1; i > 0; i--) {
338 (uint256 reserveIn, uint256 reserveOut) = getReserves(factory, path[i -
    ↪ 1], path[i]);
339 amounts[i - 1] = getAmountIn(amounts[i], reserveIn, reserveOut);
340 }
341 }
342 }

```

Listing 68: DefySwapRouter.sol

```

603 for (uint256 i; i < path.length - 1; i++) {
604 (address input, address output) = (path[i], path[i + 1]);
605 (address token0, ) = DefySwapLibrary.sortTokens(input, output);
606 uint256 amountOut = amounts[i + 1];
607 (uint256 amount0Out, uint256 amount1Out) = input == token0
608 ? (uint256(0), amountOut)
609 : (amountOut, uint256(0));
610 address to = i < path.length - 2
611 ? DefySwapLibrary.pairFor(factory, output, path[i + 2])
612 : _to;
613 IDefySwapPair(DefySwapLibrary.pairFor(factory, input, output)).swap(
614 amount0Out,
615 amount1Out,
616 to,
617 new bytes(0)
618 );
619 }

```

Listing 69: DefySwapRouter.sol

```
721 for (uint256 i; i < path.length - 1; i++) {
722     (address input, address output) = (path[i], path[i + 1]);
723     (address token0, ) = DefySwapLibrary.sortTokens(input, output);
724     IDefySwapPair pair = IDefySwapPair(
725     DefySwapLibrary.pairFor(factory, input, output)
726     );
727     uint256 amountInput;
728     uint256 amountOutput;
729     {
730     // scope to avoid stack too deep errors
731     (uint256 reserve0, uint256 reserve1, ) = pair.getReserves();
732     (uint256 reserveInput, uint256 reserveOutput) = input == token0
733     ? (reserve0, reserve1)
734     : (reserve1, reserve0);
735     amountInput = IERC20(input).balanceOf(address(pair)).sub(
736     reserveInput
737     );
738     amountOutput = DefySwapLibrary.getAmountOut(
739     amountInput,
740     reserveInput,
```

Risk Level:

Likelihood - 2

Impact - 4

Recommendation:

Avoid actions that involve looping across the entire data structure. If you really must loop over an array of unknown size, arrange for it to consume many blocs and thus multiple transactions.

Status - Acknowledged

The DefySwap team has acknowledged the risk.

G STUB.sol

G.1 Missing Address Verification [LOW]

Description:

Certain functions lack a safety check in the address, the address-type argument should include a zero-address test, otherwise, the contract's functionality may become inaccessible.

Code:

Listing 70: DefySTUB.sol

```
934 function delegateBySig(address delegatee, uint256 nonce, uint256 expiry,
    ↪ uint8 v, bytes32 r,
935 bytes32 s ) external {
936 bytes32 domainSeparator = keccak256(
937 abi.encode(
938 DOMAIN_TYPEHASH,
939 keccak256(bytes(name()))),
940 getChainId(),
941 address(this)
942 )
943 );
944 bytes32 structHash = keccak256(
945 abi.encode(DELEGATION_TYPEHASH, delegatee, nonce, expiry)
946 );
```

Risk Level:

Likelihood - 1

Impact - 3

Recommendation:

It's recommended to undertake further validation prior to user-supplied data. The concerns can be resolved by utilizing a whitelist technique or a modifier.

Status - Fixed

The DefySwap team has fixed the issue as recommended by adding `require` statements that verify the addresses provided in the arguments.

G.2 Integer Overflow [LOW]

Description:

The nonce mapping was implemented and integrated in the signature process in order to prevent the spender from claiming the reward multiple times (replay attack). The problem here is long term, when the `nonces[target]` reaches 2^{256-1} , the next increment will cause an integer overflow and the `nonces[target]` value will change to 0.

Code:

Listing 71: DefySTUB.sol

```
972 require(nonce == nonces[signatory]++, "DEFY::delegateBySig: invalid  
    ↪ nonce");
```

Risk Level:

Likelihood - 1

Impact - 3

Recommendation:

Use the add function from the [SafeMath](#) library. Also, returning an error message like would help explain why the transaction failed.

Status - Acknowledged

The DefySwap team has acknowledged the risk.

G.3 Floating Pragma [LOW]

Description:

The contract makes use of the floating-point pragma [0.6.12](#). Contracts should be deployed using the same compiler version and flags that were used during the testing process. Locking the pragma helps ensure that contracts are not unintentionally deployed using another pragma, such as an obsolete version, that may introduce issues in the contract system.

Code:

Listing 72: DefySTUB.sol

```
3 pragma solidity ^0.6.12;
```

Risk Level:

Likelihood - 2

Impact - 2

Recommendation:

Consider locking the pragma version. It is advised that floating pragma not be used in production. Both `truffle-config.js` and `hardhat.config.js` support locking the pragma version.

Status - Fixed

The DefySwap team has fixed the issue by locking the pragma version to [0.6.12](#).

Listing 73: DefySTUB.sol

```
3 pragma solidity 0.6.12;
```

H subDefyMaster.sol

H.1 Owner Can Create Duplicate Pools [MEDIUM]

Description:

The `add()` function is used to add a new pool, it turns out that it did not complete essential sanity checks to prohibit the creation of a new pool with duplicate LP tokens. If a new pool with a duplicate LP token is introduced, it is likely that an error in the reward distribution to the pools and staking will occur.

Code:

Listing 74: SubDefyMaster.sol

```
675 function add(uint256 _allocPoint, uint256 _depositFee, uint256
    ↪ _withdrawalFee, IERC20 _lpToken, uint256 _rewardEndTimestamp,
    ↪ bool _withUpdate) public onlyDev {
676     require(_depositFee <= 600 , "ADD : Max Deposit fee is 6%");
677     require(_withdrawalFee <= 600 , "ADD : Max Withdrawal fee is 6%");
678     require(_rewardEndTimestamp > block.timestamp , "ADD : invalid
    ↪ rewardEndTimestamp");
679     require(_rewardEndTimestamp <= endTimestamp , "ADD : rewardEndTimestamp
    ↪ higher than endTimesif (_withUpdate) {
680     massUpdatePools();
681 }
```

Risk Level:

Likelihood – 2

Impact – 4

Recommendation:

This might be avoided by defining a mapping from addresses to booleans, such that once added, LP tokens are mapped to true. A `require`-statement might then be added to the method to prevent the same LP token from being added again.

Status – Acknowledged

The DefySwap team has acknowledged the risk, saying that they need this as they may have to add a duplicate pool in an emergency situation while keeping 0 `allocpoints` to not used pools.

H.2 Reward Miscalculation [MEDIUM]

Description:

The `totalAllocPoint` variable is used to determine the portion of total rewards minted that each pool will get, making it a critical part in the rewards calculation. As a result, if the `totalAllocPoint` variable is changed without first updating the pending awards, the payout for each pool is calculated improperly. The following `add()` and `set()` functions modify the `totalAllocPoint` variable without updating the awards.

Code:

Listing 75: subDefyMaster.sol

```
675 function add(uint256 _allocPoint, uint256 _depositFee, uint256
    ↪ _withdrawalFee, IERC20 _lpToken,
676 uint256 _rewardEndTimestamp, bool _withUpdate) public onlyDev {
677     require(_depositFee <= 600 , "ADD : Max Deposit fee is 6%");
678     require(_withdrawalFee <= 600 , "ADD : Max Withdrawal fee is 6%");
```

```

679 require(_rewardEndTimestamp > block.timestamp , "ADD : invalid
    ↪ rewardEndTimestamp");
680 require(_rewardEndTimestamp <= endTimestamp , "ADD : rewardEndTimestamp
    ↪ higher than endTimesif (_withUpdate) {
681 massUpdatePools();
682 }

```

Risk Level:

Likelihood - 2

Impact - 4

Recommendation:

The DefySwap team should remove `_withUpdate` variable in the `set()` and `add()` functions and always calling the `massUpdatePools()` function before updating the `totalAllocPoint` variable.

Status - Fixed

The DefySwap team have fixed the issue by removing `_withUpdate` variable in the `set()` and `add()` functions and calling the `massUpdatePools()` function before updating `totalAllocPoint` variable.

H.3 For Loop Over Dynamic Array [MEDIUM]

Description:

When smart contracts are deployed or their associated functions are invoked, the execution of these operations always consumes a certain quantity of gas, according to the amount of computation required to accomplish them. Modifying an unknown-size array that grows in size over time can result in a Denial-of-Service. Simply by having an excessively huge array, users can exceed the gas limit, therefore preventing the transaction from ever succeeding.

Code:

Listing 76: subDefyMaster.sol

```
776 function massUpdatePools() public {
777     uint256 length = poolInfo.length;
778     for (uint256 pid = 0; pid < length; pid++) {
779         updatePool(pid);
780     }
781 }
```

Risk Level:

Likelihood - 1

Impact - 4

Recommendation:

Avoid actions that involve looping across the entire data structure. If you really must loop over an array of unknown size, arrange for it to consume many blocs and thus multiple transactions.

Status - Acknowledged

The DefySwap team has acknowledged the risk.

H.4 Missing Value Verification [LOW]

Description:

Certain functions lack a safety check in the values, the values of the arguments should include some safety checks test, otherwise, the contract's functionality may get hurt.

Code:

Listing 77: subDefyMaster.sol

```
665 function setStartTimestamp(uint256 sTimestamp) public onlyOwner{
666     startTimestamp = sTimestamp;
667 }
```

Listing 78: subDefyMaster.sol

```
659 function updateMultiplier(uint256 multiplierNumber) public onlyDev {
660     BONUS_MULTIPLIER = multiplierNumber;
661 }
```

Listing 79: subDefyMaster.sol

```
665 function updateTaxRatio(uint256 _tax) public onlyDev {
666     taxRatio = (10000 - _tax);
667 }
```

Recommendation:

It is recommended to undertake further validation prior to user-supplied data. The concerns can be resolved by utilizing require statements

Listing 80: subDefyMaster.sol

```
62 function updateTaxRatio(uint256 _tax) public onlyDev {
63     require(tax <= 10000, " Underflow protection");
64     taxRatio = (10000 - _tax);
65 }
```

Status - Fixed

The DefySwap team has fixed the issue by adding require statements in order to verify the values that are provided in the arguments.

H.5 Missing Address Verification [LOW]

Description:

Certain functions lack a safety check in the address, the address-type argument should include a zero-address test, otherwise, the contract's functionality may become inaccessible.

Code:

Listing 81: subDefyMaster.sol

```
616 constructor(IERC20 _rewardToken, address _devaddr, address _feeAddress,
617     uint256 _startTimestamp, uint256 _endTimestamp) public {
618     rewardToken = _rewardToken;
619     devaddr = _devaddr;
620     feeAddress = _feeAddress;
621     startTimestamp = _startTimestamp;
622     endTimestamp = _endTimestamp;
623 }
624 function setFeeAddress(address _feeAddress) public onlyDev returns (bool
    ↪ ) {
625     feeAddress = _feeAddress;
626     emit feeAddressUpdated(_feeAddress);
627     return true;
628 }
```

Risk Level:

Likelihood - 1

Impact - 3

Recommendation:

It's recommended to undertake further validation prior to user-supplied data. The concerns can be resolved by utilizing a whitelist technique or a modifier.

Status - Fixed

The DefySwap team has fixed the issue by adding `require` statements that verify the addresses provided in the arguments.

I Zapper.sol

I.1 Usage Of transfer Instead Of safeTransfer [HIGH]

Description:

The `ERC20` standard token implementation functions also return the transaction status as a Boolean. It is a good practice to check for the return status of the function call to ensure that the transaction was successful. It is the developer's responsibility to enclose these function calls with `require()` to ensure that, when the intended `ERC20` function call returns `false`, the caller transaction also fails. However, it is mostly missed by developers when they carry out checks; in effect, the transaction would always succeed, even if the token transfer did not.

Code:

Listing 82: Zap.sol

```
986 function zapInToken(address _from, uint256 amount, address _to, address
    ↪ routerAddr,
987 address _recipient) external override {
988   _approveTokenIfNeeded(_from, routerAddr);
989   if (isFeeOnTransfer[_from]) {
990     IERC20(_from).transferFrom(msg.sender, address(this), amount);
991     _swapTokenToLP(
992       _from,
993       IERC20(_from).balanceOf(address(this)),
994       _to,
995       _recipient,
996       routerAddr
997     );
```



```
998 return;
999 } else {
1000 IERC20(_from).safeTransferFrom(msg.sender, address(this), amount);
1001 _approveTokenIfNeeded(_from, routerAddr);
1002 _swapTokenToLP(_from, amount, _to, _recipient, routerAddr);
1003 return;
1004 }
1005 }
```

Risk Level:

Likelihood - 3

Impact - 5

Recommendation:

Use the `safeTransfer` function from the `safeERC20` Implementation, or put the transfer call inside and `assert` or `require` to verify that it returned `true`.

Status - Fixed

The DefySwap team has fixed the issue by adding a `require` statement that verifies if the transfer has passed correctly

I.2 Owner Can Renounce Ownership [LOW]

Description:

Typically, the contract's owner is the account that deploys the contract. As a result, the owner is able to perform certain privileged activities on his behalf. The `renounceOwnership` function is used in smart contracts to renounce ownership. Otherwise, if the contract's ownership has not been transferred previously, it will never have an Owner, which is risky.

Code:

Listing 83: Zap.sol

```
966 contract Zap is Ownable, IZap {  
967     using SafeMath for uint256;  
968     using SafeERC20 for IERC20;
```

Risk Level:

Likelihood - 1

Impact - 3

Recommendation:

It is advised that the Owner cannot call `renounceOwnership` without first transferring ownership to a different address. Additionally, if a multi-signature wallet is utilized, executing the `renounceOwnership` method for two or more users should be confirmed. Alternatively, the `RenounceOwnership` functionality can be disabled by overriding it.

Status - Acknowledged

The DefySwap team has acknowledged the risk.

I.3 Missing Address Verification [LOW]

Description:

Certain functions lack a safety check in the address, the address-type argument should include a zero-address test, otherwise, the contract's functionality may become inaccessible.

Code:

Listing 84: Zap.sol

```
978 constructor(address _WNATIVE) Ownable() {
```

```
979 WNative = _WNative;  
980 }
```

Listing 85: Zap.sol

```
986 function zapInToken(address _from, uint256 amount, address _to,  
987 address routerAddr, address _recipient) external override {  
988 _approveTokenIfNeeded(_from, routerAddr);
```

Listing 86: Zap.sol

```
1003 function estimateZapInToken(address _from, address _to, address _router,  
    ↪ uint256 _amt  
1004 ) public view override returns (uint256, uint256) {  
1005 // get pairs for desired lp  
1006 if (  
1007 _from == IUniswapV2Pair(_to).token0()  
1008 _from == IUniswapV2Pair(_to).token1()  
1009 ) {
```

Listing 87: Zap.sol

```
1031 function zapIn(address _to, address routerAddr, address _recipient)  
    ↪ external payable override {  
1032 // from Native to an LP token through the specified router  
1033 _swapNativeToLP(_to, msg.value, _recipient, routerAddr);  
1034 }  
1035 function zapAcross(address _from, uint256 amount, address _toRouter,  
    ↪ address _reci) external override {  
1036 IERC20(_from).safeTransferFrom(msg.sender, address(this), amount);  
1037 IUniswapV2Pair pair = IUniswapV2Pair(_from);
```

Risk Level:

Likelihood - 1

Impact - 3

Recommendation:

It's recommended to undertake further validation prior to user-supplied data. The concerns can be resolved by utilizing a whitelist technique or a modifier.

Status - Fixed

The DefySwap team has fixed the issue by adding `require` statements that verify the addresses provided in the arguments.

4 Static Analysis (Slither)

Description:

ShellBoxes expanded the coverage of the specific contract areas using automated testing methodologies. Slither, a Solidity static analysis framework, was one of the tools used. Slither was run on all-scoped contracts in both text and binary formats. This tool can be used to test mathematical relationships between Solidity instances statically and variables that allow for the detection of errors or inconsistent usage of the contracts' APIs throughout the entire codebase.

Results:

```
DefyMaster.safeDefyTransfer(address,uint256) (DFYMaster.sol#2417-2424)  
  ↪ ignores return value by defy.transfer(_to,defyBal) (DFYMaster.sol  
  ↪ #2420)
```

```
DefyMaster.safeDefyTransfer(address,uint256) (DFYMaster.sol#2417-2424)  
  ↪ ignores return value by defy.transfer(_to,_amount) (DFYMaster.sol  
  ↪ #2422)
```

```
DefyMaster.safeSecondRTransfer(address,uint256) (DFYMaster.sol  
  ↪ #2427-2434) ignores return value by secondR.transfer(_to,  
  ↪ secondRBal) (DFYMaster.sol#2430)
```

```
DefyMaster.safeSecondRTransfer(address,uint256) (DFYMaster.sol  
  ↪ #2427-2434) ignores return value by secondR.transfer(_to,_amount)  
  ↪ (DFYMaster.sol#2432)
```

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation>
 ↪ #unchecked-transfer

```
DefyMaster.pendingDefy(uint256,address) (DFYMaster.sol#2093-2129)  
  ↪ performs a multiplication on the result of a division:  
  -defyReward = multiplier.mul(defyPerSec).mul(pool.allocPoint).div  
  ↪ (totalAllocPoint) (DFYMaster.sol#2120-2123)  
  -accDefyPerShare = accDefyPerShare.add(defyReward.mul(1e12).div(  
  ↪ lpSupply)) (DFYMaster.sol#2124-2126)
```

DefyMaster.pendingSecondR(uint256,address) (DFYMaster.sol#2132-2171)

↪ performs a multiplication on the result of a division:

```
-secondRReward = multiplier.mul(secondRPerSec).mul(pool.  
  ↪ allocPointDR).div(totalAllocPointDR) (DFYMaster.sol  
  ↪ #2159-2162)  
-accSecondRPerShare = accSecondRPerShare.add(secondRReward.mul(1  
  ↪ e12).div(lpSupply)) (DFYMaster.sol#2163-2165)
```

DefyMaster.updatePool(uint256) (DFYMaster.sol#2183-2268) performs a

↪ multiplication on the result of a division:

```
-defyReward = multiplier.mul(defyPerSec).mul(pool.allocPoint).div  
  ↪ (totalAllocPoint) (DFYMaster.sol#2236-2238)  
-pool.accDefyPerShare = pool.accDefyPerShare.add(defyReward.mul(1  
  ↪ e12).div(lpSupply)) (DFYMaster.sol#2252-2254)
```

DefyMaster.updatePool(uint256) (DFYMaster.sol#2183-2268) performs a

↪ multiplication on the result of a division:

```
-secondRReward = multiplier_scope_0.mul(secondRPerSec).mul(pool.  
  ↪ allocPointDR).div(totalAllocPointDR) (DFYMaster.sol  
  ↪ #2245-2248)  
-pool.accSecondRPerShare = pool.accSecondRPerShare.add(  
  ↪ secondRReward.mul(1e12).div(lpSupply)) (DFYMaster.sol  
  ↪ #2255-2257)
```

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation>

↪ #divide-before-multiply

DfyToken._writeCheckpoint(address,uint32,uint256,uint256) (DFYMaster.sol

↪ #1386-1411) uses a **dangerous** strict equality:

```
- nCheckpoints > 0 && checkpoints[delegatee][nCheckpoints - 1].  
  ↪ fromBlock == blockNumber (DFYMaster.sol#1398-1399)
```

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation>

↪ #dangerous-strict-equalities

Reentrancy in DefyMaster.add(uint256,uint256,IERC20,DefySTUB,IERC20,

↪ IERC20,uint256,uint256,bool,bool,uint256,bool) (DFYMaster.sol

↪ #1966-2029):

External calls:

- massUpdatePools() (DFYMaster.sol#1990)
 - burn_vault.burn() (DFYMaster.sol#1942)
 - defy.mint(address(this),defyReward) (DFYMaster.sol#2251)
- ilp.add(address(_lpToken),_token0,_token1,_offerILP) (DFYMaster
↪ .sol#1993)

State variables written after the call(s):

- poolInfo.push(PoolInfo(_lpToken,_stub,_allocPoint,_allocPointDR
↪ ,_depositFee,_withdrawalFee,lastRewardTimestamp,
↪ lastRewardTimestamp,_rewardEndTimestamp,0,0,0,_offerILP,
↪ _issueSTUB)) (DFYMaster.sol#1999-2016)
- totalAllocPoint = totalAllocPoint.add(_allocPoint) (DFYMaster.
↪ sol#1997)
- totalAllocPointDR = totalAllocPointDR.add(_allocPointDR) (
↪ DFYMaster.sol#1998)

Reentrancy in DefyMaster.deposit(uint256,uint256) (DFYMaster.sol
↪ #2271-2340):

External calls:

- updatePool(_pid) (DFYMaster.sol#2274)
 - burn_vault.burn() (DFYMaster.sol#1942)
 - defy.mint(address(this),defyReward) (DFYMaster.sol#2251)
- safeDefyTransfer(msg.sender,pending) (DFYMaster.sol#2305)
 - defy.transfer(_to,defyBal) (DFYMaster.sol#2420)
 - defy.transfer(_to,_amount) (DFYMaster.sol#2422)
- safeSecondRTransfer(msg.sender,pending) (DFYMaster.sol#2308)
 - secondR.transfer(_to,secondRBal) (DFYMaster.sol#2430)
 - secondR.transfer(_to,_amount) (DFYMaster.sol#2432)
- ilp.defyTransfer(msg.sender,extraDefy.sub(pending)) (DFYMaster.
↪ sol#2311)
- pool.lpToken.safeTransferFrom(address(msg.sender),address(this)
↪ ,amount_) (DFYMaster.sol#2316-2320)
- pool.lpToken.safeTransfer(feeAddress,depositFee) (DFYMaster.sol
↪ #2326)

State variables written after the call(s):

```

- pool.lpSupply = pool.lpSupply.add(amount_).sub(depositFee) (
  ↪ DFYMaster.sol#2328)
- user.amount = user.amount.add(amount_).sub(depositFee) (
  ↪ DFYMaster.sol#2327)

```

Reentrancy in DefyMaster.deposit(uint256,uint256) (DFYMaster.sol
 ↪ #2271-2340):

External calls:

```

- updatePool(_pid) (DFYMaster.sol#2274)
  - burn_vault.burn() (DFYMaster.sol#1942)
  - defy.mint(address(this),defyReward) (DFYMaster.sol#2251)
- safeDefyTransfer(msg.sender,pending) (DFYMaster.sol#2305)
  - defy.transfer(_to,defyBal) (DFYMaster.sol#2420)
  - defy.transfer(_to,_amount) (DFYMaster.sol#2422)
- safeSecondRTransfer(msg.sender,pending) (DFYMaster.sol#2308)
  - secondR.transfer(_to,secondRBal) (DFYMaster.sol#2430)
  - secondR.transfer(_to,_amount) (DFYMaster.sol#2432)
- ilp.defyTransfer(msg.sender,extraDefy.sub(pending)) (DFYMaster.
  ↪ sol#2311)
- pool.lpToken.safeTransferFrom(address(msg.sender),address(this)
  ↪ ,amount_) (DFYMaster.sol#2316-2320)

```

State variables written after the call(s):

```

- pool.lpSupply = pool.lpSupply.add(amount_) (DFYMaster.sol#2331)
- user.amount = user.amount.add(amount_) (DFYMaster.sol#2330)

```

Reentrancy in DefyMaster.emergencyWithdraw(uint256) (DFYMaster.sol
 ↪ #2405-2414):

External calls:

```

- pool.lpToken.safeTransfer(address(msg.sender),user.amount) (
  ↪ DFYMaster.sol#2408)

```

State variables written after the call(s):

```

- pool.lpSupply = pool.lpSupply.sub(user.amount) (DFYMaster.sol
  ↪ #2409)
- user.amount = 0 (DFYMaster.sol#2411)
- user.rewardDebt = 0 (DFYMaster.sol#2412)
- user.rewardDebtDR = 0 (DFYMaster.sol#2413)

```



```
Reentrancy in DefyMaster.set(uint256,uint256,uint256,IERC20,IERC20,  
  ↪ uint256,uint256,bool,bool,uint256,bool) (DFYMaster.sol#2032-2081)  
↪ :
```

External calls:

- massUpdatePools() (DFYMaster.sol#2053)
 - burn_vault.burn() (DFYMaster.sol#1942)
 - defy.mint(address(this),defyReward) (DFYMaster.sol#2251)
- ilp.set(_pid,_token0,_token1,_offerILP) (DFYMaster.sol#2055)

State variables written after the call(s):

- poolInfo[_pid].allocPoint = _allocPoint (DFYMaster.sol#2063)
- poolInfo[_pid].allocPointDR = _allocPointDR (DFYMaster.sol
 ↪ #2064)
- poolInfo[_pid].depositFee = _depositFee (DFYMaster.sol#2065)
- poolInfo[_pid].withdrawalFee = _withdrawalFee (DFYMaster.sol
 ↪ #2066)
- poolInfo[_pid].rewardEndTimestamp = _rewardEndTimestamp (
 ↪ DFYMaster.sol#2067)
- poolInfo[_pid].impermanentLossProtection = _offerILP (DFYMaster
 ↪ .sol#2068)
- poolInfo[_pid].issueStub = _issueSTUB (DFYMaster.sol#2069)
- totalAllocPoint = totalAllocPoint.sub(poolInfo[_pid].allocPoint
 ↪).add(_allocPoint) (DFYMaster.sol#2057-2059)
- totalAllocPointDR = totalAllocPointDR.sub(poolInfo[_pid].
 ↪ allocPointDR).add(_allocPointDR) (DFYMaster.sol#2060-2062)

```
Reentrancy in DefyMaster.updatePool(uint256) (DFYMaster.sol#2183-2268):
```

External calls:

- defy.mint(address(this),defyReward) (DFYMaster.sol#2251)

State variables written after the call(s):

- pool.accDefyPerShare = pool.accDefyPerShare.add(defyReward.mul
 ↪ (1e12).div(lpSupply)) (DFYMaster.sol#2252-2254)
- pool.accSecondRPerShare = pool.accSecondRPerShare.add(
 ↪ secondRReward.mul(1e12).div(lpSupply)) (DFYMaster.sol
 ↪ #2255-2257)
- pool.lastRewardTimestamp = blockTimestamp (DFYMaster.sol#2260)

```
- pool.lastRewardTimestampDR = blockTimestampDR (DFYMaster.sol  
  ↪ #2261)
```

Reentrancy in DefyMaster.updatePool(uint256) (DFYMaster.sol#2183-2268):

External calls:

```
- defy.mint(address(this),defyReward) (DFYMaster.sol#2251)  
- updateReward() (DFYMaster.sol#2263)  
  - burn_vault.burn() (DFYMaster.sol#1942)
```

State variables written after the call(s):

```
- updateReward() (DFYMaster.sol#2263)  
  - defyPerSec = burnAmount.div(SECONDS_PER_CYCLE) (  
    ↪ DFYMaster.sol#1938)  
- updateReward() (DFYMaster.sol#2263)  
  - nextCycleTimestamp = (block.timestamp).add(  
    ↪ SECONDS_PER_CYCLE) (DFYMaster.sol#1940)  
- pool.lastRewardTimestamp = block.timestamp (DFYMaster.sol#2265)  
- pool.lastRewardTimestampDR = block.timestamp (DFYMaster.sol  
  ↪ #2266)
```

Reentrancy in DefyMaster.withdraw(uint256,uint256) (DFYMaster.sol
 ↪ #2343-2402):

External calls:

```
- updatePool(_pid) (DFYMaster.sol#2347)  
  - burn_vault.burn() (DFYMaster.sol#1942)  
  - defy.mint(address(this),defyReward) (DFYMaster.sol#2251)  
- safeDefyTransfer(msg.sender,pending) (DFYMaster.sol#2381)  
  - defy.transfer(_to,defyBal) (DFYMaster.sol#2420)  
  - defy.transfer(_to,_amount) (DFYMaster.sol#2422)  
- safeSecondRTransfer(msg.sender,pending) (DFYMaster.sol#2384)  
  - secondR.transfer(_to,secondRBal) (DFYMaster.sol#2430)  
  - secondR.transfer(_to,_amount) (DFYMaster.sol#2432)  
- ilp.defyTransfer(msg.sender,extraDefy.sub(pending)) (DFYMaster.  
  ↪ sol#2387)
```

State variables written after the call(s):

```
- user.amount = user.amount.sub(amount_) (DFYMaster.sol#2391)
```

Reentrancy in DefyMaster.withdraw(uint256,uint256) (DFYMaster.sol

↪ #2343-2402):

External calls:

- updatePool(_pid) (DFYMaster.sol#2347)
 - burn_vault.burn() (DFYMaster.sol#1942)
 - defy.mint(address(this),defyReward) (DFYMaster.sol#2251)
- safeDefyTransfer(msg.sender,pending) (DFYMaster.sol#2381)
 - defy.transfer(_to,defyBal) (DFYMaster.sol#2420)
 - defy.transfer(_to,_amount) (DFYMaster.sol#2422)
- safeSecondRTransfer(msg.sender,pending) (DFYMaster.sol#2384)
 - secondR.transfer(_to,secondRBal) (DFYMaster.sol#2430)
 - secondR.transfer(_to,_amount) (DFYMaster.sol#2432)
- ilp.defyTransfer(msg.sender,extraDefy.sub(pending)) (DFYMaster.
↪ sol#2387)
- pool.lpToken.safeTransfer(address(msg.sender),amount_) (
↪ DFYMaster.sol#2392)

State variables written after the call(s):

- pool.lpSupply = pool.lpSupply.sub(amount_) (DFYMaster.sol#2393)
- user.depVal = ilp.getDepositValue(user.amount,_pid) (DFYMaster.
↪ sol#2395)
- user.depositTime = block.timestamp (DFYMaster.sol#2396)
- user.rewardDebt = user.amount.mul(pool.accDefyPerShare).div(1
↪ e12) (DFYMaster.sol#2397)
- user.rewardDebtDR = user.amount.mul(pool.accSecondRPerShare).
↪ div(1e12) (DFYMaster.sol#2398)

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation>

↪ #reentrancy-vulnerabilities-1

ERC20.constructor(string,string).name (DFYMaster.sol#687) shadows:

- ERC20.name() (DFYMaster.sol#703-705) (function)
- IERC20.name() (DFYMaster.sol#129) (function)

ERC20.constructor(string,string).symbol (DFYMaster.sol#687) shadows:

- ERC20.symbol() (DFYMaster.sol#717-719) (function)
- IERC20.symbol() (DFYMaster.sol#124) (function)

ERC20.allowance(address,address).owner (DFYMaster.sol#755) shadows:

- Ownable.owner() (DFYMaster.sol#64-66) (function)

ERC20._approve(address,address,uint256).owner (DFYMaster.sol#958)

↳ shadows:

- Ownable.owner() (DFYMaster.sol#64-66) (function)

DefyMaster.getUserInfo(uint256,address).deposit (DFYMaster.sol#1852)

↳ shadows:

- DefyMaster.deposit(uint256,uint256) (DFYMaster.sol#2271-2340) (

- ↳ function)

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation>

↳ #local-variable-shadowing

DefyMaster.dev(address) (DFYMaster.sol#2437-2441) should emit an event

↳ for:

- devaddr = _devaddr (DFYMaster.sol#2440)

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation>

↳ #missing-events-access-control

DefyMaster.updateMultiplier(uint256) (DFYMaster.sol#1922-1925) should

↳ emit an event for:

- BONUS_MULTIPLIER = multiplierNumber (DFYMaster.sol#1924)

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation>

↳ #missing-events-arithmetic

DfyToken.constructor(address,address,uint256)._dev (DFYMaster.sol#1028)

↳ lacks a zero-check on :

- dev = _dev (DFYMaster.sol#1034)

DfyToken.constructor(address,address,uint256)._bunVault (DFYMaster.sol

↳ #1029) lacks a zero-check on :

- BURN_VAULT = _bunVault (DFYMaster.sol#1035)

DfyToken.setDev(address)._dev (DFYMaster.sol#1104) lacks a zero-check on

↳ :

- dev = _dev (DFYMaster.sol#1106)

DfyToken.setBurnVault(address)._burnVault (DFYMaster.sol#1111) lacks a
↪ zero-check on :

- BURN_VAULT = _burnVault (DFYMaster.sol#1112)

DfyToken.setIlpVault(address)._ilpVault (DFYMaster.sol#1117) lacks a
↪ zero-check on :

- ILP_VAULT = _ilpVault (DFYMaster.sol#1118)

DfyToken.setMaster(address).master (DFYMaster.sol#1123) lacks a zero-
↪ check on :

- defyMaster = master (DFYMaster.sol#1124)

BurnVault.setDefyMaster(address).master (DFYMaster.sol#1581) lacks a
↪ zero-check on :

- defyMaster = master (DFYMaster.sol#1582)

DefyMaster.constructor(DfyToken,DefySTUB,BurnVault,
↪ ImpermanentLossProtection,address,address,uint256,uint256).

↪ _devaddr (DFYMaster.sol#1775) lacks a zero-check on :

- devaddr = _devaddr (DFYMaster.sol#1784)

DefyMaster.constructor(DfyToken,DefySTUB,BurnVault,
↪ ImpermanentLossProtection,address,address,uint256,uint256).

↪ _feeAddress (DFYMaster.sol#1776) lacks a zero-check on :

- feeAddress = _feeAddress (DFYMaster.sol#1785)

DefyMaster.setFeeAddress(address)._feeAddress (DFYMaster.sol#1823) lacks
↪ a zero-check on :

- feeAddress = _feeAddress (DFYMaster.sol#1825)

DefyMaster.dev(address)._devaddr (DFYMaster.sol#2437) lacks a zero-check
↪ on :

- devaddr = _devaddr (DFYMaster.sol#2440)

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation>
↪ #missing-zero-address-validation

DefyMaster.updatePool(uint256) (DFYMaster.sol#2183-2268) has external
↪ calls inside a loop: defy.mint(address(this),defyReward) (
↪ DFYMaster.sol#2251)

DefyMaster.updateReward() (DFYMaster.sol#1936-1945) has external calls
↪ inside a loop: burnAmount = defy.balanceOf(address(burn_vault)) (

```
↪ DFYMaster.sol#1937)
DefyMaster.updateReward() (DFYMaster.sol#1936-1945) has external calls
↪ inside a loop: burn_vault.burn() (DFYMaster.sol#1942)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
↪ /#calls-inside-a-loop
```

```
Reentrancy in DefyMaster.add(uint256,uint256,IERC20,DefySTUB,IERC20,
↪ IERC20,uint256,uint256,bool,bool,uint256,bool) (DFYMaster.sol
↪ #1966-2029):
```

```
External calls:
```

- massUpdatePools() (DFYMaster.sol#1990)
 - burn_vault.burn() (DFYMaster.sol#1942)
 - defy.mint(address(this),defyReward) (DFYMaster.sol#2251)
- ilp.add(address(_lpToken),_token0,_token1,_offerILP) (DFYMaster
↪ .sol#1993)

```
Event emitted after the call(s):
```

- addPool(poolInfo.length - 1,address(_lpToken),_allocPoint,
↪ _allocPointDR,_depositFee,_withdrawalFee,_offerILP,
↪ _issueSTUB,_rewardEndTimestamp) (DFYMaster.sol#2018-2028)

```
Reentrancy in BurnVault.burn() (DFYMaster.sol#1590-1594):
```

```
External calls:
```

- defy.burn(amount) (DFYMaster.sol#1592)

```
Event emitted after the call(s):
```

- Burn(amount) (DFYMaster.sol#1593)

```
Reentrancy in BurnVault.burnPortion(uint256) (DFYMaster.sol#1596-1599):
```

```
External calls:
```

- defy.burn(amount) (DFYMaster.sol#1597)

```
Event emitted after the call(s):
```

- Burn(amount) (DFYMaster.sol#1598)

```
Reentrancy in DefyMaster.deposit(uint256,uint256) (DFYMaster.sol
↪ #2271-2340):
```

```
External calls:
```

- updatePool(_pid) (DFYMaster.sol#2274)
 - burn_vault.burn() (DFYMaster.sol#1942)

```

- defy.mint(address(this),defyReward) (DFYMaster.sol#2251)
- safeDefyTransfer(msg.sender,pending) (DFYMaster.sol#2305)
  - defy.transfer(_to,defyBal) (DFYMaster.sol#2420)
  - defy.transfer(_to,_amount) (DFYMaster.sol#2422)
- safeSecondRTransfer(msg.sender,pending) (DFYMaster.sol#2308)
  - secondR.transfer(_to,secondRBal) (DFYMaster.sol#2430)
  - secondR.transfer(_to,_amount) (DFYMaster.sol#2432)
- ilp.defyTransfer(msg.sender,extraDefy.sub(pending)) (DFYMaster.
  ↪ sol#2311)
- pool.lpToken.safeTransferFrom(address(msg.sender),address(this)
  ↪ ,amount_) (DFYMaster.sol#2316-2320)
- pool.lpToken.safeTransfer(feeAddress,depositFee) (DFYMaster.sol
  ↪ #2326)
- pool.stubToken.mint(msg.sender,amount_) (DFYMaster.sol#2338)
Event emitted after the call(s):
- Deposit(msg.sender,_pid,amount_) (DFYMaster.sol#2339)
Reentrancy in DefyMaster.emergencyWithdraw(uint256) (DFYMaster.sol
  ↪ #2405-2414):
  External calls:
  - pool.lpToken.safeTransfer(address(msg.sender),user.amount) (
    ↪ DFYMaster.sol#2408)
  Event emitted after the call(s):
  - EmergencyWithdraw(msg.sender,_pid,user.amount) (DFYMaster.sol
    ↪ #2410)
Reentrancy in DefyMaster.set(uint256,uint256,uint256,IERC20,IERC20,
  ↪ uint256,uint256,bool,bool,uint256,bool) (DFYMaster.sol#2032-2081)
  ↪ :
  External calls:
  - massUpdatePools() (DFYMaster.sol#2053)
    - burn_vault.burn() (DFYMaster.sol#1942)
    - defy.mint(address(this),defyReward) (DFYMaster.sol#2251)
  - ilp.set(_pid,_token0,_token1,_offerILP) (DFYMaster.sol#2055)
  Event emitted after the call(s):

```

```

- setPool(_pid,_allocPoint,_allocPointDR,_depositFee,
  ↪ _withdrawalFee,_offerILP,_issueSTUB,_rewardEndTimestamp) (
  ↪ DFYMaster.sol#2071-2080)
Reentrancy in DefyMaster.updatePool(uint256) (DFYMaster.sol#2183-2268):
  External calls:
- defy.mint(address(this),defyReward) (DFYMaster.sol#2251)
- updateReward() (DFYMaster.sol#2263)
  - burn_vault.burn() (DFYMaster.sol#1942)
  Event emitted after the call(s):
- UpdateEmissionRate(defyPerSec) (DFYMaster.sol#1944)
  - updateReward() (DFYMaster.sol#2263)
Reentrancy in DefyMaster.updateReward() (DFYMaster.sol#1936-1945):
  External calls:
- burn_vault.burn() (DFYMaster.sol#1942)
  Event emitted after the call(s):
- UpdateEmissionRate(defyPerSec) (DFYMaster.sol#1944)
Reentrancy in DefyMaster.withdraw(uint256,uint256) (DFYMaster.sol
  ↪ #2343-2402):
  External calls:
- updatePool(_pid) (DFYMaster.sol#2347)
  - burn_vault.burn() (DFYMaster.sol#1942)
  - defy.mint(address(this),defyReward) (DFYMaster.sol#2251)
- safeDefyTransfer(msg.sender,pending) (DFYMaster.sol#2381)
  - defy.transfer(_to,defyBal) (DFYMaster.sol#2420)
  - defy.transfer(_to,_amount) (DFYMaster.sol#2422)
- safeSecondRTransfer(msg.sender,pending) (DFYMaster.sol#2384)
  - secondR.transfer(_to,secondRBal) (DFYMaster.sol#2430)
  - secondR.transfer(_to,_amount) (DFYMaster.sol#2432)
- ilp.defyTransfer(msg.sender,extraDefy.sub(pending)) (DFYMaster.
  ↪ sol#2387)
- pool.lpToken.safeTransfer(address(msg.sender),amount_) (
  ↪ DFYMaster.sol#2392)
- pool.stubToken.burn(msg.sender,amount_) (DFYMaster.sol#2399)
  Event emitted after the call(s):

```


- Withdraw(msg.sender, _pid, amount_) (DFYMaster.sol#2401)

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation>

↪ #reentrancy-vulnerabilities-3

DfyToken.delegateBySig(address, uint256, uint256, uint8, bytes32, bytes32) (

↪ DFYMaster.sol#1251-1287) uses timestamp for comparisons

Dangerous comparisons:

- require(bool, string)(now <= expiry, DEFY::delegateBySig:

↪ signature expired) (DFYMaster.sol#1285)

DefyMaster._getDaysSinceDeposit(uint256, address) (DFYMaster.sol

↪ #1878-1891) uses timestamp for comparisons

Dangerous comparisons:

- block.timestamp < user.depositTime (DFYMaster.sol#1885)

DefyMaster.updateSecondReward(uint256, uint256) (DFYMaster.sol#1947-1957)

↪ uses timestamp for comparisons

Dangerous comparisons:

- require(bool, string)(_endTimeStamp > block.timestamp, invalid

↪ End timestamp) (DFYMaster.sol#1951)

DefyMaster.add(uint256, uint256, IERC20, DefySTUB, IERC20, IERC20, uint256,

↪ uint256, bool, bool, uint256, bool) (DFYMaster.sol#1966-2029) uses

↪ timestamp for comparisons

Dangerous comparisons:

- require(bool, string)(_rewardEndTimeStamp > block.timestamp, Add:

↪ invalid rewardEndTimeStamp) (DFYMaster.sol#1984-1987)

- block.timestamp > startTimestamp (DFYMaster.sol#1994-1996)

DefyMaster.set(uint256, uint256, uint256, IERC20, IERC20, uint256, uint256,

↪ bool, bool, uint256, bool) (DFYMaster.sol#2032-2081) uses timestamp

↪ for comparisons

Dangerous comparisons:

- require(bool, string)(_rewardEndTimeStamp > block.timestamp, Add:

↪ invalid rewardEndTimeStamp) (DFYMaster.sol#2047-2050)

DefyMaster.pendingDefy(uint256, address) (DFYMaster.sol#2093-2129) uses

↪ timestamp for comparisons

Dangerous comparisons:

- `block.timestamp > pool.lastRewardTimestamp && lpSupply != 0 &&`
 ↳ `totalAllocPoint != 0 (DFYMaster.sol#2103-2105)`
- `block.timestamp < nextCycleTimestamp (DFYMaster.sol#2109)`
- `block.timestamp < pool.rewardEndTimestamp (DFYMaster.sol`
 ↳ `#2110-2112)`

`DefyMaster.pendingSecondR(uint256,address) (DFYMaster.sol#2132-2171)`

↳ uses timestamp for comparisons

Dangerous comparisons:

- `block.timestamp > pool.lastRewardTimestampDR && lpSupply != 0`
 ↳ `&& totalAllocPointDR != 0 (DFYMaster.sol#2142-2144)`
- `block.timestamp < endTimestampDR (DFYMaster.sol#2148)`
- `block.timestamp < pool.rewardEndTimestamp (DFYMaster.sol`
 ↳ `#2149-2151)`

`DefyMaster.massUpdatePools() (DFYMaster.sol#2174-2180)` uses timestamp

↳ for comparisons

Dangerous comparisons:

- `pid < length (DFYMaster.sol#2176)`

`DefyMaster.updatePool(uint256) (DFYMaster.sol#2183-2268)` uses timestamp

↳ for comparisons

Dangerous comparisons:

- `block.timestamp <= pool.lastRewardTimestamp && block.timestamp`
 ↳ `<= pool.lastRewardTimestampDR (DFYMaster.sol#2188-2189)`
- `block.timestamp < nextCycleTimestamp (DFYMaster.sol#2198)`
- `block.timestamp < endTimestampDR (DFYMaster.sol#2208)`
- `totalAllocPoint != 0 (DFYMaster.sol#2231)`
- `totalAllocPointDR != 0 (DFYMaster.sol#2240)`
- `block.timestamp > nextCycleTimestamp (DFYMaster.sol#2184)`
- `block.timestamp < pool.rewardEndTimestamp (DFYMaster.sol`
 ↳ `#2199-2201)`
- `block.timestamp < pool.rewardEndTimestamp (DFYMaster.sol`
 ↳ `#2209-2211)`

`DefyMaster.deposit(uint256,uint256) (DFYMaster.sol#2271-2340)` uses

↳ timestamp for comparisons

Dangerous comparisons:

- pool.impermanentLossProtection && user.amount > 0 &&
 ↪ _getDaysSinceDeposit(_pid,msg.sender) >= 30 (DFYMaster.sol
 ↪ #2286-2288)

DefyMaster.withdraw(uint256,uint256) (DFYMaster.sol#2343-2402) uses

↪ timestamp for comparisons

Dangerous comparisons:

- require(bool,string)(user.amount > 0,withdraw: nothing to
 ↪ withdraw) (DFYMaster.sol#2346)
- pool.impermanentLossProtection && user.amount > 0 &&
 ↪ _getDaysSinceDeposit(_pid,msg.sender) >= 30 (DFYMaster.sol
 ↪ #2364-2366)

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation>

↪ #block-timestamp

Address.isContract(address) (DFYMaster.sol#423-434) uses assembly

- INLINE ASM (DFYMaster.sol#430-432)

Address._functionCallWithValue(address,bytes,uint256,string) (DFYMaster.

↪ sol#549-577) uses assembly

- INLINE ASM (DFYMaster.sol#569-572)

DfyToken.getChainId() (DFYMaster.sol#1422-1428) uses assembly

- INLINE ASM (DFYMaster.sol#1424-1426)

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation>

↪ #assembly-usage

DefyMaster.updateReward() (DFYMaster.sol#1936-1945) has costly

↪ operations inside a loop:

- defyPerSec = burnAmount.div(SECONDS_PER_CYCLE) (DFYMaster.sol
 ↪ #1938)

DefyMaster.updateReward() (DFYMaster.sol#1936-1945) has costly

↪ operations inside a loop:

- nextCycleTimestamp = (block.timestamp).add(SECONDS_PER_CYCLE) (
 ↪ DFYMaster.sol#1940)

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation>

↪ #costly-operations-inside-a-loop

`Address.functionCall(address,bytes)` (DFYMaster.sol#484-489) is never
↳ used and should be removed

`Address.functionCallWithValue(address,bytes,uint256)` (DFYMaster.sol
↳ #516-528) is never used and should be removed

`Address.functionCallWithValue(address,bytes,uint256,string)` (DFYMaster.
↳ sol#536-547) is never used and should be removed

`Address.sendValue(address,uint256)` (DFYMaster.sol#452-464) is never used
↳ and should be removed

`Context._msgData()` (DFYMaster.sol#25-28) is never used and should be
↳ removed

`ERC20._burnFrom(address,uint256)` (DFYMaster.sol#975-985) is never used
↳ and should be removed

`SafeERC20.safeApprove(IERC20,address,uint256)` (DFYMaster.sol#1475-1492)
↳ is never used and should be removed

`SafeERC20.safeDecreaseAllowance(IERC20,address,uint256)` (DFYMaster.sol
↳ #1512-1529) is never used and should be removed

`SafeERC20.safeIncreaseAllowance(IERC20,address,uint256)` (DFYMaster.sol
↳ #1494-1510) is never used and should be removed

`SafeMath.min(uint256,uint256)` (DFYMaster.sol#382-384) is never used and
↳ should be removed

`SafeMath.mod(uint256,uint256)` (DFYMaster.sol#357-359) is never used and
↳ should be removed

`SafeMath.mod(uint256,uint256,string)` (DFYMaster.sol#373-380) is never
↳ used and should be removed

`SafeMath.sqrt(uint256)` (DFYMaster.sol#387-398) is never used and should
↳ be removed

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation>
↳ #dead-code

Low level call in `Address.sendValue(address,uint256)` (DFYMaster.sol
↳ #452-464):

- (success) = recipient.call{value: amount}() (DFYMaster.sol#459)

Low level call in `Address._functionCallWithValue(address,bytes,uint256, ↪ string)` (DFYMaster.sol#549-577):
- (success, returndata) = target.call{value: weiValue}(data) (↪ DFYMaster.sol#558-560)

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation>
↪ #low-level-calls

Parameter `DfyToken.mint(address,uint256)._to` (DFYMaster.sol#1043) is not
↪ in mixedCase

Parameter `DfyToken.mint(address,uint256)._amount` (DFYMaster.sol#1043) is
↪ not in mixedCase

Parameter `DfyToken.setDev(address)._dev` (DFYMaster.sol#1104) is not in
↪ mixedCase

Parameter `DfyToken.setBurnVault(address)._burnVault` (DFYMaster.sol#1111)
↪ is not in mixedCase

Parameter `DfyToken.setIlpVault(address)._ilpVault` (DFYMaster.sol#1117)
↪ is not in mixedCase

Variable `DfyToken._burnFee` (DFYMaster.sol#992) is not in mixedCase

Variable `DfyToken._ilpFee` (DFYMaster.sol#993) is not in mixedCase

Variable `DfyToken._devFee` (DFYMaster.sol#994) is not in mixedCase

Variable `DfyToken._maxTxAmount` (DFYMaster.sol#996) is not in mixedCase

Variable `DfyToken._maxSupply` (DFYMaster.sol#997) is not in mixedCase

Variable `DfyToken.BURN_VAULT` (DFYMaster.sol#999) is not in mixedCase

Variable `DfyToken.ILP_VAULT` (DFYMaster.sol#1000) is not in mixedCase

Variable `DfyToken._delegates` (DFYMaster.sol#1185) is not in mixedCase

Parameter `BurnVault.setDefy(address)._defy` (DFYMaster.sol#1586) is not
↪ in mixedCase

Event `DefyMasteraddPool(uint256,address,uint256,uint256,uint256,uint256, ↪ bool,bool,uint256)` (DFYMaster.sol#1745-1755) is not in CapWords

Event `DefyMastersetPool(uint256,uint256,uint256,uint256,uint256,bool, ↪ bool,uint256)` (DFYMaster.sol#1757-1766) is not in CapWords

Parameter `DefyMaster.setImpermanentLossProtection(address)._ilp` (↪ DFYMaster.sol#1814) is not in mixedCase

Parameter DefyMaster.setFeeAddress(address)._feeAddress (DFYMaster.sol
↳ #1823) is not in mixedCase

Parameter DefyMaster.setDFY(DfyToken)._dfy (DFYMaster.sol#1830) is not
↳ in mixedCase

Parameter DefyMaster.setSecondaryReward(IERC20)._rewardToken (DFYMaster.
↳ sol#1837) is not in mixedCase

Parameter DefyMaster.updateSecondReward(uint256,uint256)._reward (
↳ DFYMaster.sol#1947) is not in mixedCase

Parameter DefyMaster.updateSecondReward(uint256,uint256)._endTimeStamp (
↳ DFYMaster.sol#1947) is not in mixedCase

Parameter DefyMaster.add(uint256,uint256,IERC20,DefySTUB,IERC20,IERC20,
↳ uint256,uint256,bool,bool,uint256,bool)._allocPoint (DFYMaster.
↳ sol#1967) is not in mixedCase

Parameter DefyMaster.add(uint256,uint256,IERC20,DefySTUB,IERC20,IERC20,
↳ uint256,uint256,bool,bool,uint256,bool)._allocPointDR (DFYMaster.
↳ sol#1968) is not in mixedCase

Parameter DefyMaster.add(uint256,uint256,IERC20,DefySTUB,IERC20,IERC20,
↳ uint256,uint256,bool,bool,uint256,bool)._lpToken (DFYMaster.sol
↳ #1969) is not in mixedCase

Parameter DefyMaster.add(uint256,uint256,IERC20,DefySTUB,IERC20,IERC20,
↳ uint256,uint256,bool,bool,uint256,bool)._stub (DFYMaster.sol
↳ #1970) is not in mixedCase

Parameter DefyMaster.add(uint256,uint256,IERC20,DefySTUB,IERC20,IERC20,
↳ uint256,uint256,bool,bool,uint256,bool)._token0 (DFYMaster.sol
↳ #1971) is not in mixedCase

Parameter DefyMaster.add(uint256,uint256,IERC20,DefySTUB,IERC20,IERC20,
↳ uint256,uint256,bool,bool,uint256,bool)._token1 (DFYMaster.sol
↳ #1972) is not in mixedCase

Parameter DefyMaster.add(uint256,uint256,IERC20,DefySTUB,IERC20,IERC20,
↳ uint256,uint256,bool,bool,uint256,bool)._depositFee (DFYMaster.
↳ sol#1973) is not in mixedCase

Parameter DefyMaster.add(uint256,uint256,IERC20,DefySTUB,IERC20,IERC20,
↳ uint256,uint256,bool,bool,uint256,bool)._withdrawalFee (DFYMaster
↳ .sol#1974) is not in mixedCase

Parameter DefyMaster.add(uint256,uint256,IERC20,DefySTUB,IERC20,IERC20,
↳ uint256,uint256,bool,bool,uint256,bool)._offerILP (DFYMaster.sol
↳ #1975) is not in mixedCase

Parameter DefyMaster.add(uint256,uint256,IERC20,DefySTUB,IERC20,IERC20,
↳ uint256,uint256,bool,bool,uint256,bool)._issueSTUB (DFYMaster.sol
↳ #1976) is not in mixedCase

Parameter DefyMaster.add(uint256,uint256,IERC20,DefySTUB,IERC20,IERC20,
↳ uint256,uint256,bool,bool,uint256,bool)._rewardEndTimestamp (
↳ DFYMaster.sol#1977) is not in mixedCase

Parameter DefyMaster.add(uint256,uint256,IERC20,DefySTUB,IERC20,IERC20,
↳ uint256,uint256,bool,bool,uint256,bool)._withUpdate (DFYMaster.
↳ sol#1978) is not in mixedCase

Parameter DefyMaster.set(uint256,uint256,uint256,IERC20,IERC20,uint256,
↳ uint256,bool,bool,uint256,bool)._pid (DFYMaster.sol#2033) is not
↳ in mixedCase

Parameter DefyMaster.set(uint256,uint256,uint256,IERC20,IERC20,uint256,
↳ uint256,bool,bool,uint256,bool)._allocPoint (DFYMaster.sol#2034)
↳ is not in mixedCase

Parameter DefyMaster.set(uint256,uint256,uint256,IERC20,IERC20,uint256,
↳ uint256,bool,bool,uint256,bool)._allocPointDR (DFYMaster.sol
↳ #2035) is not in mixedCase

Parameter DefyMaster.set(uint256,uint256,uint256,IERC20,IERC20,uint256,
↳ uint256,bool,bool,uint256,bool)._token0 (DFYMaster.sol#2036) is
↳ not in mixedCase

Parameter DefyMaster.set(uint256,uint256,uint256,IERC20,IERC20,uint256,
↳ uint256,bool,bool,uint256,bool)._token1 (DFYMaster.sol#2037) is
↳ not in mixedCase

Parameter DefyMaster.set(uint256,uint256,uint256,IERC20,IERC20,uint256,
↳ uint256,bool,bool,uint256,bool)._depositFee (DFYMaster.sol#2038)
↳ is not in mixedCase

Parameter DefyMaster.set(uint256,uint256,uint256,IERC20,IERC20,uint256,
↳ uint256,bool,bool,uint256,bool)._withdrawalFee (DFYMaster.sol
↳ #2039) is not in mixedCase

Parameter DefyMaster.set(uint256,uint256,uint256,IERC20,IERC20,uint256,
↳ uint256,bool,bool,uint256,bool)._offerILP (DFYMaster.sol#2040) is
↳ not in mixedCase

Parameter DefyMaster.set(uint256,uint256,uint256,IERC20,IERC20,uint256,
↳ uint256,bool,bool,uint256,bool)._issueSTUB (DFYMaster.sol#2041)
↳ is not in mixedCase

Parameter DefyMaster.set(uint256,uint256,uint256,IERC20,IERC20,uint256,
↳ uint256,bool,bool,uint256,bool)._rewardEndTimestamp (DFYMaster.
↳ sol#2042) is not in mixedCase

Parameter DefyMaster.set(uint256,uint256,uint256,IERC20,IERC20,uint256,
↳ uint256,bool,bool,uint256,bool)._withUpdate (DFYMaster.sol#2043)
↳ is not in mixedCase

Parameter DefyMaster.getMultiplier(uint256,uint256)._from (DFYMaster.sol
↳ #2084) is not in mixedCase

Parameter DefyMaster.getMultiplier(uint256,uint256)._to (DFYMaster.sol
↳ #2084) is not in mixedCase

Parameter DefyMaster.pendingDefy(uint256,address)._pid (DFYMaster.sol
↳ #2093) is not in mixedCase

Parameter DefyMaster.pendingDefy(uint256,address)._user (DFYMaster.sol
↳ #2093) is not in mixedCase

Parameter DefyMaster.pendingSecondR(uint256,address)._pid (DFYMaster.sol
↳ #2132) is not in mixedCase

Parameter DefyMaster.pendingSecondR(uint256,address)._user (DFYMaster.
↳ sol#2132) is not in mixedCase

Parameter DefyMaster.updatePool(uint256)._pid (DFYMaster.sol#2183) is
↳ not in mixedCase

Parameter DefyMaster.deposit(uint256,uint256)._pid (DFYMaster.sol#2271)
↳ is not in mixedCase

Parameter DefyMaster.deposit(uint256,uint256)._amount (DFYMaster.sol
↳ #2271) is not in mixedCase

Parameter DefyMaster.withdraw(uint256,uint256)._pid (DFYMaster.sol#2343)
↳ is not in mixedCase

Parameter DefyMaster.withdraw(uint256,uint256)._amount (DFYMaster.sol
↳ #2343) is not in mixedCase

Parameter DefyMaster.emergencyWithdraw(uint256)._pid (DFYMaster.sol
↳ #2405) is not in mixedCase

Parameter DefyMaster.safeDefyTransfer(address,uint256)._to (DFYMaster.
↳ sol#2417) is not in mixedCase

Parameter DefyMaster.safeDefyTransfer(address,uint256)._amount (
↳ DFYMaster.sol#2417) is not in mixedCase

Parameter DefyMaster.safeSecondRTransfer(address,uint256)._to (DFYMaster
↳ .sol#2427) is not in mixedCase

Parameter DefyMaster.safeSecondRTransfer(address,uint256)._amount (
↳ DFYMaster.sol#2427) is not in mixedCase

Parameter DefyMaster.dev(address)._devaddr (DFYMaster.sol#2437) is not
↳ in mixedCase

Variable DefyMaster.burn_vault (DFYMaster.sol#1687) is not in mixedCase

Variable DefyMaster.BONUS_MULTIPLIER (DFYMaster.sol#1699) is not in
↳ mixedCase

Variable DefyMaster.SECONDS_PER_CYCLE (DFYMaster.sol#1703) is not in
↳ mixedCase

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation>
↳ #conformance-to-solidity-naming-conventions

Redundant expression "this (DFYMaster.sol#26)" inContext (DFYMaster.sol
↳ #16-29)

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation>
↳ #redundant-statements

DfyToken._maxSupply (DFYMaster.sol#997) should be constant

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation>
↳ #state-variables-that-could-be-declared-constant

renounceOwnership() should be declared external:

- Ownable.renounceOwnership() (DFYMaster.sol#83-86)

transferOwnership(address) should be declared external:

- Ownable.transferOwnership(address) (DFYMaster.sol#92-94)

decimals() should be declared external:

- ERC20.decimals() (DFYMaster.sol#710-712)

symbol() should be declared external:

- ERC20.symbol() (DFYMaster.sol#717-719)

transfer(address,uint256) should be declared external:

- ERC20.transfer(address,uint256) (DFYMaster.sol#743-750)

allowance(address,address) should be declared external:

- ERC20.allowance(address,address) (DFYMaster.sol#755-762)

approve(address,uint256) should be declared external:

- ERC20.approve(address,uint256) (DFYMaster.sol#771-778)

transferFrom(address,address,uint256) should be declared external:

- ERC20.transferFrom(address,address,uint256) (DFYMaster.sol
↪ #792-807)

increaseAllowance(address,uint256) should be declared external:

- ERC20.increaseAllowance(address,uint256) (DFYMaster.sol
↪ #821-831)

decreaseAllowance(address,uint256) should be declared external:

- ERC20.decreaseAllowance(address,uint256) (DFYMaster.sol
↪ #847-860)

mint(uint256) should be declared external:

- DfyToken.mint(uint256) (DFYMaster.sol#1052-1055)
- ERC20.mint(uint256) (DFYMaster.sol#870-873)

setMaster(address) should be declared external:

- DfyToken.setMaster(address) (DFYMaster.sol#1123-1126)

burnToVault(uint256) should be declared external:

- DfyToken.burnToVault(uint256) (DFYMaster.sol#1132-1134)

burn(uint256) should be declared external:

- DfyToken.burn(uint256) (DFYMaster.sol#1138-1141)

burn() should be declared external:

- BurnVault.burn() (DFYMaster.sol#1590-1594)

burnPortion(uint256) should be declared external:

- BurnVault.burnPortion(uint256) (DFYMaster.sol#1596-1599)

setImpermanentLossProtection(address) should be declared external:

- DefyMaster.setImpermanentLossProtection(address) (DFYMaster.sol
↪ #1814-1821)

`setFeeAddress(address)` should be declared `external`:

- `DefyMaster.setFeeAddress(address)` (DFYMaster.sol#1823-1828)

`setDFY(DfyToken)` should be declared `external`:

- `DefyMaster.setDFY(DfyToken)` (DFYMaster.sol#1830-1835)

`setSecondaryReward(IERC20)` should be declared `external`:

- `DefyMaster.setSecondaryReward(IERC20)` (DFYMaster.sol#1837-1846)

`getUserInfo(uint256,address)` should be declared `external`:

- `DefyMaster.getUserInfo(uint256,address)` (DFYMaster.sol
 ↪ #1848-1867)

`setStartTimestamp(uint256)` should be declared `external`:

- `DefyMaster.setStartTimestamp(uint256)` (DFYMaster.sol#1916-1920)

`updateMultiplier(uint256)` should be declared `external`:

- `DefyMaster.updateMultiplier(uint256)` (DFYMaster.sol#1922-1925)

`updateSecondReward(uint256,uint256)` should be declared `external`:

- `DefyMaster.updateSecondReward(uint256,uint256)` (DFYMaster.sol
 ↪ #1947-1957)

`add(uint256,uint256,IERC20,DefySTUB,IERC20,IERC20,uint256,uint256,bool,
 ↪ bool,uint256,bool)` should be declared `external`:

- `DefyMaster.add(uint256,uint256,IERC20,DefySTUB,IERC20,IERC20,
 ↪ uint256,uint256,bool,bool,uint256,bool)` (DFYMaster.sol
 ↪ #1966-2029)

`set(uint256,uint256,uint256,IERC20,IERC20,uint256,uint256,bool,bool,
 ↪ uint256,bool)` should be declared `external`:

- `DefyMaster.set(uint256,uint256,uint256,IERC20,IERC20,uint256,
 ↪ uint256,bool,bool,uint256,bool)` (DFYMaster.sol#2032-2081)

`deposit(uint256,uint256)` should be declared `external`:

- `DefyMaster.deposit(uint256,uint256)` (DFYMaster.sol#2271-2340)

`withdraw(uint256,uint256)` should be declared `external`:

- `DefyMaster.withdraw(uint256,uint256)` (DFYMaster.sol#2343-2402)

`emergencyWithdraw(uint256)` should be declared `external`:

- `DefyMaster.emergencyWithdraw(uint256)` (DFYMaster.sol#2405-2414)

`dev(address)` should be declared `external`:

- `DefyMaster.dev(address)` (DFYMaster.sol#2437-2441)

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation>
↳ #public-function-that-could-be-declared-external

DFYMaster.sol analyzed (13 contracts with 78 detectors), 176 result(s)
↳ found

DfyToken._writeCheckpoint(address,uint32,uint256,uint256) (DFYtoken.sol
↳ #1373-1398) uses a dangerous strict equality:

- nCheckpoints > 0 && checkpoints[delegatee][nCheckpoints - 1].
↳ fromBlock == blockNumber (DFYtoken.sol#1385-1386)

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation>
↳ #dangerous-strict-equalities

ERC20.constructor(string,string).name (DFYtoken.sol#628) shadows:

- ERC20.name() (DFYtoken.sol#644-646) (function)
- IERC20.name() (DFYtoken.sol#129) (function)

ERC20.constructor(string,string).symbol (DFYtoken.sol#628) shadows:

- ERC20.symbol() (DFYtoken.sol#658-660) (function)
- IERC20.symbol() (DFYtoken.sol#124) (function)

ERC20.allowance(address,address).owner (DFYtoken.sol#696) shadows:

- Ownable.owner() (DFYtoken.sol#64-66) (function)

ERC20._approve(address,address,uint256).owner (DFYtoken.sol#899) shadows

↳ :

- Ownable.owner() (DFYtoken.sol#64-66) (function)

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation>
↳ #local-variable-shadowing

DfyToken.constructor(address,address,uint256)._dev (DFYtoken.sol#971)

↳ lacks a zero-check on :

- dev = _dev (DFYtoken.sol#977)

DfyToken.constructor(address,address,uint256)._bunVault (DFYtoken.sol

↳ #972) lacks a zero-check on :

- BURN_VAULT = _bunVault (DFYtoken.sol#978)

DfyToken.setRouter(address)._router (DFYtoken.sol#1019) lacks a zero-

↳ check on :

```

- router = _router (DFYtoken.sol#1021)
DfyToken.setDev(address)._dev (DFYtoken.sol#1058) lacks a zero-check on
↳ :
- dev = _dev (DFYtoken.sol#1061)
DfyToken.setBurnVault(address)._burnVault (DFYtoken.sol#1066) lacks a
↳ zero-check on :
- BURN_VAULT = _burnVault (DFYtoken.sol#1068)
DfyToken.setIlpVault(address)._ilpVault (DFYtoken.sol#1073) lacks a zero
↳ -check on :
- ILP_VAULT = _ilpVault (DFYtoken.sol#1075)
DfyToken.setMaster(address).master (DFYtoken.sol#1080) lacks a zero-
↳ check on :
- defyMaster = master (DFYtoken.sol#1082)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
↳ #missing-zero-address-validation

DfyToken.delegateBySig(address,uint256,uint256,uint8,bytes32,bytes32) (
↳ DFYtoken.sol#1237-1274) uses timestamp for comparisons
Dangerous comparisons:
- require(bool,string)(now <= expiry,DEFY::delegateBySig:
↳ signature expired) (DFYtoken.sol#1272)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
↳ #block-timestamp

Address.isContract(address) (DFYtoken.sol#423-434) uses assembly
- INLINE ASM (DFYtoken.sol#430-432)
Address._functionCallWithValue(address,bytes,uint256,string) (DFYtoken.
↳ sol#549-577) uses assembly
- INLINE ASM (DFYtoken.sol#569-572)
DfyToken.getChainId() (DFYtoken.sol#1409-1415) uses assembly
- INLINE ASM (DFYtoken.sol#1411-1413)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
↳ #assembly-usage

```

`Address._functionCallWithValue(address,bytes,uint256,string)` (DFYtoken.

↳ sol#549-577) is never used and should be removed

`Address.functionCall(address,bytes)` (DFYtoken.sol#484-489) is never used

↳ and should be removed

`Address.functionCall(address,bytes,string)` (DFYtoken.sol#497-503) is

↳ never used and should be removed

`Address.functionCallWithValue(address,bytes,uint256)` (DFYtoken.sol

↳ #516-528) is never used and should be removed

`Address.functionCallWithValue(address,bytes,uint256,string)` (DFYtoken.

↳ sol#536-547) is never used and should be removed

`Address.isContract(address)` (DFYtoken.sol#423-434) is never used and

↳ should be removed

`Address.sendValue(address,uint256)` (DFYtoken.sol#452-464) is never used

↳ and should be removed

`Context._msgData()` (DFYtoken.sol#25-28) is never used and should be

↳ removed

`ERC20._burnFrom(address,uint256)` (DFYtoken.sol#916-926) is never used

↳ and should be removed

`SafeMath.min(uint256,uint256)` (DFYtoken.sol#382-384) is never used and

↳ should be removed

`SafeMath.mod(uint256,uint256)` (DFYtoken.sol#357-359) is never used and

↳ should be removed

`SafeMath.mod(uint256,uint256,string)` (DFYtoken.sol#373-380) is never

↳ used and should be removed

`SafeMath.sqrt(uint256)` (DFYtoken.sol#387-398) is never used and should

↳ be removed

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation>

↳ #dead-code

Low level call in `Address.sendValue(address,uint256)` (DFYtoken.sol

↳ #452-464):

- (success) = recipient.call{value: amount}() (DFYtoken.sol#459)

Low level call in `Address._functionCallWithValue(address,bytes,uint256,`

↳ `string)` (DFYtoken.sol#549-577):

```
- (success, returndata) = target.call{value: weiValue}(data) (  
  ↪ DFYtoken.sol#558-560)
```

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation>
↪ #low-level-calls

Variable ERC20._allowances (DFYtoken.sol#611) is not in mixedCase

Parameter DfyToken.mint(address,uint256)._to (DFYtoken.sol#986) is not
↪ in mixedCase

Parameter DfyToken.mint(address,uint256)._amount (DFYtoken.sol#986) is
↪ not in mixedCase

Parameter DfyToken.setRouter(address)._router (DFYtoken.sol#1019) is not
↪ in mixedCase

Parameter DfyToken.setDev(address)._dev (DFYtoken.sol#1058) is not in
↪ mixedCase

Parameter DfyToken.setBurnVault(address)._burnVault (DFYtoken.sol#1066)
↪ is not in mixedCase

Parameter DfyToken.setIlpVault(address)._ilpVault (DFYtoken.sol#1073) is
↪ not in mixedCase

Variable DfyToken._burnFee (DFYtoken.sol#933) is not in mixedCase

Variable DfyToken._ilpFee (DFYtoken.sol#934) is not in mixedCase

Variable DfyToken._devFee (DFYtoken.sol#935) is not in mixedCase

Variable DfyToken._maxTxAmount (DFYtoken.sol#937) is not in mixedCase

Variable DfyToken._maxSupply (DFYtoken.sol#938) is not in mixedCase

Variable DfyToken.BURN_VAULT (DFYtoken.sol#940) is not in mixedCase

Variable DfyToken.ILP_VAULT (DFYtoken.sol#941) is not in mixedCase

Variable DfyToken._delegates (DFYtoken.sol#1170) is not in mixedCase

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation>
↪ #conformance-to-solidity-naming-conventions

Redundant expression "this (DFYtoken.sol#26)" inContext (DFYtoken.sol
↪ #16-29)

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation>
↪ #redundant-statements

DfyToken._maxSupply (DFYtoken.sol#938) should be constant

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation>

↪ #state-variables-that-could-be-declared-constant

renounceOwnership() should be declared external:

- Ownable.renounceOwnership() (DFYtoken.sol#83-86)

transferOwnership(address) should be declared external:

- Ownable.transferOwnership(address) (DFYtoken.sol#92-94)

decimals() should be declared external:

- ERC20.decimals() (DFYtoken.sol#651-653)

symbol() should be declared external:

- ERC20.symbol() (DFYtoken.sol#658-660)

transfer(address,uint256) should be declared external:

- ERC20.transfer(address,uint256) (DFYtoken.sol#684-691)

allowance(address,address) should be declared external:

- ERC20.allowance(address,address) (DFYtoken.sol#696-703)

approve(address,uint256) should be declared external:

- ERC20.approve(address,uint256) (DFYtoken.sol#712-719)

transferFrom(address,address,uint256) should be declared external:

- ERC20.transferFrom(address,address,uint256) (DFYtoken.sol
↪ #733-748)

increaseAllowance(address,uint256) should be declared external:

- ERC20.increaseAllowance(address,uint256) (DFYtoken.sol#762-772)

decreaseAllowance(address,uint256) should be declared external:

- ERC20.decreaseAllowance(address,uint256) (DFYtoken.sol#788-801)

mint(uint256) should be declared external:

- DfyToken.mint(uint256) (DFYtoken.sol#996-999)
- ERC20.mint(uint256) (DFYtoken.sol#811-814)

setMaster(address) should be declared external:

- DfyToken.setMaster(address) (DFYtoken.sol#1080-1084)

burnToVault(uint256) should be declared external:

- DfyToken.burnToVault(uint256) (DFYtoken.sol#1091-1093)

burn(uint256) should be declared external:

- DfyToken.burn(uint256) (DFYtoken.sol#1097-1100)

transferTaxFree(address,uint256) should be declared external:
- DfyToken.transferTaxFree(address,uint256) (DFYtoken.sol
↪ #1102-1110)

transferFromTaxFree(address,address,uint256) should be declared external
↪ :
- DfyToken.transferFromTaxFree(address,address,uint256) (DFYtoken
↪ .sol#1112-1128)

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation>
↪ #public-function-that-could-be-declared-external

DFYtoken.sol analyzed (7 contracts with 78 detectors), 64 result(s)
↪ found

```
// [+]BurnVault
ERC20.constructor(string,string).name (BurnVault.sol#628) shadows:
- ERC20.name() (BurnVault.sol#644-646) (function)
- IERC20.name() (BurnVault.sol#129) (function)
ERC20.constructor(string,string).symbol (BurnVault.sol#628) shadows:
- ERC20.symbol() (BurnVault.sol#658-660) (function)
- IERC20.symbol() (BurnVault.sol#124) (function)
ERC20.allowance(address,address).owner (BurnVault.sol#696) shadows:
- Ownable.owner() (BurnVault.sol#64-66) (function)
ERC20._approve(address,address,uint256).owner (BurnVault.sol#899)
↪ shadows:
- Ownable.owner() (BurnVault.sol#64-66) (function)
```

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation>
↪ #local-variable-shadowing

BurnVault.setDefyMaster(address).master (BurnVault.sol#952) lacks a zero
↪ -check on :
- defyMaster = master (BurnVault.sol#954)

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation>
↪ #missing-zero-address-validation

Reentrancy in BurnVault.burn() (BurnVault.sol#963-967):

External calls:

- defy.burn(amount) (BurnVault.sol#965)

Event emitted after the call(s):

- Burn(amount) (BurnVault.sol#966)

Reentrancy in BurnVault.burnPortion(uint256) (BurnVault.sol#969-972):

External calls:

- defy.burn(amount) (BurnVault.sol#970)

Event emitted after the call(s):

- Burn(amount) (BurnVault.sol#971)

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation>

↪ #reentrancy-vulnerabilities-3

Address.isContract(address) (BurnVault.sol#423-434) uses assembly

- INLINE ASM (BurnVault.sol#430-432)

Address._functionCallWithValue(address,bytes,uint256,string) (BurnVault.

↪ sol#549-577) uses assembly

- INLINE ASM (BurnVault.sol#569-572)

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation>

↪ #assembly-usage

Address._functionCallWithValue(address,bytes,uint256,string) (BurnVault.

↪ sol#549-577) is never used and should be removed

Address.functionCall(address,bytes) (BurnVault.sol#484-489) is never

↪ used and should be removed

Address.functionCall(address,bytes,string) (BurnVault.sol#497-503) is

↪ never used and should be removed

Address.functionCallWithValue(address,bytes,uint256) (BurnVault.sol

↪ #516-528) is never used and should be removed

Address.functionCallWithValue(address,bytes,uint256,string) (BurnVault.

↪ sol#536-547) is never used and should be removed

Address.isContract(address) (BurnVault.sol#423-434) is never used and

↪ should be removed

`Address.sendValue(address,uint256)` (BurnVault.sol#452-464) is never used
↔ and should be removed

`Context._msgData()` (BurnVault.sol#25-28) is never used and should be
↔ removed

`ERC20._burn(address,uint256)` (BurnVault.sol#874-883) is never used and
↔ should be removed

`ERC20._burnFrom(address,uint256)` (BurnVault.sol#916-926) is never used
↔ and should be removed

`SafeMath.div(uint256,uint256)` (BurnVault.sol#317-319) is never used and
↔ should be removed

`SafeMath.div(uint256,uint256,string)` (BurnVault.sol#333-343) is never
↔ used and should be removed

`SafeMath.min(uint256,uint256)` (BurnVault.sol#382-384) is never used and
↔ should be removed

`SafeMath.mod(uint256,uint256)` (BurnVault.sol#357-359) is never used and
↔ should be removed

`SafeMath.mod(uint256,uint256,string)` (BurnVault.sol#373-380) is never
↔ used and should be removed

`SafeMath.mul(uint256,uint256)` (BurnVault.sol#291-303) is never used and
↔ should be removed

`SafeMath.sqrt(uint256)` (BurnVault.sol#387-398) is never used and should
↔ be removed

`SafeMath.sub(uint256,uint256)` (BurnVault.sol#256-258) is never used and
↔ should be removed

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation>
↔ #dead-code

Low level call in `Address.sendValue(address,uint256)` (BurnVault.sol
↔ #452-464):

- `(success) = recipient.call{value: amount}()` (BurnVault.sol#459)

Low level call in `Address._functionCallWithValue(address,bytes,uint256,
↔ string)` (BurnVault.sol#549-577):

- `(success, returndata) = target.call{value: weiValue}(data)` (
↔ BurnVault.sol#558-560)

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation>
↪ #low-level-calls

Parameter `BurnVault.setDefy(address)._defy` (`BurnVault.sol#958`) is not in
↪ `mixedCase`

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation>
↪ #conformance-to-solidity-naming-conventions

Redundant expression `"this (BurnVault.sol#26)" inContext` (`BurnVault.sol`
↪ #16-29)

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation>
↪ #redundant-statements

`renounceOwnership()` should be declared `external`:

- `Ownable.renounceOwnership()` (`BurnVault.sol#83-86`)

`transferOwnership(address)` should be declared `external`:

- `Ownable.transferOwnership(address)` (`BurnVault.sol#92-94`)

`name()` should be declared `external`:

- `ERC20.name()` (`BurnVault.sol#644-646`)

`decimals()` should be declared `external`:

- `ERC20.decimals()` (`BurnVault.sol#651-653`)

`symbol()` should be declared `external`:

- `ERC20.symbol()` (`BurnVault.sol#658-660`)

`totalSupply()` should be declared `external`:

- `ERC20.totalSupply()` (`BurnVault.sol#665-667`)

`balanceOf(address)` should be declared `external`:

- `ERC20.balanceOf(address)` (`BurnVault.sol#672-674`)

`transfer(address,uint256)` should be declared `external`:

- `ERC20.transfer(address,uint256)` (`BurnVault.sol#684-691`)

`allowance(address,address)` should be declared `external`:

- `ERC20.allowance(address,address)` (`BurnVault.sol#696-703`)

`approve(address,uint256)` should be declared `external`:

- `ERC20.approve(address,uint256)` (`BurnVault.sol#712-719`)

`transferFrom(address,address,uint256)` should be declared `external`:

- ERC20.transferFrom(address,address,uint256) (BurnVault.sol
↪ #733-748)

increaseAllowance(address,uint256) should be declared external:

- ERC20.increaseAllowance(address,uint256) (BurnVault.sol
↪ #762-772)

decreaseAllowance(address,uint256) should be declared external:

- ERC20.decreaseAllowance(address,uint256) (BurnVault.sol
↪ #788-801)

mint(uint256) should be declared external:

- ERC20.mint(uint256) (BurnVault.sol#811-814)

burn() should be declared external:

- BurnVault.burn() (BurnVault.sol#963-967)

burnPortion(uint256) should be declared external:

- BurnVault.burnPortion(uint256) (BurnVault.sol#969-972)

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation>

↪ #public-function-that-could-be-declared-external

BurnVault.sol analyzed (8 contracts with 78 detectors), 47 result(s)

↪ found

DefySwapPair._update(uint256,uint256,uint112,uint112) (factory.sol

↪ #574-599) uses a weak PRNG: "blockTimestamp = uint32(block.

↪ timestamp % 2 ** 32) (factory.sol#584)"

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation>

↪ #weak-PRNG

DefySwapPair._safeTransfer(address,address,uint256) (factory.sol

↪ #513-526) uses a dangerous strict equality:

- require(bool,string)(success && (data.length == 0 || abi.decode
↪ (data,(bool))),DefySwap: TRANSFER_FAILED) (factory.sol
↪ #522-525)

DefySwapPair._safeTransferTaxFree(address,address,uint256) (factory.sol

↪ #528-542) uses a dangerous strict equality:

- require(bool,string)(success && (data.length == 0 || abi.decode
↪ (data,(bool))),TransferHelper: TRANSFER_FAILED) (factory.

↪ sol#538-541)

DefySwapPair.mint(address) (factory.sol#626-651) uses a dangerous strict

↪ equality:

- _totalSupply == 0 (factory.sol#636)

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation>

↪ #dangerous-strict-equalities

Reentrancy in DefySwapPair.burn(address) (factory.sol#654-694):

External calls:

```
- _safeTransferTaxFree(_token0,to,amount0) (factory.sol#678)
  - (success,data) = token.call(abi.encodeWithSelector(0
    ↪ xdfc1a11,to,value)) (factory.sol#535-537)
- _safeTransfer(_token0,to,amount0) (factory.sol#680)
  - (success,data) = token.call(abi.encodeWithSelector(
    ↪ SELECTOR,to,value)) (factory.sol#519-521)
- _safeTransferTaxFree(_token1,to,amount1) (factory.sol#683)
  - (success,data) = token.call(abi.encodeWithSelector(0
    ↪ xdfc1a11,to,value)) (factory.sol#535-537)
- _safeTransfer(_token1,to,amount1) (factory.sol#685)
  - (success,data) = token.call(abi.encodeWithSelector(
    ↪ SELECTOR,to,value)) (factory.sol#519-521)
```

State variables written after the call(s):

```
- _update(balance0,balance1,_reserve0,_reserve1) (factory.sol
  ↪ #691)
  - blockTimestampLast = blockTimestamp (factory.sol#597)
- kLast = uint256(reserve0).mul(reserve1) (factory.sol#692)
- _update(balance0,balance1,_reserve0,_reserve1) (factory.sol
  ↪ #691)
  - reserve0 = uint112(balance0) (factory.sol#595)
- _update(balance0,balance1,_reserve0,_reserve1) (factory.sol
  ↪ #691)
  - reserve1 = uint112(balance1) (factory.sol#596)
```

Reentrancy in DefySwapFactory.createPair(address,address) (factory.sol

↪ #816-838):

External calls:

- IDefySwapPair(pair).initialize(token0,token1) (factory.sol#833)

State variables written after the call(s):

- getPair[token0][token1] = pair (factory.sol#834)

- getPair[token1][token0] = pair (factory.sol#835)

Reentrancy in DefySwapPair.swap(uint256,uint256,address,bytes) (factory.
↪ sol#697-760):

External calls:

- _safeTransfer(_token0,to,amount0Out) (factory.sol#721)

- (success,data) = token.call(abi.encodeWithSelector(
↪ SELECTOR,to,value)) (factory.sol#519-521)

- _safeTransfer(_token1,to,amount1Out) (factory.sol#722)

- (success,data) = token.call(abi.encodeWithSelector(
↪ SELECTOR,to,value)) (factory.sol#519-521)

- IDefySwapCallee(to).defyswapCall(msg.sender,amount0Out,
↪ amount1Out,data) (factory.sol#724-729)

State variables written after the call(s):

- _update(balance0,balance1,_reserve0,_reserve1) (factory.sol
↪ #758)

- blockTimestampLast = blockTimestamp (factory.sol#597)

- _update(balance0,balance1,_reserve0,_reserve1) (factory.sol
↪ #758)

- reserve0 = uint112(balance0) (factory.sol#595)

- _update(balance0,balance1,_reserve0,_reserve1) (factory.sol
↪ #758)

- reserve1 = uint112(balance1) (factory.sol#596)

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation>
↪ #reentrancy-vulnerabilities-1

DefySwapPair.initialize(address,address)._token0 (factory.sol#566) lacks
↪ a zero-check on :

- token0 = _token0 (factory.sol#569)

DefySwapPair.initialize(address,address)._token1 (factory.sol#566) lacks
↪ a zero-check on :

```

        - token1 = _token1 (factory.sol#570)
DefySwapFactory.constructor(address)._feeToSetter (factory.sol#807)
    ↪ lacks a zero-check on :
        - feeToSetter = _feeToSetter (factory.sol#809)
DefySwapFactory.setFeeTo(address)._feeTo (factory.sol#840) lacks a zero-
    ↪ check on :
        - feeTo = _feeTo (factory.sol#843)
DefySwapFactory.setFeeToSetter(address)._feeToSetter (factory.sol#846)
    ↪ lacks a zero-check on :
        - feeToSetter = _feeToSetter (factory.sol#849)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #missing-zero-address-validation

```

Reentrancy in DefySwapPair.burn(address) (factory.sol#654-694):

External calls:

```

- _safeTransferTaxFree(_token0,to,amount0) (factory.sol#678)
    - (success,data) = token.call(abi.encodeWithSelector(0
        ↪ xdfc1a11,to,value)) (factory.sol#535-537)
- _safeTransfer(_token0,to,amount0) (factory.sol#680)
    - (success,data) = token.call(abi.encodeWithSelector(
        ↪ SELECTOR,to,value)) (factory.sol#519-521)
- _safeTransferTaxFree(_token1,to,amount1) (factory.sol#683)
    - (success,data) = token.call(abi.encodeWithSelector(0
        ↪ xdfc1a11,to,value)) (factory.sol#535-537)
- _safeTransfer(_token1,to,amount1) (factory.sol#685)
    - (success,data) = token.call(abi.encodeWithSelector(
        ↪ SELECTOR,to,value)) (factory.sol#519-521)

```

State variables written after the call(s):

```

- _update(balance0,balance1,_reserve0,_reserve1) (factory.sol
    ↪ #691)
    - price0CumulativeLast += uint256(UQ112x112.encode(
        ↪ _reserve1).uqdiv(_reserve0)) * timeElapsed (factory
        ↪ .sol#588-590)

```



```

- _update(balance0,balance1,_reserve0,_reserve1) (factory.sol
  ↪ #691)
    - price1CumulativeLast += uint256(UQ112x112.encode(
      ↪ _reserve0).uqdiv(_reserve1)) * timeElapsed (factory
      ↪ .sol#591-593)
Reentrancy in DefySwapFactory.createPair(address,address) (factory.sol
  ↪ #816-838):
  External calls:
  - IDefySwapPair(pair).initialize(token0,token1) (factory.sol#833)
  State variables written after the call(s):
  - allPairs.push(pair) (factory.sol#836)
Reentrancy in DefySwapPair.swap(uint256,uint256,address,bytes) (factory.
  ↪ sol#697-760):
  External calls:
  - _safeTransfer(_token0,to,amount0Out) (factory.sol#721)
    - (success,data) = token.call(abi.encodeWithSelector(
      ↪ SELECTOR,to,value)) (factory.sol#519-521)
  - _safeTransfer(_token1,to,amount1Out) (factory.sol#722)
    - (success,data) = token.call(abi.encodeWithSelector(
      ↪ SELECTOR,to,value)) (factory.sol#519-521)
  - IDefySwapCallee(to).defyswapCall(msg.sender,amount0Out,
    ↪ amount1Out,data) (factory.sol#724-729)
  State variables written after the call(s):
  - _update(balance0,balance1,_reserve0,_reserve1) (factory.sol
    ↪ #758)
    - price0CumulativeLast += uint256(UQ112x112.encode(
      ↪ _reserve1).uqdiv(_reserve0)) * timeElapsed (factory
      ↪ .sol#588-590)
  - _update(balance0,balance1,_reserve0,_reserve1) (factory.sol
    ↪ #758)
    - price1CumulativeLast += uint256(UQ112x112.encode(
      ↪ _reserve0).uqdiv(_reserve1)) * timeElapsed (factory
      ↪ .sol#591-593)

```

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation>
↪ #reentrancy-vulnerabilities-2

Reentrancy in DefySwapPair.burn(address) (factory.sol#654-694):

External calls:

- _safeTransferTaxFree(_token0,to,amount0) (factory.sol#678)
 - (success,data) = token.call(abi.encodeWithSelector(0 ↪ xdfc1a11,to,value)) (factory.sol#535-537)
- _safeTransfer(_token0,to,amount0) (factory.sol#680)
 - (success,data) = token.call(abi.encodeWithSelector(↪ SELECTOR,to,value)) (factory.sol#519-521)
- _safeTransferTaxFree(_token1,to,amount1) (factory.sol#683)
 - (success,data) = token.call(abi.encodeWithSelector(0 ↪ xdfc1a11,to,value)) (factory.sol#535-537)
- _safeTransfer(_token1,to,amount1) (factory.sol#685)
 - (success,data) = token.call(abi.encodeWithSelector(↪ SELECTOR,to,value)) (factory.sol#519-521)

Event emitted after the call(s):

- Burn(msg.sender,amount0,amount1,to) (factory.sol#693)
- Sync(reserve0,reserve1) (factory.sol#598)
 - _update(balance0,balance1,_reserve0,_reserve1) (factory. ↪ sol#691)

Reentrancy in DefySwapFactory.createPair(address,address) (factory.sol ↪ #816-838):

External calls:

- IDefySwapPair(pair).initialize(token0,token1) (factory.sol#833)

Event emitted after the call(s):

- PairCreated(token0,token1,pair,allPairs.length) (factory.sol ↪ #837)

Reentrancy in DefySwapPair.swap(uint256,uint256,address,bytes) (factory. ↪ sol#697-760):

External calls:

- _safeTransfer(_token0,to,amount0Out) (factory.sol#721)

```

- (success,data) = token.call(abi.encodeWithSelector(
  ↪ SELECTOR,to,value)) (factory.sol#519-521)
- _safeTransfer(_token1,to,amount1Out) (factory.sol#722)
- (success,data) = token.call(abi.encodeWithSelector(
  ↪ SELECTOR,to,value)) (factory.sol#519-521)
- IDefySwapCallee(to).defyswapCall(msg.sender,amount0Out,
  ↪ amount1Out,data) (factory.sol#724-729)
Event emitted after the call(s):
- Swap(msg.sender,amount0In,amount1In,amount0Out,amount1Out,to) (
  ↪ factory.sol#759)
- Sync(reserve0,reserve1) (factory.sol#598)
  - _update(balance0,balance1,_reserve0,_reserve1) (factory.
    ↪ sol#758)

```

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation>
 ↪ #reentrancy-vulnerabilities-3

```

DefySwapERC20.permit(address,address,uint256,uint256,uint8,bytes32,
  ↪ bytes32) (factory.sol#315-347) uses timestamp for comparisons

```

Dangerous comparisons:

```

- require(bool,string)(deadline >= block.timestamp,DefySwap:
  ↪ EXPIRED) (factory.sol#324)

```

```

DefySwapPair._update(uint256,uint256,uint112,uint112) (factory.sol
  ↪ #574-599) uses timestamp for comparisons

```

Dangerous comparisons:

```

- timeElapsed > 0 && _reserve0 != 0 && _reserve1 != 0 (factory.
  ↪ sol#586)

```

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation>
 ↪ #block-timestamp

```

DefySwapERC20.constructor() (factory.sol#242-258) uses assembly
- INLINE ASM (factory.sol#244-246)

```

```

DefySwapFactory.createPair(address,address) (factory.sol#816-838) uses
  ↪ assembly

```

```

- INLINE ASM (factory.sol#829-831)

```

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation>
↪ #assembly-usage

Low level call in DefySwapPair._safeTransfer(address,address,uint256) (
↪ factory.sol#513-526):
- (success,data) = token.call(abi.encodeWithSelector(SELECTOR,to,
↪ value)) (factory.sol#519-521)

Low level call in DefySwapPair._safeTransferTaxFree(address,address,
↪ uint256) (factory.sol#528-542):
- (success,data) = token.call(abi.encodeWithSelector(0xdffc1a11,
↪ to,value)) (factory.sol#535-537)

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation>
↪ #low-level-calls

Function IDefySwapPair.DOMAIN_SEPARATOR() (factory.sol#66) is not in
↪ mixedCase

Function IDefySwapPair.PERMIT_TYPEHASH() (factory.sol#68) is not in
↪ mixedCase

Function IDefySwapPair.MINIMUM_LIQUIDITY() (factory.sol#99) is not in
↪ mixedCase

Parameter IDefySwapPair.setLiqTax(bool,bool).token0_Tax (factory.sol
↪ #116) is not in mixedCase

Parameter IDefySwapPair.setLiqTax(bool,bool).token1_tax (factory.sol
↪ #116) is not in mixedCase

Parameter IDefySwapPair.setRLiqTax(bool,bool).token0_Tax (factory.sol
↪ #120) is not in mixedCase

Parameter IDefySwapPair.setRLiqTax(bool,bool).token1_tax (factory.sol
↪ #120) is not in mixedCase

Function IDefySwapERC20.DOMAIN_SEPARATOR() (factory.sol#187) is not in
↪ mixedCase

Function IDefySwapERC20.PERMIT_TYPEHASH() (factory.sol#189) is not in
↪ mixedCase

Variable DefySwapERC20.DOMAIN_SEPARATOR (factory.sol#229) is not in
↪ mixedCase

Parameter DefySwapPair.setLiqTax(**bool, bool**).token0_Tax (factory.sol#477)
↳ is not in mixedCase

Parameter DefySwapPair.setLiqTax(**bool, bool**).token1_tax (factory.sol#477)
↳ is not in mixedCase

Parameter DefySwapPair.setRLiqTax(**bool, bool**).token0_Tax (factory.sol
↳ #490) is not in mixedCase

Parameter DefySwapPair.setRLiqTax(**bool, bool**).token1_tax (factory.sol
↳ #490) is not in mixedCase

Parameter DefySwapPair.initialize(**address, address**)._token0 (factory.sol
↳ #566) is not in mixedCase

Parameter DefySwapPair.initialize(**address, address**)._token1 (factory.sol
↳ #566) is not in mixedCase

Parameter DefySwapFactory.setFeeTo(**address**)._feeTo (factory.sol#840) is
↳ not in mixedCase

Parameter DefySwapFactory.setFeeToSetter(**address**)._feeToSetter (factory.
↳ sol#846) is not in mixedCase

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation>
↳ #conformance-to-solidity-naming-conventions

Variable DefySwapPair.swap(**uint256, uint256, address, bytes**).
↳ balance0Adjusted (factory.sol#745-747) is too similar to
↳ DefySwapPair.swap(**uint256, uint256, address, bytes**).balance1Adjusted
↳ (factory.sol#748-750)

Variable DefySwapPair.price0CumulativeLast (factory.sol#447) is too
↳ similar to DefySwapPair.price1CumulativeLast (factory.sol#448)

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation>
↳ #variable-names-are-too-similar

DefySwapFactory.createPair(**address, address**) (factory.sol#816-838) uses
↳ literals with too many digits:

- bytecode = type(**address**)(DefySwapPair).creationCode (factory.
↳ sol#827)

DefySwapFactory.slitherConstructorConstantVariables() (factory.sol
↳ #790-851) uses literals with too many digits:

```
- INIT_CODE_PAIR_HASH = keccak256(bytes)(abi.encodePacked(type(
  ↪ address)(DefySwapPair).creationCode)) (factory.sol
  ↪ #791-792)
```

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation>
↪ #too-many-digits

isAddTaxFree(address) should be declared external:

```
- DefySwapPair.isAddTaxFree(address) (factory.sol#503-506)
```

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation>
↪ #public-function-that-could-be-declared-external

factory.sol analyzed (11 contracts with 78 detectors), 47 result(s)

↪ found

```
ImpermanentLossProtection.defyTransfer(address,uint256) (ILP.sol
  ↪ #887-892) ignores return value by IERC20(defy).transfer(_to,xfAmt
  ↪ ) (ILP.sol#891)
```

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation>
↪ #unchecked-transfer

ImpermanentLossProtection.dev(address) (ILP.sol#895-899) should emit an
↪ event for:

```
- devAddr = _devAddr (ILP.sol#898)
```

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation>
↪ #missing-events-access-control

```
ImpermanentLossProtection.constructor(address,address)._defy (ILP.sol
  ↪ #858) lacks a zero-check on :
```

```
- defy = _defy (ILP.sol#860)
```

```
ImpermanentLossProtection.constructor(address,address)._defyMaster (ILP.
  ↪ sol#858) lacks a zero-check on :
```

```
- defyMaster = _defyMaster (ILP.sol#861)
```

```
ImpermanentLossProtection.setAddresses(address,address)._defy (ILP.sol
  ↪ #877) lacks a zero-check on :
```

```
- defy = _defy (ILP.sol#879)
```

`ImpermanentLossProtection.setAddresses(address,address)._defyMaster` (ILP.sol#877) lacks a zero-check on :

- `defyMaster = _defyMaster` (ILP.sol#880)

`ImpermanentLossProtection.dev(address)._devAddr` (ILP.sol#895) lacks a

↪ zero-check on :

- `devAddr = _devAddr` (ILP.sol#898)

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation>

↪ #missing-zero-address-validation

`Address.isContract(address)` (ILP.sol#580-591) uses assembly

- `INLINE ASM` (ILP.sol#587-589)

`Address._functionCallWithValue(address,bytes,uint256,string)` (ILP.sol

↪ #706-734) uses assembly

- `INLINE ASM` (ILP.sol#726-729)

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation>

↪ #assembly-usage

`Address._functionCallWithValue(address,bytes,uint256,string)` (ILP.sol

↪ #706-734) is never used and should be removed

`Address.functionCall(address,bytes)` (ILP.sol#641-646) is never used and

↪ should be removed

`Address.functionCall(address,bytes,string)` (ILP.sol#654-660) is never

↪ used and should be removed

`Address.functionCallWithValue(address,bytes,uint256)` (ILP.sol#673-685)

↪ is never used and should be removed

`Address.functionCallWithValue(address,bytes,uint256,string)` (ILP.sol

↪ #693-704) is never used and should be removed

`Address.isContract(address)` (ILP.sol#580-591) is never used and should

↪ be removed

`Address.sendValue(address,uint256)` (ILP.sol#609-621) is never used and

↪ should be removed

`Context._msgData()` (ILP.sol#25-28) is never used and should be removed

`SafeERC20._callOptionalReturn(IERC20,bytes)` (ILP.sol#351-368) is never

↪ used and should be removed

SafeERC20.safeApprove(IERC20,address,uint256) (ILP.sol#289-306) is never
↔ used and should be removed

SafeERC20.safeDecreaseAllowance(IERC20,address,uint256) (ILP.sol
↔ #326-343) is never used and should be removed

SafeERC20.safeIncreaseAllowance(IERC20,address,uint256) (ILP.sol
↔ #308-324) is never used and should be removed

SafeERC20.safeTransfer(IERC20,address,uint256) (ILP.sol#259-268) is
↔ never used and should be removed

SafeERC20.safeTransferFrom(IERC20,address,address,uint256) (ILP.sol
↔ #270-280) is never used and should be removed

SafeMath.add(uint256,uint256) (ILP.sol#396-401) is never used and should
↔ be removed

SafeMath.min(uint256,uint256) (ILP.sol#539-541) is never used and should
↔ be removed

SafeMath.mod(uint256,uint256) (ILP.sol#514-516) is never used and should
↔ be removed

SafeMath.mod(uint256,uint256,string) (ILP.sol#530-537) is never used and
↔ should be removed

SafeMath.sqrt(uint256) (ILP.sol#544-555) is never used and should be
↔ removed

SafeMath.sub(uint256,uint256) (ILP.sol#413-415) is never used and should
↔ be removed

SafeMath.sub(uint256,uint256,string) (ILP.sol#427-436) is never used and
↔ should be removed

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation>
↔ #dead-code

Low level call in Address.sendValue(address,uint256) (ILP.sol#609-621):
- (success) = recipient.call{value: amount}() (ILP.sol#616)

Low level call in Address._functionCallWithValue(address,bytes,uint256,
↔ string) (ILP.sol#706-734):
- (success,returndata) = target.call{value: weiValue}(data) (ILP.
↔ sol#715-717)

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation>

↪ #low-level-calls

Function IDefySwapPair.DOMAIN_SEPARATOR() (ILP.sol#64) is not in

↪ mixedCase

Function IDefySwapPair.PERMIT_TYPEHASH() (ILP.sol#66) is not in

↪ mixedCase

Function IDefySwapPair.MINIMUM_LIQUIDITY() (ILP.sol#97) is not in

↪ mixedCase

Parameter ImpermanentLossProtection.setAddresses(address,address)._defy

↪ (ILP.sol#877) is not in mixedCase

Parameter ImpermanentLossProtection.setAddresses(address,address).

↪ _defyMaster (ILP.sol#877) is not in mixedCase

Parameter ImpermanentLossProtection.defyTransfer(address,uint256)._to (

↪ ILP.sol#887) is not in mixedCase

Parameter ImpermanentLossProtection.defyTransfer(address,uint256).

↪ _amount (ILP.sol#887) is not in mixedCase

Parameter ImpermanentLossProtection.dev(address)._devAddr (ILP.sol#895)

↪ is not in mixedCase

Parameter ImpermanentLossProtection.add(address,IERC20,IERC20,bool).

↪ _lpToken (ILP.sol#902) is not in mixedCase

Parameter ImpermanentLossProtection.add(address,IERC20,IERC20,bool).

↪ _token0 (ILP.sol#903) is not in mixedCase

Parameter ImpermanentLossProtection.add(address,IERC20,IERC20,bool).

↪ _token1 (ILP.sol#904) is not in mixedCase

Parameter ImpermanentLossProtection.add(address,IERC20,IERC20,bool).

↪ _offerILP (ILP.sol#905) is not in mixedCase

Parameter ImpermanentLossProtection.set(uint256,IERC20,IERC20,bool)._pid

↪ (ILP.sol#921) is not in mixedCase

Parameter ImpermanentLossProtection.set(uint256,IERC20,IERC20,bool).

↪ _token0 (ILP.sol#922) is not in mixedCase

Parameter ImpermanentLossProtection.set(uint256,IERC20,IERC20,bool).

↪ _token1 (ILP.sol#923) is not in mixedCase

Parameter `ImpermanentLossProtection.set(uint256,IERC20,IERC20,bool)`.

↪ `_offerILP` (ILP.sol#924) is not in mixedCase

Parameter `ImpermanentLossProtection.getDepositValue(uint256,uint256)`.

↪ `_pid` (ILP.sol#936) is not in mixedCase

Parameter `ImpermanentLossProtection.getDefyPrice(uint256)`. `_pid` (ILP.sol

↪ #970) is not in mixedCase

Parameter `ImpermanentLossProtection.getReserves(uint256)`. `_pid` (ILP.sol

↪ #1033) is not in mixedCase

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation>

↪ #conformance-to-solidity-naming-conventions

Redundant expression `"this (ILP.sol#26)" inContext` (ILP.sol#16-29)

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation>

↪ #redundant-statements

`renounceOwnership()` should be declared `external`:

- `Ownable.renounceOwnership()` (ILP.sol#789-792)

`transferOwnership(address)` should be declared `external`:

- `Ownable.transferOwnership(address)` (ILP.sol#798-800)

`setAddresses(address,address)` should be declared `external`:

- `ImpermanentLossProtection.setAddresses(address,address)` (ILP.
↪ sol#877-884)

`dev(address)` should be declared `external`:

- `ImpermanentLossProtection.dev(address)` (ILP.sol#895-899)

`add(address,IERC20,IERC20,bool)` should be declared `external`:

- `ImpermanentLossProtection.add(address,IERC20,IERC20,bool)` (ILP.
↪ sol#901-918)

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation>

↪ #public-function-that-could-be-declared-external

ILP.sol analyzed (8 `contracts` with 78 detectors), 57 result(s) found

Compilation warnings/errors on `router.sol`:

Warning: `Contract` code size exceeds 24576 `bytes` (a limit introduced in

↪ Spurious Dragon). `This contract` may not be deployable on mainnet.

↪ Consider enabling the optimizer (with a low "runs" value!),
↪ turning off revert strings, or using libraries.

```
--> router.sol:643:1:  
|  
643 | contract DefySwapRouter is IDefySwapRouter02 {  
| ^ (Relevant source part starts here and spans across multiple  
↪ lines).
```

DefySwapRouter.addLiquidityETH(address,uint256,uint256,uint256,address,
↪ uint256) (router.sol#767-819) sends eth to arbitrary user

Dangerous calls:

- IWETH(WETH).deposit{value: amountETH}() (router.sol#813)

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation>

↪ #functions-that-send-ether-to-arbitrary-destinations

DefySwapRouter.removeLiquidity(address,address,uint256,uint256,uint256,
↪ address,uint256) (router.sol#822-847) ignores return value by
↪ IDefySwapPair(pair).transferFrom(msg.sender,pair,liquidity) (
↪ router.sol#839)

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation>

↪ #unchecked-transfer

DefySwapLibrary.getAmountsOut(address,uint256,address[]).i (router.sol
↪ #571) is a local variable never initialized

DefySwapRouter._swap(uint256[],address[],address).i (router.sol#1048) is
↪ a local variable never initialized

DefySwapRouter._swapSupportingFeeOnTransferTokens(address[],address).i (
↪ router.sol#1256) is a local variable never initialized

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation>

↪ #uninitialized-local-variables

```
DefySwapRouter._addLiquidity(address,address,uint256,uint256,uint256,  
    ↪ uint256) (router.sol#665-711) ignores return value by  
    ↪ IDefySwapFactory(factory).createPair(tokenA,tokenB) (router.sol  
    ↪ #676)
```

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation>
 ↪ #unused-return

```
DefySwapRouter.constructor(address,address)._factory (router.sol#654)  
    ↪ lacks a zero-check on :  
        - factory = _factory (router.sol#656)
```

```
DefySwapRouter.constructor(address,address)._WETH (router.sol#654) lacks  
    ↪ a zero-check on :  
        - WETH = _WETH (router.sol#657)
```

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation>
 ↪ #missing-zero-address-validation

```
TransferHelper.safeApprove(address,address,uint256) (router.sol#37-50)  
    ↪ is never used and should be removed
```

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation>
 ↪ #dead-code

```
Pragma version=0.6.6 (router.sol#1) allows old versions  
solc-0.6.6 is not recommended for deployment
```

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation>
 ↪ #incorrect-versions-of-solidity

```
Low level call in TransferHelper.safeApprove(address,address,uint256) (  
    ↪ router.sol#37-50):  
    - (success,data) = token.call(abi.encodeWithSelector(0x095ea7b3,  
        ↪ to,value)) (router.sol#43-45)
```

```
Low level call in TransferHelper.safeTransfer(address,address,uint256) (  
    ↪ router.sol#52-65):  
    - (success,data) = token.call(abi.encodeWithSelector(0xa9059cbb,  
        ↪ to,value)) (router.sol#58-60)
```

Low level call in TransferHelper.safeTransferTaxFree(address,address,
↪ uint256) (router.sol#67-80):

- (success,data) = token.call(abi.encodeWithSelector(0xdffc1a11,
↪ to,value)) (router.sol#73-75)

Low level call in TransferHelper.safeTransferFrom(address,address,
↪ address,uint256) (router.sol#82-96):

- (success,data) = token.call(abi.encodeWithSelector(0x23b872dd,
↪ from,to,value)) (router.sol#89-91)

Low level call in TransferHelper.safeTransferFromTaxFree(address,address
↪ ,address,uint256) (router.sol#98-112):

- (success,data) = token.call(abi.encodeWithSelector(0x57dd378e,
↪ from,to,value)) (router.sol#105-107)

Low level call in TransferHelper.safeTransferETH(address,uint256) (
↪ router.sol#114-117):

- (success) = to.call{value: value}(new bytes(0)) (router.sol
↪ #115)

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation>
↪ #low-level-calls

Function IDefySwapFactory.INIT_CODE_PAIR_HASH() (router.sol#32) is not
↪ in mixedCase

Function IDefySwapRouter01.WETH() (router.sol#123) is not in mixedCase

Function IDefySwapPair.DOMAIN_SEPARATOR() (router.sol#359) is not in
↪ mixedCase

Function IDefySwapPair.PERMIT_TYPEHASH() (router.sol#361) is not in
↪ mixedCase

Function IDefySwapPair.MINIMUM_LIQUIDITY() (router.sol#392) is not in
↪ mixedCase

Parameter IDefySwapPair.setLiqTax(bool,bool).token0_Tax (router.sol#409)
↪ is not in mixedCase

Parameter IDefySwapPair.setLiqTax(bool,bool).token1_tax (router.sol#409)
↪ is not in mixedCase

Parameter IDefySwapPair.setRLiqTax(bool,bool).token0_Tax (router.sol
↪ #413) is not in mixedCase

Parameter IDefySwapPair.setRLiqTax(`bool,bool`).token1_tax (router.sol
↳ #413) is not in mixedCase

Variable DefySwapRouter.WETH (router.sol#647) is not in mixedCase

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation>
↳ #conformance-to-solidity-naming-conventions

Variable IDefySwapRouter01.addLiquidity(`address,address,uint256,uint256,`
↳ `uint256,uint256,address,uint256`).amountADesired (router.sol#128)
↳ is too similar to IDefySwapRouter01.addLiquidity(`address,address,`
↳ `uint256,uint256,uint256,uint256,address,uint256`).amountBDesired (
↳ router.sol#129)

Variable DefySwapRouter._addLiquidity(`address,address,uint256,uint256,`
↳ `uint256,uint256`).amountADesired (router.sol#668) is too similar
↳ to DefySwapRouter._addLiquidity(`address,address,uint256,uint256,`
↳ `uint256,uint256`).amountBDesired (router.sol#669)

Variable DefySwapRouter._addLiquidity(`address,address,uint256,uint256,`
↳ `uint256,uint256`).amountADesired (router.sol#668) is too similar
↳ to DefySwapRouter.addLiquidity(`address,address,uint256,uint256,`
↳ `uint256,uint256,address,uint256`).amountBDesired (router.sol#717)

Variable IDefySwapRouter01.addLiquidity(`address,address,uint256,uint256,`
↳ `uint256,uint256,address,uint256`).amountADesired (router.sol#128)
↳ is too similar to DefySwapRouter._addLiquidity(`address,address,`
↳ `uint256,uint256,uint256,uint256`).amountBDesired (router.sol#669)

Variable IDefySwapRouter01.addLiquidity(`address,address,uint256,uint256,`
↳ `uint256,uint256,address,uint256`).amountADesired (router.sol#128)
↳ is too similar to DefySwapRouter.addLiquidity(`address,address,`
↳ `uint256,uint256,uint256,uint256,address,uint256`).amountBDesired (
↳ router.sol#717)

Variable DefySwapRouter.addLiquidity(`address,address,uint256,uint256,`
↳ `uint256,uint256,address,uint256`).amountADesired (router.sol#716)
↳ is too similar to DefySwapRouter.addLiquidity(`address,address,`
↳ `uint256,uint256,uint256,uint256,address,uint256`).amountBDesired (
↳ router.sol#717)

Variable DefySwapRouter.addLiquidity(address,address,uint256,uint256,
 ↪ uint256,uint256,address,uint256).amountADesired (router.sol#716)
 ↪ is too similar to DefySwapRouter._addLiquidity(address,address,
 ↪ uint256,uint256,uint256,uint256).amountBDesired (router.sol#669)

Variable DefySwapRouter._addLiquidity(address,address,uint256,uint256,
 ↪ uint256,uint256).amountADesired (router.sol#668) is too similar
 ↪ to IDefySwapRouter01.addLiquidity(address,address,uint256,uint256
 ↪ ,uint256,uint256,address,uint256).amountBDesired (router.sol#129)

Variable DefySwapRouter.addLiquidity(address,address,uint256,uint256,
 ↪ uint256,uint256,address,uint256).amountADesired (router.sol#716)
 ↪ is too similar to IDefySwapRouter01.addLiquidity(address,address,
 ↪ uint256,uint256,uint256,uint256,address,uint256).amountBDesired (
 ↪ router.sol#129)

Variable DefySwapRouter._addLiquidity(address,address,uint256,uint256,
 ↪ uint256,uint256).amountAOptimal (router.sol#698-702) is too
 ↪ similar to DefySwapRouter._addLiquidity(address,address,uint256,
 ↪ uint256,uint256,uint256).amountBOptimal (router.sol#686-690)

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation>
 ↪ #variable-names-are-too-similar

quote(uint256,uint256,uint256) should be declared external:

- DefySwapRouter.quote(uint256,uint256,uint256) (router.sol
 ↪ #1361-1367)

getAmountOut(uint256,uint256,uint256) should be declared external:

- DefySwapRouter.getAmountOut(uint256,uint256,uint256) (router.
 ↪ sol#1369-1375)

getAmountIn(uint256,uint256,uint256) should be declared external:

- DefySwapRouter.getAmountIn(uint256,uint256,uint256) (router.sol
 ↪ #1377-1383)

getAmountsOut(uint256,address[]) should be declared external:

- DefySwapRouter.getAmountsOut(uint256,address[]) (router.sol
 ↪ #1385-1393)

getAmountsIn(uint256,address[]) should be declared external:

- DefySwapRouter.getAmountsIn(uint256,address[]) (router.sol
↪ #1395-1403)

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation>
↪ #public-function-that-could-be-declared-external

router.sol analyzed (10 contracts with 78 detectors), 42 result(s) found

DefySTUB._writeCheckpoint(address,uint32,uint256,uint256) (STUB.sol
↪ #1151-1176) uses a dangerous strict equality:

- nCheckpoints > 0 && checkpoints[delegatee][nCheckpoints - 1].
↪ fromBlock == blockNumber (STUB.sol#1163-1164)

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation>
↪ #dangerous-strict-equalities

ERC20.constructor(string,string).name (STUB.sol#628) shadows:

- ERC20.name() (STUB.sol#644-646) (function)
- IERC20.name() (STUB.sol#129) (function)

ERC20.constructor(string,string).symbol (STUB.sol#628) shadows:

- ERC20.symbol() (STUB.sol#658-660) (function)
- IERC20.symbol() (STUB.sol#124) (function)

ERC20.allowance(address,address).owner (STUB.sol#696) shadows:

- Ownable.owner() (STUB.sol#64-66) (function)

ERC20._approve(address,address,uint256).owner (STUB.sol#899) shadows:

- Ownable.owner() (STUB.sol#64-66) (function)

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation>
↪ #local-variable-shadowing

DefySTUB.delegateBySig(address,uint256,uint256,uint8,bytes32,bytes32) (
↪ STUB.sol#1015-1052) uses timestamp for comparisons

Dangerous comparisons:

- require(bool,string)(now <= expiry,DEFY::delegateBySig:
↪ signature expired) (STUB.sol#1050)

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation>
↪ #block-timestamp

`Address.isContract(address)` (STUB.sol#423-434) uses assembly
- INLINE ASM (STUB.sol#430-432)

`Address._functionCallWithValue(address,bytes,uint256,string)` (STUB.sol
↪ #549-577) uses assembly
- INLINE ASM (STUB.sol#569-572)

`DefySTUB.getChainId()` (STUB.sol#1187-1193) uses assembly
- INLINE ASM (STUB.sol#1189-1191)

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation>
↪ #assembly-usage

`Address._functionCallWithValue(address,bytes,uint256,string)` (STUB.sol
↪ #549-577) is never used and should be removed

`Address.functionCall(address,bytes)` (STUB.sol#484-489) is never used and
↪ should be removed

`Address.functionCall(address,bytes,string)` (STUB.sol#497-503) is never
↪ used and should be removed

`Address.functionCallWithValue(address,bytes,uint256)` (STUB.sol#516-528)
↪ is never used and should be removed

`Address.functionCallWithValue(address,bytes,uint256,string)` (STUB.sol
↪ #536-547) is never used and should be removed

`Address.isContract(address)` (STUB.sol#423-434) is never used and should
↪ be removed

`Address.sendValue(address,uint256)` (STUB.sol#452-464) is never used and
↪ should be removed

`Context._msgData()` (STUB.sol#25-28) is never used and should be removed

`ERC20._burnFrom(address,uint256)` (STUB.sol#916-926) is never used and
↪ should be removed

`SafeMath.div(uint256,uint256)` (STUB.sol#317-319) is never used and
↪ should be removed

`SafeMath.div(uint256,uint256,string)` (STUB.sol#333-343) is never used
↪ and should be removed

`SafeMath.min(uint256,uint256)` (STUB.sol#382-384) is never used and
↪ should be removed

SafeMath.mod(uint256,uint256) (STUB.sol#357-359) is never used and
↪ should be removed

SafeMath.mod(uint256,uint256,string) (STUB.sol#373-380) is never used
↪ and should be removed

SafeMath.mul(uint256,uint256) (STUB.sol#291-303) is never used and
↪ should be removed

SafeMath.sqrt(uint256) (STUB.sol#387-398) is never used and should be
↪ removed

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation>
↪ #dead-code

Low level call in Address.sendValue(address,uint256) (STUB.sol#452-464):
- (success) = recipient.call{value: amount}() (STUB.sol#459)

Low level call in Address._functionCallWithValue(address,bytes,uint256,
↪ string) (STUB.sol#549-577):
- (success, returndata) = target.call{value: weiValue}(data) (STUB
↪ .sol#558-560)

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation>
↪ #low-level-calls

Parameter DefySTUB.mint(address,uint256)._to (STUB.sol#932) is not in
↪ mixedCase

Parameter DefySTUB.mint(address,uint256)._amount (STUB.sol#932) is not
↪ in mixedCase

Parameter DefySTUB.burn(address,uint256)._from (STUB.sol#937) is not in
↪ mixedCase

Parameter DefySTUB.burn(address,uint256)._amount (STUB.sol#937) is not
↪ in mixedCase

Variable DefySTUB._delegates (STUB.sol#949) is not in mixedCase

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation>
↪ #conformance-to-solidity-naming-conventions

Redundant expression "this (STUB.sol#26)" inContext (STUB.sol#16-29)

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation>

↔ #redundant-statements

renounceOwnership() should be declared external:

- Ownable.renounceOwnership() (STUB.sol#83-86)

transferOwnership(address) should be declared external:

- Ownable.transferOwnership(address) (STUB.sol#92-94)

decimals() should be declared external:

- ERC20.decimals() (STUB.sol#651-653)

symbol() should be declared external:

- ERC20.symbol() (STUB.sol#658-660)

totalSupply() should be declared external:

- ERC20.totalSupply() (STUB.sol#665-667)

transfer(address,uint256) should be declared external:

- ERC20.transfer(address,uint256) (STUB.sol#684-691)

allowance(address,address) should be declared external:

- ERC20.allowance(address,address) (STUB.sol#696-703)

approve(address,uint256) should be declared external:

- ERC20.approve(address,uint256) (STUB.sol#712-719)

transferFrom(address,address,uint256) should be declared external:

- ERC20.transferFrom(address,address,uint256) (STUB.sol#733-748)

increaseAllowance(address,uint256) should be declared external:

- ERC20.increaseAllowance(address,uint256) (STUB.sol#762-772)

decreaseAllowance(address,uint256) should be declared external:

- ERC20.decreaseAllowance(address,uint256) (STUB.sol#788-801)

mint(uint256) should be declared external:

- ERC20.mint(uint256) (STUB.sol#811-814)

mint(address,uint256) should be declared external:

- DefySTUB.mint(address,uint256) (STUB.sol#932-935)

burn(address,uint256) should be declared external:

- DefySTUB.burn(address,uint256) (STUB.sol#937-940)

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation>

↔ #public-function-that-could-be-declared-external

STUB.sol analyzed (7 contracts with 78 detectors), 47 result(s) found

SubDefyMaster.safeDefyTransfer(address,uint256) (subDefyMaster.sol
↪ #1037-1044) ignores return value by rewardToken.transfer(_to,
↪ rewardBal) (subDefyMaster.sol#1040)

SubDefyMaster.safeDefyTransfer(address,uint256) (subDefyMaster.sol
↪ #1037-1044) ignores return value by rewardToken.transfer(_to,
↪ _amount) (subDefyMaster.sol#1042)

SubDefyMaster.withdrawRemainder() (subDefyMaster.sol#1046-1052) ignores
↪ return value by rewardToken.transfer(feeAddress,rewardToken.
↪ balanceOf(address(this))) (subDefyMaster.sol#1051)

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation>
↪ #unchecked-transfer

SubDefyMaster.pendingReward(uint256,address) (subDefyMaster.sol#877-906)
↪ performs a multiplication on the result of a division:
-reward = multiplier.mul(rewardPerSecond).mul(pool.allocPoint).
↪ div(totalAllocPoint) (subDefyMaster.sol#894-897)
-accDefyPerShare = accDefyPerShare.add(reward.mul(1e18).div(
↪ lpSupply)) (subDefyMaster.sol#898-900)

SubDefyMaster.updatePool(uint256) (subDefyMaster.sol#917-949) performs a
↪ multiplication on the result of a division:
-reward = multiplier.mul(rewardPerSecond).mul(pool.allocPoint).
↪ div(totalAllocPoint) (subDefyMaster.sol#938-941)
-pool.accDefyPerShare = pool.accDefyPerShare.add(reward.mul(1e18)
↪ .div(lpSupply)) (subDefyMaster.sol#943-945)

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation>
↪ #divide-before-multiply

Reentrancy in SubDefyMaster.deposit(uint256,uint256) (subDefyMaster.sol
↪ #952-996):

External calls:

- safeDefyTransfer(msg.sender,pending) (subDefyMaster.sol#969)
- rewardToken.transfer(_to,rewardBal) (subDefyMaster.sol
↪ #1040)

```

- rewardToken.transfer(_to,_amount) (subDefyMaster.sol
  ↪ #1042)
- pool.lpToken.safeTransferFrom(address(msg.sender),address(this)
  ↪ ,xfAmt) (subDefyMaster.sol#974-978)
- pool.lpToken.safeTransfer(feeAddress,depositFee) (subDefyMaster
  ↪ .sol#984)

```

State variables written after the call(s):

```

- pool.lpSupply = pool.lpSupply.add(xfAmt).sub(depositFee) (
  ↪ subDefyMaster.sol#986)
- user.amount = user.amount.add(xfAmt).sub(depositFee) (
  ↪ subDefyMaster.sol#985)

```

Reentrancy in SubDefyMaster.deposit(uint256,uint256) (subDefyMaster.sol
 ↪ #952-996):

External calls:

```

- safeDefyTransfer(msg.sender,pending) (subDefyMaster.sol#969)
  - rewardToken.transfer(_to,rewardBal) (subDefyMaster.sol
    ↪ #1040)
  - rewardToken.transfer(_to,_amount) (subDefyMaster.sol
    ↪ #1042)
- pool.lpToken.safeTransferFrom(address(msg.sender),address(this)
  ↪ ,xfAmt) (subDefyMaster.sol#974-978)

```

State variables written after the call(s):

```

- pool.lpSupply = pool.lpSupply.add(xfAmt) (subDefyMaster.sol
  ↪ #989)
- user.amount = user.amount.add(xfAmt) (subDefyMaster.sol#988)

```

Reentrancy in SubDefyMaster.emergencyWithdraw(uint256) (subDefyMaster.
 ↪ sol#1026-1034):

External calls:

```

- pool.lpToken.safeTransfer(address(msg.sender),user.amount) (
  ↪ subDefyMaster.sol#1029)

```

State variables written after the call(s):

```

- pool.lpSupply = pool.lpSupply.sub(user.amount) (subDefyMaster.
  ↪ sol#1030)
- user.amount = 0 (subDefyMaster.sol#1032)

```

```

- user.rewardDebt = 0 (subDefyMaster.sol#1033)
Reentrancy in SubDefyMaster.withdraw(uint256,uint256) (subDefyMaster.sol
↳ #999-1023):
  External calls:
  - safeDefyTransfer(msg.sender,pending) (subDefyMaster.sol#1012)
    - rewardToken.transfer(_to,rewardBal) (subDefyMaster.sol
      ↳ #1040)
    - rewardToken.transfer(_to,_amount) (subDefyMaster.sol
      ↳ #1042)
  State variables written after the call(s):
  - pool.lpSupply = pool.lpSupply.sub(xfAmt) (subDefyMaster.sol
    ↳ #1016)
  - user.amount = user.amount.sub(xfAmt) (subDefyMaster.sol#1015)
Reentrancy in SubDefyMaster.withdraw(uint256,uint256) (subDefyMaster.sol
↳ #999-1023):
  External calls:
  - safeDefyTransfer(msg.sender,pending) (subDefyMaster.sol#1012)
    - rewardToken.transfer(_to,rewardBal) (subDefyMaster.sol
      ↳ #1040)
    - rewardToken.transfer(_to,_amount) (subDefyMaster.sol
      ↳ #1042)
  - pool.lpToken.safeTransfer(address(msg.sender),xfAmt) (
    ↳ subDefyMaster.sol#1017)
  State variables written after the call(s):
  - user.depositTime = block.timestamp (subDefyMaster.sol#1020)
  - user.rewardDebt = user.amount.mul(pool.accDefyPerShare).div(1
    ↳ e18) (subDefyMaster.sol#1021)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
↳ #reentrancy-vulnerabilities-1

SubDefyMaster.getUserInfo(uint256,address).deposit (subDefyMaster.sol
↳ #747) shadows:
  - SubDefyMaster.deposit(uint256,uint256) (subDefyMaster.sol
    ↳ #952-996) (function)

```

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation>
↪ #local-variable-shadowing

`SubDefyMaster.dev(address)` (`subDefyMaster.sol#1055-1058`) should emit an
↪ event for:
- `devaddr = _devaddr` (`subDefyMaster.sol#1057`)

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation>
↪ #missing-events-access-control

`SubDefyMaster.updateReward(uint256,uint256)` (`subDefyMaster.sol#760-768`)
↪ should emit an event for:
- `rewardPerSecond = _reward.div((_endTimeStamp).sub(block.
↪ timestamp))` (`subDefyMaster.sol#766`)

`SubDefyMaster.updateMultiplier(uint256)` (`subDefyMaster.sol#775-778`)
↪ should emit an event for:

- `BONUS_MULTIPLIER = multiplierNumber` (`subDefyMaster.sol#777`)

`SubDefyMaster.updateTaxRatio(uint256)` (`subDefyMaster.sol#782-785`) should
↪ emit an event for:

- `taxRatio = (10000 - _tax)` (`subDefyMaster.sol#784`)

`SubDefyMaster.add(uint256,uint256,uint256,IERC20,uint256,bool)` (
↪ `subDefyMaster.sol#793-832`) should emit an event for:
- `totalAllocPoint = totalAllocPoint.add(_allocPoint)` (
↪ `subDefyMaster.sol#819`)

`SubDefyMaster.set(uint256,uint256,uint256,uint256,uint256,bool)` (
↪ `subDefyMaster.sol#835-865`) should emit an event for:
- `totalAllocPoint = totalAllocPoint.sub(poolInfo[_pid].allocPoint
↪).add(_allocPoint)` (`subDefyMaster.sol#858-860`)

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation>
↪ #missing-events-arithmetic

`SubDefyMaster.constructor(IERC20,address,address,uint256,uint256).`
↪ `_devaddr` (`subDefyMaster.sol#722`) lacks a zero-check on :
- `devaddr = _devaddr` (`subDefyMaster.sol#730`)

SubDefyMaster.constructor(IERC20,address,address,uint256,uint256).

↪ _feeAddress (subDefyMaster.sol#723) lacks a zero-check on :

- feeAddress = _feeAddress (subDefyMaster.sol#731)

SubDefyMaster.setFeeAddress(address)._feeAddress (subDefyMaster.sol#736)

↪ lacks a zero-check on :

- feeAddress = _feeAddress (subDefyMaster.sol#738)

SubDefyMaster.dev(address)._devaddr (subDefyMaster.sol#1055) lacks a

↪ zero-check on :

- devaddr = _devaddr (subDefyMaster.sol#1057)

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation>

↪ #missing-zero-address-validation

Reentrancy in SubDefyMaster.deposit(uint256,uint256) (subDefyMaster.sol

↪ #952-996):

External calls:

- safeDefyTransfer(msg.sender,pending) (subDefyMaster.sol#969)

- rewardToken.transfer(_to,rewardBal) (subDefyMaster.sol
↪ #1040)

- rewardToken.transfer(_to,_amount) (subDefyMaster.sol
↪ #1042)

- pool.lpToken.safeTransferFrom(address(msg.sender),address(this)
↪ ,xfAmt) (subDefyMaster.sol#974-978)

- pool.lpToken.safeTransfer(feeAddress,depositFee) (subDefyMaster
↪ .sol#984)

Event emitted after the call(s):

- Deposit(msg.sender,_pid,xfAmt) (subDefyMaster.sol#995)

Reentrancy in SubDefyMaster.emergencyWithdraw(uint256) (subDefyMaster.

↪ sol#1026-1034):

External calls:

- pool.lpToken.safeTransfer(address(msg.sender),user.amount) (
↪ subDefyMaster.sol#1029)

Event emitted after the call(s):

- EmergencyWithdraw(msg.sender,_pid,user.amount) (subDefyMaster.
↪ sol#1031)

Reentrancy in SubDefyMaster.withdraw(uint256,uint256) (subDefyMaster.sol
↔ #999-1023):

External calls:

- safeDefyTransfer(msg.sender,pending) (subDefyMaster.sol#1012)
 - rewardToken.transfer(_to,rewardBal) (subDefyMaster.sol
↔ #1040)
 - rewardToken.transfer(_to,_amount) (subDefyMaster.sol
↔ #1042)
- pool.lpToken.safeTransfer(address(msg.sender),xfAmt) (
↔ subDefyMaster.sol#1017)

Event emitted after the call(s):

- Withdraw(msg.sender,_pid,xfAmt) (subDefyMaster.sol#1022)

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation>
↔ #reentrancy-vulnerabilities-3

SubDefyMaster.updateReward(uint256,uint256) (subDefyMaster.sol#760-768)

↔ uses timestamp for comparisons

Dangerous comparisons:

- require(bool,string)(_endTimeStamp > block.timestamp,invalid
↔ end timestamp) (subDefyMaster.sol#764)

SubDefyMaster.add(uint256,uint256,uint256,IERC20,uint256,bool) (
↔ subDefyMaster.sol#793-832) uses timestamp for comparisons

Dangerous comparisons:

- require(bool,string)(_rewardEndTimeStamp > block.timestamp,ADD
↔ : invalid rewardEndTimeStamp) (subDefyMaster.sol#803-806)
- block.timestamp > startTimeStamp (subDefyMaster.sol#816-818)

SubDefyMaster.set(uint256,uint256,uint256,uint256,uint256,bool) (
↔ subDefyMaster.sol#835-865) uses timestamp for comparisons

Dangerous comparisons:

- require(bool,string)(_rewardEndTimeStamp > block.timestamp,SET
↔ : invalid rewardEndTimeStamp) (subDefyMaster.sol#845-848)

SubDefyMaster.pendingReward(uint256,address) (subDefyMaster.sol#877-906)

↔ uses timestamp for comparisons

Dangerous comparisons:

- `block.timestamp > pool.lastRewardTimestamp && lpSupply != 0` (
↳ `subDefyMaster.sol#886`)
- `block.timestamp < pool.rewardEndTimestamp` (`subDefyMaster.sol`
↳ `#887-889`)

`SubDefyMaster.massUpdatePools()` (`subDefyMaster.sol#909-914`) uses

↳ timestamp for comparisons

Dangerous comparisons:

- `pid < length` (`subDefyMaster.sol#911`)

`SubDefyMaster.updatePool(uint256)` (`subDefyMaster.sol#917-949`) uses

↳ timestamp for comparisons

Dangerous comparisons:

- `block.timestamp <= pool.lastRewardTimestamp` (`subDefyMaster.sol`
↳ `#920`)
- `block.timestamp < pool.rewardEndTimestamp` (`subDefyMaster.sol`
↳ `#924-928`)

`SubDefyMaster.withdraw(uint256,uint256)` (`subDefyMaster.sol#999-1023`)

↳ uses timestamp for comparisons

Dangerous comparisons:

- `require(bool,string)(user.amount > 0,Nothing deposited.)` (
↳ `subDefyMaster.sol#1002`)

`SubDefyMaster.withdrawRemainder()` (`subDefyMaster.sol#1046-1052`) uses

↳ timestamp for comparisons

Dangerous comparisons:

- `require(bool,string)(block.timestamp > endTimeStamp.add(604800)`
↳ `,only withdrawable after 1 week from rewarding period end)`
↳ (`subDefyMaster.sol#1047-1050`)

`SubDefyMaster._getDaysSinceDeposit(uint256,address)` (`subDefyMaster.sol`

↳ `#1069-1081`) uses timestamp for comparisons

Dangerous comparisons:

- `block.timestamp < user.depositTime` (`subDefyMaster.sol#1076`)

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation>

↳ `#block-timestamp`

`Address.isContract(address)` (`subDefyMaster.sol#417-428`) uses assembly

- INLINE ASM (subDefyMaster.sol#424-426)

`Address._functionCallWithValue(address,bytes,uint256,string)` (

- ↳ subDefyMaster.sol#543-571) uses `assembly`

- INLINE ASM (subDefyMaster.sol#563-566)

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation>
 ↳ #assembly-usage

`Address.functionCall(address,bytes)` (subDefyMaster.sol#478-483) is never
 ↳ used and should be removed

`Address.functionCallWithValue(address,bytes,uint256)` (subDefyMaster.sol
 ↳ #510-522) is never used and should be removed

`Address.functionCallWithValue(address,bytes,uint256,string)` (

- ↳ subDefyMaster.sol#530-541) is never used and should be removed

`Address.sendValue(address,uint256)` (subDefyMaster.sol#446-458) is never
 ↳ used and should be removed

`Context._msgData()` (subDefyMaster.sol#10-13) is never used and should be
 ↳ removed

`SafeERC20.safeApprove(IERC20,address,uint256)` (subDefyMaster.sol
 ↳ #146-163) is never used and should be removed

`SafeERC20.safeDecreaseAllowance(IERC20,address,uint256)` (subDefyMaster.
 ↳ sol#183-200) is never used and should be removed

`SafeERC20.safeIncreaseAllowance(IERC20,address,uint256)` (subDefyMaster.
 ↳ sol#165-181) is never used and should be removed

`SafeMath.mod(uint256,uint256)` (subDefyMaster.sol#370-372) is never used
 ↳ and should be removed

`SafeMath.mod(uint256,uint256,string)` (subDefyMaster.sol#386-393) is
 ↳ never used and should be removed

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation>
 ↳ #dead-code

Low level call in `Address.sendValue(address,uint256)` (subDefyMaster.sol
 ↳ #446-458):

- (success) = recipient.call{value: amount}() (subDefyMaster.sol
 ↳ #453)

Low level call in `Address._functionCallWithValue(address,bytes,uint256, string)` (subDefyMaster.sol#543-571):
- (success, returndata) = target.call{value: weiValue}(data) (
↳ subDefyMaster.sol#552-554)

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation>
↳ #low-level-calls

Event `SubDefyMaster.feeAddressUpdated(address)` (subDefyMaster.sol#703) is
↳ not in CapWords

Parameter `SubDefyMaster.setFeeAddress(address)._feeAddress` (
↳ subDefyMaster.sol#736) is not in mixedCase

Parameter `SubDefyMaster.updateReward(uint256,uint256)._reward` (
↳ subDefyMaster.sol#760) is not in mixedCase

Parameter `SubDefyMaster.updateReward(uint256,uint256)._endTimestamp` (
↳ subDefyMaster.sol#760) is not in mixedCase

Parameter `SubDefyMaster.updateTaxRatio(uint256)._tax` (subDefyMaster.sol
↳ #782) is not in mixedCase

Parameter `SubDefyMaster.add(uint256,uint256,uint256,IERC20,uint256,bool)`
↳ `._allocPoint` (subDefyMaster.sol#794) is not in mixedCase

Parameter `SubDefyMaster.add(uint256,uint256,uint256,IERC20,uint256,bool)`
↳ `._depositFee` (subDefyMaster.sol#795) is not in mixedCase

Parameter `SubDefyMaster.add(uint256,uint256,uint256,IERC20,uint256,bool)`
↳ `._withdrawalFee` (subDefyMaster.sol#796) is not in mixedCase

Parameter `SubDefyMaster.add(uint256,uint256,uint256,IERC20,uint256,bool)`
↳ `._lpToken` (subDefyMaster.sol#797) is not in mixedCase

Parameter `SubDefyMaster.add(uint256,uint256,uint256,IERC20,uint256,bool)`
↳ `._rewardEndTimestamp` (subDefyMaster.sol#798) is not in mixedCase

Parameter `SubDefyMaster.add(uint256,uint256,uint256,IERC20,uint256,bool)`
↳ `._withUpdate` (subDefyMaster.sol#799) is not in mixedCase

Parameter `SubDefyMaster.set(uint256,uint256,uint256,uint256,uint256,bool)`
↳ `)._pid` (subDefyMaster.sol#836) is not in mixedCase

Parameter `SubDefyMaster.set(uint256,uint256,uint256,uint256,uint256,bool)`
↳ `)._allocPoint` (subDefyMaster.sol#837) is not in mixedCase

Parameter SubDefyMaster.set(uint256,uint256,uint256,uint256,uint256,bool
↪)._depositFee (subDefyMaster.sol#838) is not in mixedCase

Parameter SubDefyMaster.set(uint256,uint256,uint256,uint256,uint256,bool
↪)._withdrawalFee (subDefyMaster.sol#839) is not in mixedCase

Parameter SubDefyMaster.set(uint256,uint256,uint256,uint256,uint256,bool
↪)._rewardEndTimestamp (subDefyMaster.sol#840) is not in mixedCase

Parameter SubDefyMaster.set(uint256,uint256,uint256,uint256,uint256,bool
↪)._withUpdate (subDefyMaster.sol#841) is not in mixedCase

Parameter SubDefyMaster.getMultiplier(uint256,uint256)._from (
↪ subDefyMaster.sol#868) is not in mixedCase

Parameter SubDefyMaster.getMultiplier(uint256,uint256)._to (
↪ subDefyMaster.sol#868) is not in mixedCase

Parameter SubDefyMaster.pendingReward(uint256,address)._pid (
↪ subDefyMaster.sol#877) is not in mixedCase

Parameter SubDefyMaster.pendingReward(uint256,address)._user (
↪ subDefyMaster.sol#877) is not in mixedCase

Parameter SubDefyMaster.updatePool(uint256)._pid (subDefyMaster.sol#917)
↪ is not in mixedCase

Parameter SubDefyMaster.deposit(uint256,uint256)._pid (subDefyMaster.sol
↪ #952) is not in mixedCase

Parameter SubDefyMaster.deposit(uint256,uint256)._amount (subDefyMaster.
↪ sol#952) is not in mixedCase

Parameter SubDefyMaster.withdraw(uint256,uint256)._pid (subDefyMaster.
↪ sol#999) is not in mixedCase

Parameter SubDefyMaster.withdraw(uint256,uint256)._amount (subDefyMaster
↪ .sol#999) is not in mixedCase

Parameter SubDefyMaster.emergencyWithdraw(uint256)._pid (subDefyMaster.
↪ sol#1026) is not in mixedCase

Parameter SubDefyMaster.safeDefyTransfer(address,uint256)._to (
↪ subDefyMaster.sol#1037) is not in mixedCase

Parameter SubDefyMaster.safeDefyTransfer(address,uint256)._amount (
↪ subDefyMaster.sol#1037) is not in mixedCase

Parameter SubDefyMaster.dev(address)._devaddr (subDefyMaster.sol#1055)
↪ is not in mixedCase

Variable `SubDefyMaster.BONUS_MULTIPLIER` (`subDefyMaster.sol#690`) is not
↳ in mixedCase

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation>
↳ #conformance-to-solidity-naming-conventions

Redundant expression `"this (subDefyMaster.sol#11)"` inContext (
↳ `subDefyMaster.sol#5-14`)

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation>
↳ #redundant-statements

`renounceOwnership()` should be declared `external`:

- `Ownable.renounceOwnership()` (`subDefyMaster.sol#625-628`)

`transferOwnership(address)` should be declared `external`:

- `Ownable.transferOwnership(address)` (`subDefyMaster.sol#634-641`)

`setFeeAddress(address)` should be declared `external`:

- `SubDefyMaster.setFeeAddress(address)` (`subDefyMaster.sol`
↳ `#736-741`)

`getUserInfo(uint256,address)` should be declared `external`:

- `SubDefyMaster.getUserInfo(uint256,address)` (`subDefyMaster.sol`
↳ `#743-758`)

`updateReward(uint256,uint256)` should be declared `external`:

- `SubDefyMaster.updateReward(uint256,uint256)` (`subDefyMaster.sol`
↳ `#760-768`)

`setStartTimestamp(uint256)` should be declared `external`:

- `SubDefyMaster.setStartTimestamp(uint256)` (`subDefyMaster.sol`
↳ `#770-773`)

`updateMultiplier(uint256)` should be declared `external`:

- `SubDefyMaster.updateMultiplier(uint256)` (`subDefyMaster.sol`
↳ `#775-778`)

`updateTaxRatio(uint256)` should be declared `external`:

- `SubDefyMaster.updateTaxRatio(uint256)` (`subDefyMaster.sol`
↳ `#782-785`)

`add(uint256,uint256,uint256,IERC20,uint256,bool)` should be declared
↳ `external`:

```

- SubDefyMaster.add(uint256,uint256,uint256,IERC20,uint256,bool)
  ↳ (subDefyMaster.sol#793-832)
set(uint256,uint256,uint256,uint256,uint256,bool) should be declared
  ↳ external:
- SubDefyMaster.set(uint256,uint256,uint256,uint256,uint256,bool)
  ↳ (subDefyMaster.sol#835-865)
deposit(uint256,uint256) should be declared external:
- SubDefyMaster.deposit(uint256,uint256) (subDefyMaster.sol
  ↳ #952-996)
withdraw(uint256,uint256) should be declared external:
- SubDefyMaster.withdraw(uint256,uint256) (subDefyMaster.sol
  ↳ #999-1023)
emergencyWithdraw(uint256) should be declared external:
- SubDefyMaster.emergencyWithdraw(uint256) (subDefyMaster.sol
  ↳ #1026-1034)
withdrawRemainder() should be declared external:
- SubDefyMaster.withdrawRemainder() (subDefyMaster.sol#1046-1052)
dev(address) should be declared external:
- SubDefyMaster.dev(address) (subDefyMaster.sol#1055-1058)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
  ↳ #public-function-that-could-be-declared-external
subDefyMaster.sol analyzed (7 contracts with 78 detectors), 94 result(s)
  ↳ found

```

Compilation warnings/errors on zapper.sol:

```

Warning: Contract code size exceeds 24576 bytes (a limit introduced in
  ↳ Spurious Dragon). This contract may not be deployable on mainnet.
  ↳ Consider enabling the optimizer (with a low "runs" value!),
  ↳ turning off revert strings, or using libraries.
--> zapper.sol:1401:1:
  |
1401 | contract Zap is
  | ^ (Relevant source part starts here and spans across multiple
  ↳ lines).

```

Zap.zapInToken(address,uint256,address,address,address) (zapper.sol
↳ #1426-1455) ignores return value by IERC20(_from).transferFrom(
↳ msg.sender,address(this),amount) (zapper.sol#1437)

Zap.withdraw(address) (zapper.sol#2123-2130) ignores return value by
↳ IERC20(token).transfer(owner(),IERC20(token).balanceOf(address(
↳ this))) (zapper.sol#2129)

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation>
↳ #unchecked-transfer

Zap.zapAcross(address,uint256,address,address) (zapper.sol#1543-1570)
↳ ignores return value by IUniswapV2Router01(_toRouter).
↳ addLiquidity(pair.token0(),pair.token1(),amt0,amt1,0,0,_recipient
↳ ,block.timestamp) (zapper.sol#1560-1569)

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation>
↳ #unused-return

Zap.constructor(address)._WNATIVE (zapper.sol#1417) lacks a zero-check
↳ on :
- WNATIVE = _WNATIVE (zapper.sol#1419)

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation>
↳ #missing-zero-address-validation

Address.isContract(address) (zapper.sol#576-587) uses assembly
- INLINE ASM (zapper.sol#583-585)

Address._verifyCallResult(bool,bytes,string) (zapper.sol#781-802) uses
↳ assembly
- INLINE ASM (zapper.sol#794-797)

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation>
↳ #assembly-usage

Different versions of Solidity are used:

- Version used: ['0.8.4', '^0.8.0']
- 0.8.4 (zapper.sol#3)
- ^0.8.0 (zapper.sol#1240)
- ^0.8.0 (zapper.sol#1265)

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation>

↔ #different-pragma-directives-are-used

`Address.functionCall(address,bytes)` (zapper.sol#637-642) is never used

↔ and should be removed

`Address.functionCallWithValue(address,bytes,uint256)` (zapper.sol

↔ #669-681) is never used and should be removed

`Address.functionDelegateCall(address,bytes)` (zapper.sol#751-761) is

↔ never used and should be removed

`Address.functionDelegateCall(address,bytes,string)` (zapper.sol#769-779)

↔ is never used and should be removed

`Address.functionStaticCall(address,bytes)` (zapper.sol#714-725) is never

↔ used and should be removed

`Address.functionStaticCall(address,bytes,string)` (zapper.sol#733-743) is

↔ never used and should be removed

`Address.sendValue(address,uint256)` (zapper.sol#605-617) is never used

↔ and should be removed

`Context._msgData()` (zapper.sol#1257-1260) is never used and should be

↔ removed

`SafeERC20.safeDecreaseAllowance(IERC20,address,uint256)` (zapper.sol

↔ #1189-1210) is never used and should be removed

`SafeERC20.safeIncreaseAllowance(IERC20,address,uint256)` (zapper.sol

↔ #1173-1187) is never used and should be removed

`SafeMath.div(uint256,uint256,string)` (zapper.sol#1016-1025) is never

↔ used and should be removed

`SafeMath.mod(uint256,uint256)` (zapper.sol#972-974) is never used and

↔ should be removed

`SafeMath.mod(uint256,uint256,string)` (zapper.sol#1042-1051) is never

↔ used and should be removed

SafeMath.mul(uint256,uint256) (zapper.sol#942-944) is never used and
↔ should be removed

SafeMath.sub(uint256,uint256,string) (zapper.sol#989-998) is never used
↔ and should be removed

SafeMath.tryAdd(uint256,uint256) (zapper.sol#823-833) is never used and
↔ should be removed

SafeMath.tryDiv(uint256,uint256) (zapper.sol#877-886) is never used and
↔ should be removed

SafeMath.tryMod(uint256,uint256) (zapper.sol#893-902) is never used and
↔ should be removed

SafeMath.tryMul(uint256,uint256) (zapper.sol#856-870) is never used and
↔ should be removed

SafeMath.trySub(uint256,uint256) (zapper.sol#840-849) is never used and
↔ should be removed

TransferHelper.safeApprove(address,address,uint256) (zapper.sol
↔ #1058-1071) is never used and should be removed

TransferHelper.safeTransfer(address,address,uint256) (zapper.sol
↔ #1073-1086) is never used and should be removed

TransferHelper.safeTransferFrom(address,address,address,uint256) (zapper
↔ .sol#1088-1102) is never used and should be removed

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation>
↔ #dead-code

Pragma version^0.8.0 (zapper.sol#1240) allows old versions

Pragma version^0.8.0 (zapper.sol#1265) allows old versions

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation>
↔ #incorrect-versions-of-solidity

Low level call in Address.sendValue(address,uint256) (zapper.sol
↔ #605-617):

- (success) = recipient.call{value: amount}() (zapper.sol#612)

Low level call in Address.functionCallWithValue(address,bytes,uint256,
↔ string) (zapper.sol#689-706):

```

- (success, returndata) = target.call{value: value}(data) (zapper.
  ↪ sol#702-704)
Low level call in Address.functionStaticCall(address, bytes, string) (
  ↪ zapper.sol#733-743):
- (success, returndata) = target.staticcall(data) (zapper.sol#741)
Low level call in Address.functionDelegateCall(address, bytes, string) (
  ↪ zapper.sol#769-779):
- (success, returndata) = target.delegatecall(data) (zapper.sol
  ↪ #777)
Low level call in TransferHelper.safeApprove(address, address, uint256) (
  ↪ zapper.sol#1058-1071):
- (success, data) = token.call(abi.encodeWithSelector(0x095ea7b3,
  ↪ to, value)) (zapper.sol#1064-1066)
Low level call in TransferHelper.safeTransfer(address, address, uint256) (
  ↪ zapper.sol#1073-1086):
- (success, data) = token.call(abi.encodeWithSelector(0xa9059cbb,
  ↪ to, value)) (zapper.sol#1079-1081)
Low level call in TransferHelper.safeTransferFrom(address, address,
  ↪ address, uint256) (zapper.sol#1088-1102):
- (success, data) = token.call(abi.encodeWithSelector(0x23b872dd,
  ↪ from, to, value)) (zapper.sol#1095-1097)
Low level call in TransferHelper.safeTransferETH(address, uint256) (
  ↪ zapper.sol#1104-1107):
- (success) = to.call{value: value}(new bytes(0)) (zapper.sol
  ↪ #1105)

```

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation>
 ↪ #low-level-calls

```

Function IHyperswapRouter01.WFTM() (zapper.sol#10) is not in mixedCase
Function IUniswapV2Pair.DOMAIN_SEPARATOR() (zapper.sol#201) is not in
  ↪ mixedCase
Function IUniswapV2Pair.PERMIT_TYPEHASH() (zapper.sol#203) is not in
  ↪ mixedCase

```

Function IUniswapV2Pair.MINIMUM_LIQUIDITY() (zapper.sol#234) is not in
↳ mixedCase

Function IUniswapV2Router01.WETH() (zapper.sol#282) is not in mixedCase

Parameter Zap.zapInToken(address,uint256,address,address,address)._from
↳ (zapper.sol#1427) is not in mixedCase

Parameter Zap.zapInToken(address,uint256,address,address,address)._to (
↳ zapper.sol#1429) is not in mixedCase

Parameter Zap.zapInToken(address,uint256,address,address,address).
↳ _recipient (zapper.sol#1431) is not in mixedCase

Parameter Zap.estimateZapInToken(address,address,address,uint256)._from
↳ (zapper.sol#1458) is not in mixedCase

Parameter Zap.estimateZapInToken(address,address,address,uint256)._to (
↳ zapper.sol#1459) is not in mixedCase

Parameter Zap.estimateZapInToken(address,address,address,uint256).
↳ _router (zapper.sol#1460) is not in mixedCase

Parameter Zap.estimateZapInToken(address,address,address,uint256)._amt (
↳ zapper.sol#1461) is not in mixedCase

Parameter Zap.zapIn(address,address,address)._to (zapper.sol#1534) is
↳ not in mixedCase

Parameter Zap.zapIn(address,address,address)._recipient (zapper.sol
↳ #1536) is not in mixedCase

Parameter Zap.zapAcross(address,uint256,address,address)._from (zapper.
↳ sol#1544) is not in mixedCase

Parameter Zap.zapAcross(address,uint256,address,address)._toRouter (
↳ zapper.sol#1546) is not in mixedCase

Parameter Zap.zapAcross(address,uint256,address,address)._recipient (
↳ zapper.sol#1547) is not in mixedCase

Parameter Zap.zapOut(address,uint256,address,address)._from (zapper.sol
↳ #1573) is not in mixedCase

Parameter Zap.zapOut(address,uint256,address,address)._recipient (zapper
↳ .sol#1576) is not in mixedCase

Parameter Zap.zapOutToken(address,uint256,address,address,address)._from
↳ (zapper.sol#1627) is not in mixedCase

Parameter Zap.zapOutToken(address,uint256,address,address,address)._to (zapper.sol#1629) is not in mixedCase
Parameter Zap.zapOutToken(address,uint256,address,address,address)._recipient (zapper.sol#1631) is not in mixedCase
Parameter Zap.swapToken(address,uint256,address,address,address)._from (zapper.sol#1662) is not in mixedCase
Parameter Zap.swapToken(address,uint256,address,address,address)._to (zapper.sol#1664) is not in mixedCase
Parameter Zap.swapToken(address,uint256,address,address,address)._recipient (zapper.sol#1666) is not in mixedCase
Parameter Zap.swapToNative(address,uint256,address,address)._from (zapper.sol#1674) is not in mixedCase
Parameter Zap.swapToNative(address,uint256,address,address)._recipient (zapper.sol#1677) is not in mixedCase
Variable Zap.WNATIVE (zapper.sol#1410) is not in mixedCase
Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation>
↳ #conformance-to-solidity-naming-conventions

Redundant expression "this (zapper.sol#1258)" inContext (zapper.sol#1252-1261)

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation>
↳ #redundant-statements

Variable IHyperswapRouter01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountADesired (zapper.sol#15) is too similar to IHyperswapRouter01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountBDesired (zapper.sol#16)

Variable IUniswapV2Router01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountADesired (zapper.sol#287) is too similar to IUniswapV2Router01.addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256).amountBDesired (zapper.sol#288)

Variable Zap._swapNativeToEqualTokensAndProvide(address,address,uint256,
↪ address,address).token0Amount (zapper.sol#1837-1842) is too
↪ similar to Zap._swapNativeToEqualTokensAndProvide(address,address
↪ ,uint256,address,address).token1Amount (zapper.sol#1843-1848)

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation>
↪ #variable-names-are-too-similar

renounceOwnership() should be declared external:

- Ownable.renounceOwnership() (zapper.sol#1318-1321)

transferOwnership(address) should be declared external:

- Ownable.transferOwnership(address) (zapper.sol#1327-1334)

estimateZapInToken(address,address,address,uint256) should be declared

↪ external:

- Zap.estimateZapInToken(address,address,address,uint256) (zapper
↪ .sol#1457-1531)

Reference: <https://github.com/crytic/slither/wiki/Detector-Documentation>
↪ #public-function-that-could-be-declared-external

zapper.sol analyzed (13 contracts with 78 detectors), 75 result(s) found

Conclusion:

Most of the vulnerabilities found by the analysis have already been addressed by the smart contract code review.

5 Conclusion

In this audit, we examined the design and implementation of DefySwap contract and discovered several issues of varying severity. DefySwap team addressed 18 issues raised in the initial report and implemented the necessary fixes, while classifying the rest as a risk with low-probability of occurrence. Shellboxes' auditors advised DefySwap Team to maintain a high level of vigilance and to keep those findings in mind in order to avoid any future complications.



For a Contract Audit, contact us at contact@shellboxes.com