



Planet Finance

Smart Contract Security Audit

Prepared by: Halborn

Date of Engagement: November 18, 2021 - January 14th, 2022

Visit: Halborn.com

DOCUMENT REVISION HISTORY	8
CONTACTS	9
1 EXECUTIVE OVERVIEW	10
1.1 INTRODUCTION	11
1.2 AUDIT SUMMARY	11
1.3 TEST APPROACH & METHODOLOGY	11
RISK METHODOLOGY	12
1.4 SCOPE	14
2 ASSESSMENT SUMMARY & FINDINGS OVERVIEW	15
3 FINDINGS & TECH DETAILS	17
3.1 (HAL-01) MISUSE OF AN ORACLE – MEDIUM	19
Description	19
Code Location	19
Risk Level	20
Recommendation	20
Remediation Plan	20
3.2 (HAL-02) UNCHECKED TRANSFER – MEDIUM	21
Description	21
Code Location	21
Risk Level	22
Recommendation	22
Remediation Plan	22
3.3 (HAL-03) COMPTROLLER ERC777 LEADS TO RE-ENTRANCY – LOW	23
Description	23

Code Location	24
Recommendation	24
Remediation Plan	26
3.4 (HAL-04) INCOMPATIBILITY WITH NON-STANDARD ERC20 TOKENS - LOW	
27	
Description	27
Code Location	27
Risk Level	28
Recommendation	28
Remediation Plan	29
3.5 (HAL-05) EXTERNAL FUNCTION CALLS WITHIN LOOP - LOW	30
Description	30
Code Location	30
Risk Level	34
Recommendation	34
Remediation Plan	34
3.6 (HAL-06) MISSING ZERO ADDRESS CHECK - LOW	36
Description	36
Code Location	36
Risk Level	40
Recommendation	40
Remediation Plan	40
3.7 (HAL-07) EXPERIMENTAL FEATURES ENABLED - LOW	41
Description	41
Code Location	42
Risk Level	42

Recommendation	42
Reference	42
Remediation Plan	43
3.8 (HAL-08) PRAGMA VERSION DEPRECATED - LOW	44
Description	44
Code Location	44
Risk Level	44
Recommendation	45
Remediation Plan	45
3.9 (HAL-09) MULTIPLE PRAGMA DEFINITION - LOW	46
Description	46
Recommendation	46
Remediation Plan	47
3.10 (HAL-10) FLOATING PRAGMA - LOW	48
Description	48
Code Location	48
Risk Level	49
Recommendation	49
Remediation Plan	49
3.11 (HAL-11) IGNORE RETURN VALUES - INFORMATIONAL	50
Description	50
Code Location	50
Risk Level	51
Recommendation	51

Remediation Plan	51
3.12 (HAL-12) MISUSE OF A BOOLEAN CONSTANT - INFORMATIONAL	52
Description	52
Code Location	52
Risk Level	54
Recommendation	54
Remediation Plan	54
3.13 (HAL-13) MISSING ZERO VALUE CHECK MIGHT LEADS TO UNNECESSARY GAS CONSUMPTION - INFORMATIONAL	55
Description	55
Code Location	55
Risk Level	56
Recommendation	56
Remediation Plan	57
3.14 (HAL-14) USE OF BLOCK.TIMESTAMP - INFORMATIONAL	58
Description	58
Code Location	58
Risk Level	59
Recommendation	59
Remediation Plan	59
3.15 (HAL-15) MISSING EVENTS EMITTING - INFORMATIONAL	60
Description	60
Code Location	60
Risk Level	60
Recommendation	60
Remediation Plan	61

3.16 (HAL-16) LACK OF TEST COVERAGE AND DOCUMENTATION - INFORMATIONAL	62
Description	62
Risk Level	62
Recommendation	62
Remediation Plan	62
3.17 (HAL-17) UNIMPLEMENTED FUNCTIONS - INFORMATIONAL	63
Description	63
Code Location	63
Risk Level	64
Recommendation	65
Remediation Plan	65
3.18 (HAL-18) REDUNDANT BOOLEAN COMPARISON - INFORMATIONAL	66
Description	66
Code Location	66
Risk Level	66
Recommendation	67
Remediation Plan	67
3.19 (HAL-19) UNUSED REDUNDANT CODE - INFORMATIONAL	68
Description	68
Code Location	68
Risk Level	69
Recommendation	69
Remediation Plan	69

3.20 (HAL-20) USE OF INLINE ASSEMBLY - INFORMATIONAL	70
Description	70
Code Location	70
Risk Level	71
Recommendation	71
Remediation Plan	71
3.21 (HAL-21) WRONG CODE NOMENCLATURE - INFORMATIONAL	72
Description	72
Code Location	72
Risk Level	72
Recommendation	72
Remediation Plan	72
3.22 (HAL-22) POSSIBLE MISUSE OF PUBLIC FUNCTIONS - INFORMATIONAL	73
Description	73
Code Location	73
Risk Level	74
Recommendation	74
Remediation Plan	75
4 AUTOMATED TESTING	76
4.1 STATIC ANALYSIS REPORT	77
Description	77
Results	77
4.2 AUTOMATED SECURITY SCAN	84
Description	84

DOCUMENT REVISION HISTORY

VERSION	MODIFICATION	DATE	AUTHOR
0.1	Document Creation	11/24/2021	Alessandro Cara
0.2	Document Amended	11/25/2021	Juned Ansari
0.3	Document Updates	12/03/2021	Juned Ansari
0.4	Document Updates	12/16/2021	Juned Ansari
0.5	Draft Review	01/14/2022	Gabi Urrutia
1.0	Remediation Plan	01/27/2022	Juned Ansari
1.1	Remediation Plan Review	01/27/2022	Gabi Urrutia
1.2	Remediation Plan Update	02/09/2022	Juned Ansari
1.3	Remediation Plan Update Review	02/09/2022	Gabi Urrutia
1.4	Remediation Plan New Update	02/25/2022	Juned Ansari
1.5	Remediation Plan New Update Review	02/25/2022	Gabi Urrutia

CONTACTS

CONTACT	COMPANY	EMAIL
Rob Behnke	Halborn	Rob.Behnke@halborn.com
Steven Walbroehl	Halborn	Steven.Walbroehl@halborn.com
Gabi Urrutia	Halborn	Gabi.Urrutia@halborn.com
Juned Ansari	Halborn	Juned.Anṣari@halborn.com

EXECUTIVE OVERVIEW

1.1 INTRODUCTION

Planet Finance engaged Halborn to conduct a security assessment on their Green protocol smart contracts beginning on 18th November, 2021 and ending 14th January, 2022. This security assessment was scoped to the smart contracts code in Solidity for the Green protocol.

Although Halborn was able to identify several security risks, these were not considered posing a high risk to Planet Finance and its users.

1.2 AUDIT SUMMARY

The team at Halborn was provided ten weeks for the engagement and assigned two full-time security engineers to audit the security of the smart contract. The security engineers are a blockchain and smart-contract security experts with advanced penetration testing, smart-contract hacking, and deep knowledge of multiple blockchain protocols.

The purpose of this audit was to achieve the following:

- Ensure that all Contract functions are intended
- Identify potential security issues with the assets in scope

In summary, Halborn identified some security risks that were addressed and accepted by the Planet Finance team.

1.3 TEST APPROACH & METHODOLOGY

Halborn performed a combination of manual and automated security testing to balance efficiency, timeliness, practicality, and accuracy regarding the scope of this audit. While manual testing is recommended to uncover flaws in logic, process, and implementation; automated testing techniques help enhance coverage of the Planet Finance contract solidity code and

can quickly identify items that do not follow security best practices. The following phases and associated tools were used throughout the term of the audit:

- Research into architecture and purpose.
- Smart contract manual code review and walkthrough.
- Graphing out functionality and contract logic/connectivity/functions ([solgraph](#))
- Manual assessment of use and safety for the critical Solidity variables and functions in scope to identify any arithmetic related vulnerability classes.
- Manual testing by custom scripts.
- Scanning of solidity files for vulnerabilities, security hotspots or bugs. ([MythX](#))
- Static Analysis of security for scoped contract, and imported functions. ([Slither](#))
- Testnet deployment ([Remix IDE](#))

RISK METHODOLOGY:

Vulnerabilities or issues observed by Halborn are ranked based on the risk assessment methodology by measuring the **LIKELIHOOD** of a security incident and the **IMPACT** should an incident occur. This framework works for communicating the characteristics and impacts of technology vulnerabilities. The quantitative model ensures repeatable and accurate measurement while enabling users to see the underlying vulnerability characteristics that were used to generate the Risk scores. For every vulnerability, a risk level will be calculated on a scale of 5 to 1 with 5 being the highest likelihood or impact.

RISK SCALE - LIKELIHOOD

- 5 - Almost certain an incident will occur.
- 4 - High probability of an incident occurring.
- 3 - Potential of a security incident in the long term.
- 2 - Low probability of an incident occurring.
- 1 - Very unlikely issue will cause an incident.

RISK SCALE – IMPACT

5 – May cause devastating and unrecoverable impact or loss.

4 – May cause a significant level of impact or loss.

3 – May cause a partial impact or loss to many.

2 – May cause temporary impact or loss.

1 – May cause minimal or un-noticeable impact.

The risk level is then calculated using a sum of these two values, creating a value of 10 to 1 with 10 being the highest level of security risk.

CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL
----------	------	--------	-----	---------------

10 – CRITICAL

9 – 8 – HIGH

7 – 6 – MEDIUM

5 – 4 – LOW

3 – 1 – VERY LOW AND INFORMATIONAL

1.4 SCOPE

IN-SCOPE : Planet Finance Green protocol

The security assessment was scoped to the following smart contracts:

Listing 1

```
1 BNB_INTEREST_RATE_MODEL.sol
2 PlanetDiscountStorage.sol
3 compoundAnchored.sol
4 gGAMMA_Delegate.sol
5 GAMMA.sol
6 Reservoir.sol
7 gGAMMA_Delegator.sol
8 Gammatroller.sol
9 UniswapAnchoredView.sol
10 gToken_Delegate.sol
11 Maximillion.sol
12 UniswapConfig.sol
13 gToken_Delegator.sol
14 Ownable.sol
15 UniswapLib.sol
16 gAQUA_Delegate.sol
17 PlanetDiscountDelegate.sol
18 Unitroller.sol
19 gAQUA_Delegator.sol
20 interestRateModel.sol
21 PlanetDiscountDelegator.sol
22 changesInContracts
23 gBNB.sol
```

COMMIT-ID : 3306216998712f894e19addefca1bb21ceb613e2

OUT-OF-SCOPE : External libraries and economics attacks

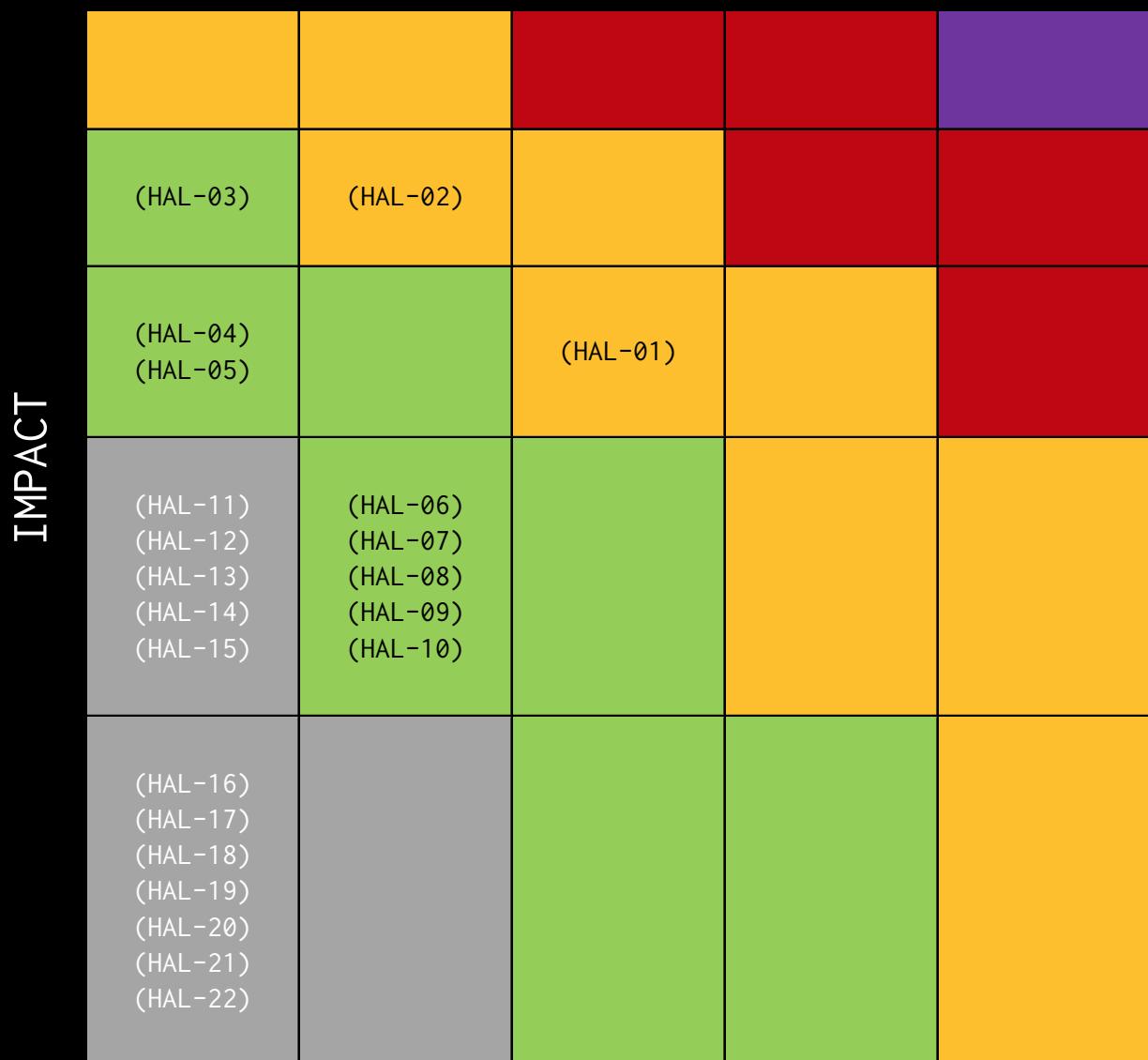
Fixed-Commit-Id :

- 9517d388979cbaf4830e300d52458f32ed54010
- 19beabb2bdf7e66857207135e5afac0717271118

2. ASSESSMENT SUMMARY & FINDINGS OVERVIEW

CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL
0	0	2	8	12

LIKELIHOOD



EXECUTIVE OVERVIEW

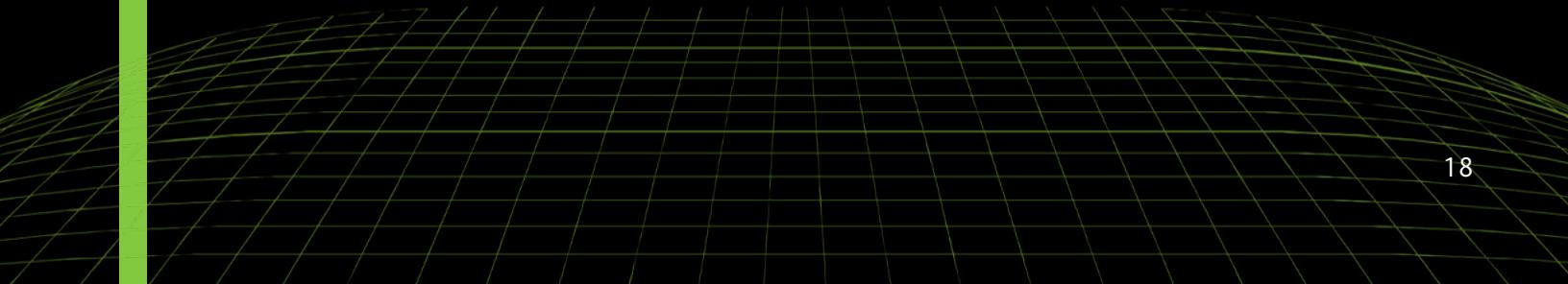
SECURITY ANALYSIS	RISK LEVEL	REMEDIATION DATE
MISUSE OF AN ORACLE	Medium	SOLVED - 02/25/2021
UNCHECKED TRANSFER	Medium	SOLVED - 02/25/2021
COMPTROLLER ERC777 LEADS TO RE-ENTRANCY	Low	RISK ACCEPTED
INCOMPATIBILITY WITH NON-STANDARD ERC20 TOKENS	Low	RISK ACCEPTED
EXTERNAL FUNCTION CALLS WITHIN LOOP	Low	RISK ACCEPTED
MISSING ZERO ADDRESS CHECK	Low	RISK ACCEPTED
EXPERIMENTAL FEATURES ENABLED	Low	RISK ACCEPTED
PRAGMA VERSION DEPRECATED	Low	RISK ACCEPTED
MULTIPLE PRAGMA DEFINITION	Low	RISK ACCEPTED
FLOATING PRAGMA	Low	RISK ACCEPTED
IGNORE RETURN VALUES	Informational	ACKNOWLEDGED
MISUSE OF A BOOLEAN CONSTANT	Informational	ACKNOWLEDGED
MISSING ZERO VALUE CHECK MIGHT LEADS TO UNNECESSARY GAS CONSUMPTION	Informational	ACKNOWLEDGED
USE OF BLOCK.TIMESTAMP	Informational	ACKNOWLEDGED
MISSING EVENTS EMITTING	Informational	ACKNOWLEDGED
LACK OF TEST COVERAGE AND DOCUMENTATION	Informational	ACKNOWLEDGED
UNIMPLEMENTED FUNCTIONS	Informational	ACKNOWLEDGED
REDUNDANT BOOLEAN COMPARISON	Informational	ACKNOWLEDGED
UNUSED REDUNDANT CODE	Informational	ACKNOWLEDGED

EXECUTIVE OVERVIEW

USE OF INLINE ASSEMBLY	Informational	ACKNOWLEDGED
WRONG CODE NOMENCLATURE	Informational	ACKNOWLEDGED
POSSIBLE MISUSE OF PUBLIC FUNCTIONS	Informational	ACKNOWLEDGED



FINDINGS & TECH DETAILS



3.1 (HAL-01) MISUSE OF AN ORACLE - MEDIUM

Description:

The `UniswapAnchoredView` contract is calling `latestAnswer` to get the last asset price. This method will return the last value, but it will not allow checking if the data is fresh. On the other hand, calling the method `latestRoundData` allows running additional validation.

Code Location:

Listing 2

```
1 function validate(address gToken) external returns (Error, bool
    valid) {
2
3
4     //Anyone can call validate if the market is calling it
        fetch the config using the Gtoken market address
        otherwise fetch the config using the gToken address
        given
5     // NOTE: We don't do any access control on msg.sender here
        . The access control is done in
        getTokenConfigByReporter,
6     // which will REVERT if an unauthorized address is passed.
7     TokenConfig memory config = getTokenConfigByGToken(msg.
        sender);
8
9     config = config.gToken == msg.sender ? config :
        getTokenConfigByGToken(gToken);
10
11    int256 currentAnswer = (AggregatorValidatorInterface(
        config.reporter).latestAnswer());
12    (Error error2,uint256 reportedPrice) =
        convertReportedPrice(config, currentAnswer);
13
14    if(error2 != Error.NO_ERROR){
15        emit Fail(error2);
16        return(error2,false);
```

```
17      }
18
19      [Redacted for brevity]
20  }
```

Risk Level:

Likelihood - 3

Impact - 3

Recommendation:

It is recommended to use the `latestRoundData` function to retrieve an asset's price instead. Checks on the return data should be introduced with proper revert messages if the price is stale or the round is `incomplete`.

Remediation Plan:

SOLVED: The `Planet Finance` team fixed this issue in the commit [9517d388979cbaf4830e300d52458f32ed54010](#) (Refer:[UniswapAnchoredView.sol](#)). As a result, now, code uses `latestRoundData` function to retrieve an asset's price.

3.2 (HAL-02) UNCHECKED TRANSFER - MEDIUM

Description:

In the contracts `Reservoir` and `Gammatroller`, the return values of the external transfer calls are not checked. It should be noted that token does not revert in case of failure and return false. If one of these tokens is used, a deposit would not revert if the transfer fails, and an attacker could deposit tokens for free.

Code Location:

`Reservoir.sol` Lines #396-398

Listing 3: Reservoir.sol (Lines 396,397,398)

```
396     token_.transfer(gammaTroller, deltaDripGammaTroller_);
397     token_.transfer(foundation, deltaDripFoundation_);
398     token_.transfer(treasury, deltaDripTreasury_);
```

`Reservoir.sol` Lines #410

Listing 4: Reservoir.sol (Line 410)

```
406     function dripOnFarm(uint _amount) external onlyFarm {
407
408         farmDripped += _amount;
409
410         token.transfer(farmAddress, _amount);
411
412         emit FarmDripped(farmDripped, block.timestamp);
413
414     }
```

Gammatroller.sol Lines #4590

Listing 5: Gammatroller.sol (Line 4590)

```
4586     function grantGammaInternal(address user, uint amount)
4587         internal returns (uint) {
4588             Gamma gamma= Gamma(getGammaAddress());
4589             uint gammaRemaining = gamma.balanceOf(address(this));
4590             if (amount > 0 && amount <= gammaRemaining) {
4591                 gamma.transfer(user, amount);
4592             }
4593             return amount;
4594 }
```

Risk Level:

Likelihood - 2

Impact - 4

Recommendation:

It is recommended to use [SafeERC20](#), or ensure that the transfer return value is checked. It is recommended to check after transfer balance.

Remediation Plan:

SOLVED: The [Planet Finance team](#) fixed this issue in the commit [19beabb2bdf7e66857207135e5afac0717271118](#). As a result, now code using the [SafeERC20](#) implementation and add the [safetransfer](#) function to the [Reservoir.sol](#) contract code. However, the team added that the team does not use the [grantGammaInternal](#) function in the [gammatroller](#) contract.

3.3 (HAL-03) COMPTROLLER ERC777
LEADS TO RE-ENTRANCY - LOW

Description:

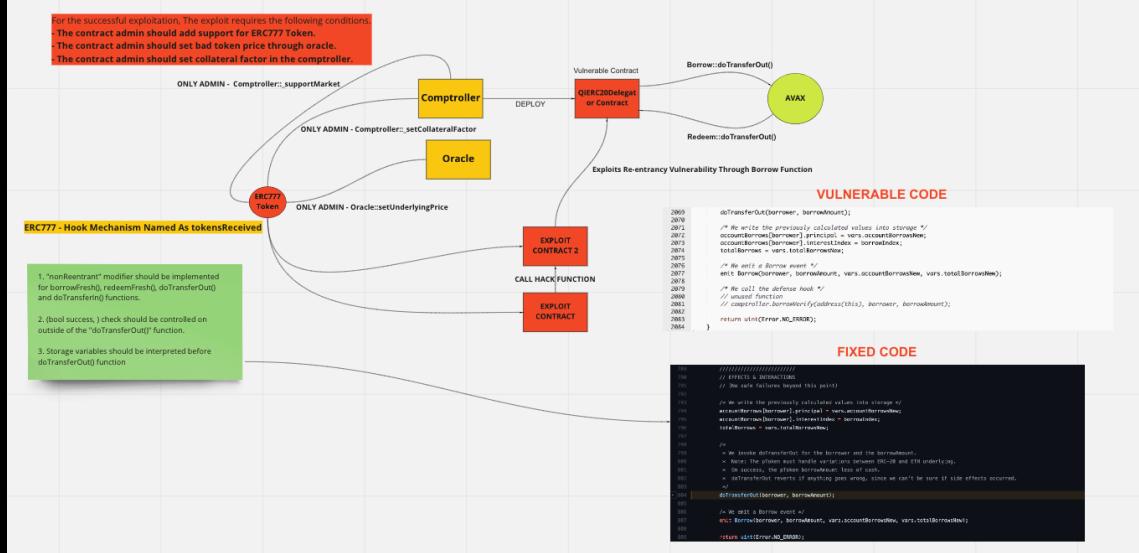
It is noted that forked comptroller contract `Unitroller.sol` uses `ERC777` that have functions vulnerable to Re-enterancy Vulnerability. It should be noted that all compound forks are vulnerable, unless `nonReentrant` modifier is used.

Exploitation Scenario:

For the successful exploitation, the exploit requires the following conditions

- The contract admin should add support for ERC777 Token.
 - The contract admin should set the bad token price through oracle.
 - The contract admin should set collateral factor in the comptroller.

Exploitation Flow-Graph:



Code Location:

Listing 6: Unitroller.sol (Line 1801)

```

1794     /*
1795      * We invoke doTransferOut for the borrower and the
1796      * borrowAmount.
1797      * Note: The gToken must handle variations between ERC-20
1798      * and ETH underlying.
1799      * On success, the gToken borrowAmount less of cash.
1800      * If doTransferOut fails despite the fact we checked pre
1801      * -conditions,
1802      * we revert because we can't be sure if side effects
1803      * occurred.
1804      */
1805
1806     vars.err = doTransferOut(borrower, borrowAmount);
1807     require(vars.err == Error.NO_ERROR, "borrow transfer out
1808             failed");
1809
1810     /* We write the previously calculated values into storage
1811     */
1812     accountBorrows[borrower].principal = vars.
1813         accountBorrowsNew;
1814     accountBorrows[borrower].interestIndex = borrowIndex;
1815     totalBorrows = vars.totalBorrowsNew;
1816
1817     /* We emit a Borrow event */
1818     emit Borrow(borrower, borrowAmount, vars.accountBorrowsNew
1819                 , vars.totalBorrowsNew);
1820
1821     /* We call the defense hook */
1822     gammatroller.borrowVerify(address(this), borrower,
1823                               borrowAmount);
1824
1825     return uint(Error.NO_ERROR);
1826 }
```

Recommendation:

It is recommended to implement the below controls

- “nonReentrant” modifier should be implemented for `borrowFresh()`, `redeemFresh()`, `doTransferIn()` and `doTransferOut()`

- (bool, success) check should be controlled on the outside of the “doTransferOut()” functionality
- Storage variables should be intercepted before doTransferOut() function

Below is the fix to the vulnerable code.

Listing 7

```
1      /* We write the previously calculated values into storage
2      */
3      accountBorrows[borrower].principal = vars.
4          accountBorrowsNew;
5      accountBorrows[borrower].interestIndex = borrowIndex;
6      totalBorrows = vars.totalBorrowsNew;
7
8      /*
9      * We invoke doTransferOut for the borrower and the
10     borrowAmount.
11     *
12     * Note: The gToken must handle variations between ERC-20
13     and ETH underlying.
14     *
15     * On success, the gToken borrowAmount less of cash.
16     *
17     * If doTransferOut fails despite the fact we checked pre-
18     -conditions,
19     *
20     * we revert because we can't be sure if side effects
21     occurred.
22     */
23     vars.err = doTransferOut(borrower, borrowAmount);
24     require(vars.err == Error.NO_ERROR, "borrow transfer out
25         failed");
26
27     /* We emit a Borrow event */
28     emit Borrow(borrower, borrowAmount, vars.accountBorrowsNew
29         , vars.totalBorrowsNew);
30
31     /* We call the defense hook */
32     gammatroller.borrowVerify(address(this), borrower,
33         borrowAmount);
34
35     return uint(Error.NO_ERROR);
36 }
```

FINDINGS & TECH DETAILS

Remediation Plan:

RISK ACCEPTED: The Planet Finance team acknowledged this issue. The team confirmed that the deployed contracts are not implemented with ERC-777, and the ERC777 standard will not be used in the future.

3.4 (HAL-04) INCOMPATIBILITY WITH NON-STANDARD ERC20 TOKENS - LOW

Description:

Some tokens (like USDT) don't correctly implement the EIP20 standard and their transfer/transferFrom functions return void, instead of a success boolean. Calling these functions with the correct EIP20 function signatures will always revert.

In the contract `Gammatroller.sol`, `gAQUA_Delegate.sol`, `gAQUA_Delegator.sol`, `gBNB.sol`, `gGAMMA_Delegate.sol`, `gGAMMA_Delegator.sol`, `gToken_Delegate.sol`, `gToken_Delegator.sol`, `Maximillion.sol`, and `Unitroller.sol` their transfer or `transferFrom` does not return a boolean for ERC20 functions if implemented. A contract compiled with solidity version greater than 0.4.22 interacting with these functions will fail to execute them, as the return value is missing. Tokens that don't correctly implement the latest EIP20 spec, like USDT, will be unusable in the smart contract as they revert the transaction because of the missing return value.

We recommend using OpenZeppelin's SafeERC20 versions with the `safeTransfer` and `safeTransferFrom` functions that handle the return value check as well as non-standard-compliant tokens.

Code Location:

Following contracts `transfer` or `transferFrom` does not return a boolean.

Listing 8

```
1 (Gammatroller.sol#1379-1441):EIP20NonStandardInterface.transfer(  
    address,uint256)  
2 (Gammatroller.sol#1379-1441):EIP20NonStandardInterface.  
    transferFrom(address,address,uint256)  
3 (gAQUA_Delegate.sol#906-968):EIP20NonStandardInterface.transfer(  
    address,uint256)  
4 (gAQUA_Delegate.sol#906-968):EIP20NonStandardInterface.  
    transferFrom(address,address,uint256)
```

```
5 (gAQUA_Delegator.sol#728-790):EIP20NonStandardInterface.transfer(
    address,uint256)
6 (gAQUA_Delegator.sol#728-790):EIP20NonStandardInterface.
    transferFrom(address,address,uint256)
7 (gBNB.sol#687-749):EIP20NonStandardInterface.transfer(address,
    uint256)
8 (gBNB.sol#687-749):EIP20NonStandardInterface.transferFrom(address,
    address,uint256)
9 (gGAMMA_Delgate.sol#766-828):EIP20NonStandardInterface.transfer(
    address,uint256)
10 (gGAMMA_Delgate.sol#766-828):EIP20NonStandardInterface.
    transferFrom(address,address,uint256)
11 (gGAMMA_Delegator.sol#14-76):EIP20NonStandardInterface.transfer(
    address,uint256)
12 (gGAMMA_Delegator.sol#14-76):EIP20NonStandardInterface.
    transferFrom(address,address,uint256)
13 (gToken_Delgate.sol#906-968):EIP20NonStandardInterface.transfer(
    address,uint256)
14 (gToken_Delgate.sol#906-968):EIP20NonStandardInterface.
    transferFrom(address,address,uint256)
15 (gToken_Delegator.sol#728-790):EIP20NonStandardInterface.transfer(
    address,uint256)
16 (gToken_Delegator.sol#728-790):EIP20NonStandardInterface.
    transferFrom(address,address,uint256)
17 (Maximillion.sol#672-734):EIP20NonStandardInterface.transfer(
    address,uint256)
18 (Maximillion.sol#672-734):EIP20NonStandardInterface.transferFrom(
    address,address,uint256)
19 (Unitroller.sol#670-732):EIP20NonStandardInterface.transfer(
    address,uint256)
20 (Unitroller.sol#670-732):EIP20NonStandardInterface.transferFrom(
    address,address,uint256)
```

Risk Level:

Likelihood - 1

Impact - 3

Recommendation:

It is recommended to use SafeERC20: `safeTransfer` and `safeTransferFrom`.

FINDINGS & TECH DETAILS

Remediation Plan:

RISK ACCEPTED: The Planet Finance team accepts the risk for this issue, as the team confirmed that Planet Finance does not use any non-standard tokens in the protocol.

3.5 (HAL-05) EXTERNAL FUNCTION CALLS WITHIN LOOP - LOW

Description:

Calls inside a loop increase Gas usage or might lead to a denial-of-service attack. In some functions discovered there is a for loop on variable `i` that iterates up to the `gTokens.length`, `assets.length`, `usersWhoHaveBorrow.length`, and `usersWhoHaveSupply.length` array length. If this integer is evaluated at large numbers, this can cause a DoS.

Code Location:

Listing 9: Gammatroller.sol (Line 4561)

```

4556     function claimGamma(address[] memory holders, GToken[] memory
4557         gTokens, bool borrowers, bool suppliers) public {
4558         for (uint i = 0; i < gTokens.length; i++) {
4559             GToken gToken = gTokens[i];
4560             require(markets[address(gToken)].isListed, "market
4561                 must be listed");
4560             if (borrowers == true) {
4561                 Exp memory borrowIndex = Exp({mantissa: gToken.
4562                     borrowIndex()});
4562                 updateGammaBorrowIndex(address(gToken),
4563                     borrowIndex);
4563                 for (uint j = 0; j < holders.length; j++) {
4564                     distributeBorrowerGamma(address(gToken),
4565                         holders[j], borrowIndex);
4565                 }
4566             }
4567             if (suppliers == true) {
4568                 updateGammaSupplyIndex(address(gToken));
4569                 for (uint j = 0; j < holders.length; j++) {
4570                     distributeSupplierGamma(address(gToken),
4571                         holders[j]);
4571                 }
4572             }
4573         }
4574         for (uint j = 0; j < holders.length; j++) {

```

```

4575         gammaAccrued[holders[j]] = grantGammaInternal(holders[
4576             j], gammaAccrued[holders[j]]);
4577     }

```

Listing 10: Gammatroller.sol (Lines 4050, 4058)

```

4035     function getHypotheticalAccountLiquidityInternal(
4036         address account,
4037         GToken gTokenModify,
4038         uint redeemTokens,
4039         uint borrowAmount) internal view returns (Error, uint,
4040                                         uint) {
4041
4042         AccountLiquidityLocalVars memory vars; // Holds all our
4043                                         calculation results
4044         uint oErr;
4045
4046         // For each asset the account is in
4047         GToken[] memory assets = accountAssets[account];
4048         for (uint i = 0; i < assets.length; i++) {
4049             GToken asset = assets[i];
4050
4051             // Read the balances and exchange rate from the gToken
4052             (oErr, vars.gTokenBalance, vars.borrowBalance, vars.
4053              exchangeRateMantissa) = asset.getAccountSnapshot(
4054                  account);
4055             if (oErr != 0) { // semi-opaque error code, we assume
4056                 NO_ERROR == 0 is invariant between upgrades
4057                 return (Error.SNAPSHOT_ERROR, 0, 0);
4058             }
4059             vars.collateralFactor = Exp({mantissa: markets[address
4060                 (asset)].collateralFactorMantissa});
4061             vars.exchangeRate = Exp({mantissa: vars.
4062                 exchangeRateMantissa});
4063
4064             // Get the normalized price of the asset
4065             vars.oraclePriceMantissa = oracle.getUnderlyingPrice(
4066                 asset);
4067             if (vars.oraclePriceMantissa == 0) {
4068                 return (Error.PRICE_ERROR, 0, 0);
4069             }
4070             vars.oraclePrice = Exp({mantissa: vars.
4071                 oraclePriceMantissa});

```

```
4063
4064      // Pre-compute a conversion factor from tokens ->
4065      // ether (normalized price value)
4066      vars.tokensToDenom = mul_(mul_(vars.collateralFactor,
4067                                    vars.exchangeRate), vars.oraclePrice);
4068
4069      // sumCollateral += tokensToDenom * gTokenBalance
4070      vars.sumCollateral = mul_ScalarTruncateAddUInt(vars.
4071                           tokensToDenom, vars.gTokenBalance, vars.
4072                           sumCollateral);
4073
4074      // sumBorrowPlusEffects += oraclePrice * borrowBalance
4075      vars.sumBorrowPlusEffects = mul_ScalarTruncateAddUInt(
4076                           vars.oraclePrice, vars.borrowBalance, vars.
4077                           sumBorrowPlusEffects);
4078
4079      // Calculate effects of interacting with gTokenModify
4080      if (asset == gTokenModify) {
4081          // redeem effect
4082          // sumBorrowPlusEffects += tokensToDenom *
4083          // redeemTokens
4084          vars.sumBorrowPlusEffects =
4085              mul_ScalarTruncateAddUInt(vars.tokensToDenom,
4086              redeemTokens, vars.sumBorrowPlusEffects);
4087
4088          // borrow effect
4089          // sumBorrowPlusEffects += oraclePrice *
4090          // borrowAmount
4091          vars.sumBorrowPlusEffects =
4092              mul_ScalarTruncateAddUInt(vars.oraclePrice,
4093              borrowAmount, vars.sumBorrowPlusEffects);
4094      }
4095
4096      // These are safe, as the underflow condition is checked
4097      // first
4098      if (vars.sumCollateral > vars.sumBorrowPlusEffects) {
4099          return (Error.NO_ERROR, vars.sumCollateral - vars.
4100                  sumBorrowPlusEffects, 0);
4101      } else {
4102          return (Error.NO_ERROR, 0, vars.sumBorrowPlusEffects -
4103                  vars.sumCollateral);
4104      }
4105  }
```

Listing 11: gBNB.sol (Lines 2660,2667)

```

2653     function updateDiscountForAll() external {
2654
2655         address[] memory usersWhoHaveBorrow = PlanetDiscount(
2656             discountLevel).returnBorrowUserArr(address(this));
2657         address[] memory usersWhoHaveSupply = PlanetDiscount(
2658             discountLevel).returnSupplyUserArr(address(this));
2659
2660         for(uint i = 0 ; i < usersWhoHaveBorrow.length ; i++){
2661             (bool exist,,,) = PlanetDiscount(discountLevel).
2662                 borrowDiscountSnap(address(this),usersWhoHaveBorrow
2663                 [i]);
2664             if(usersWhoHaveBorrow[i] != address(0) && exist){
2665                 changeUserBorrowDiscountInternal(
2666                     usersWhoHaveBorrow[i]);
2667                 changeLastBorrowBalanceAtBorrow(usersWhoHaveBorrow
2668                     [i]);
2669             }
2670         }
2671
2672         for(uint i = 0 ; i < usersWhoHaveSupply.length ; i++){
2673             (bool exist,,,) = PlanetDiscount(discountLevel).
2674                 supplyDiscountSnap(address(this),usersWhoHaveSupply
2675                 [i]);
2676             if(usersWhoHaveSupply[i] != address(0) && exist){
2677                 changeUserSupplyDiscountInternal(usersWhoHaveSupply
2678                     [i]);
2679                 changeLastExchangeRateAtSupply(usersWhoHaveSupply[i]
2680                     );
2681             }
2682         }
2683     }

```

Listing 12: Maximillion.sol (Lines 2625,2632)

```

2618     function updateDiscountForAll() external {
2619
2620         address[] memory usersWhoHaveBorrow = PlanetDiscount(
2621             discountLevel).returnBorrowUserArr(address(this));
2622         address[] memory usersWhoHaveSupply = PlanetDiscount(
2623             discountLevel).returnSupplyUserArr(address(this));

```

```
2624         for(uint i = 0 ; i < usersWhoHaveBorrow.length ; i++){
2625             (bool exist,,,) = PlanetDiscount(discountLevel).
2626                 borrowDiscountSnap(address(this),usersWhoHaveBorrow
2627 [i]);
2628             if(usersWhoHaveBorrow[i] != address(0) && exist){
2629                 changeUserBorrowDiscountInternal(
2630                     usersWhoHaveBorrow[i]);
2631                 changeLastBorrowBalanceAtBorrow(usersWhoHaveBorrow
2632 [i]);
2633             }
2634         }
2635         for(uint i = 0 ; i < usersWhoHaveSupply.length ; i++){
2636             (bool exist,,,) = PlanetDiscount(discountLevel).
2637                 supplyDiscountSnap(address(this),usersWhoHaveSupply
2638 [i]);
2639             if(usersWhoHaveSupply[i] != address(0) && exist){
2640                 changeUserSupplyDiscountInternal(usersWhoHaveSupply
2641 [i]);
2642                 changeLastExchangeRateAtSupply(usersWhoHaveSupply[i]
2643 );
2644             }
2645         }
2646     }
```

Risk Level:

Likelihood - 1

Impact - 3

Recommendation:

It is recommended to set the max length to which a for loop can iterate, and avoid usage of nested loops. If possible, use pull over push strategy for external calls.

Remediation Plan:

RISK ACCEPTED: The [Planet Finance team](#) accepts the risk for this issue, as the team confirmed that these functions have a finite amount of things

FINDINGS & TECH DETAILS

to loop through.

3.6 (HAL-06) MISSING ZERO ADDRESS CHECK - LOW

Description:

There are multiple instances found where Address validation is missing. Lack of zero address validation has been found when assigning user supplied address values to state variables directly.

Code Location:

- In contract `Gammatroller.sol`:
 - `_setBorrowCapGuardian` lacks a zero address check on `newBorrowCapGuardian`.
 - `_setPauseGuardian` lacks a zero address check on `newPauseGuardian`.
 - `_setPendingAdmin` lacks a zero address check on `newPendingAdmin`, `newPendingAdmin`.
 - `_setPendingImplementation` lacks a zero address check on `newPendingImplementation`.
- In contract `gAQUA_Delegate.sol`:
 - `initialize` lacks a zero address check on `underlying_`.
 - `_setPendingAdmin` lacks a zero address check on `newPendingAdmin`.
- In contract `gAQUA_Delegator.sol`:
 - `changeAddress` lacks a zero address check on `_newgGammaAddress`, `_newGammatroller`, `_newOracle`.
 - `constructor` lacks a zero address check on `admin_`.
 - `_setImplementation` lacks a zero address check on `implementation_`.
- In contract `gBNB.sol`:
 - `_setPendingAdmin` lacks a zero address check on `newPendingAdmin`.
- In contract `gGAMMA_Delegate.sol`:

- `initialize` lacks a zero address check on `underlying_`.
 - `_setPendingAdmin` lacks a zero address check on `newPendingAdmin`.
 - `_setWithdrawFeeAddress` lacks a zero address check on `newWithdrawFeeAddress`.
- In contract `gGAMMA_Delegator.sol`:
 - `constructor` lacks a zero address check on `admin_`.
 - `_setImplementation` lacks a zero address check on `implementation_`
- In contract `gToken_Delegate.sol`:
 - `initialize` lacks a zero address check on `underlying_`.
 - `_setPendingAdmin` lacks a zero address check on `newPendingAdmin`.
- In contract `gToken_Delegator.sol`:
 - `changeAddress` lacks a zero address check on `_newgGammaAddress`,
`_newGammatroller`, `_newOracle`.
 - `constructor` lacks a zero address check on `admin_`.
 - `_setImplementation` lacks a zero address check on `implementation_`
- In contract `Maximillion.sol`:
 - `_setPendingAdmin` lacks a zero address check on `newPendingAdmin`.
- In contract `PlanetDiscountDelegate.sol`:
 - `changeAddress` lacks a zero address check on `_newgGammaAddress`,
`_newGammatroller`, `_newOracle`.
- In contract `PlanetDiscountDelegator.sol`:
 - `constructor` lacks a zero address check on `admin_`.
 - `_setAdmin` lacks a zero address check on `newAdmin`.
 - `_setImplementation` lacks a zero address check on `implementation_`
- In contract `Reservoir.sol`:
 - `changeFoundationAddress` lacks a zero address check on
`_newFoundation`.
 - `changeTreasuryAddress` lacks a zero address check on `_newTreasury`

- In contract Unitroller.sol:
 - `_setPendingAdmin` lacks a zero address check on `newPendingAdmin`, `newPendingAdmin`.
 - `_setPendingImplementation` lacks a zero address check on `newPendingImplementation`.

Zero Address Validation missing before address assignment to these state variable.

Listing 13: Gammatroller.sol

```

1 pendingGammatrollerImplementation = newPendingImplementation
    (#448)
2 pendingAdmin = newPendingAdmin (#497)
3 pendingAdmin = newPendingAdmin (#3004)
4 borrowCapGuardian = newBorrowCapGuardian (#4304)
5 pauseGuardian = newPauseGuardian (#4324)

```

Listing 14: gAQUA_Delegate.sol

```

1 underlying = underlying_ (#2918)
2 pendingAdmin = newPendingAdmin (#2560)

```

Listing 15: gAQUA_Delegator.sol

```

1 gGammaAddress = _newgGammaAddress (#636)
2 gammatroller = _newGammatroller (#637)
3 oracle = _newOracle (#638)
4 admin = admin_ (#1225)
5 implementation = implementation_ (#1242)

```

Listing 16: gBNB.sol

```

1 pendingAdmin = newPendingAdmin (#2269)

```

Listing 17: gGAMMA_Delegate.sol

```

1 underlying = underlying_ (#2833)
2 pendingAdmin = newPendingAdmin (#2434)

```

```
3 withdrawFeeAddress = newWithdrawFeeAddress (#2705)
```

Listing 18: gGAMMA_Delegator.sol

```
1 admin = admin_ (#573)
2 implementation = implementation_ (#590)
```

Listing 19: gToken_Delegate.sol

```
1 underlying = underlying_ (#2919)
2 pendingAdmin = newPendingAdmin (#2561)
```

Listing 20: gToken_Delegator.sol

```
1 gGammaAddress = _newgGammaAddress (#636)
2 gammatroller = _newGammatroller (#637)
3 oracle = _newOracle (#638)
4 admin = admin_ (#1225)
5 implementation = implementation_ (#1242)
```

Listing 21: Maximillion.sol

```
1 pendingAdmin = newPendingAdmin (#2240)
```

Listing 22: PlanetDiscountDelegate.sol

```
1 gGammaAddress = _newgGammaAddress (#743)
2 gammatroller = _newGammatroller (#744)
3 oracle = _newOracle (#745)
```

Listing 23: PlanetDiscountDelegator.sol

```
1 admin = admin_ (#14)
2 implementation = implementation_ (#25)
3 admin = newAdmin (#38)
```

Listing 24: Reservoir.sol

```
1 foundation = _newFoundation (#420)
2 treasury = _newTreasury (#430)
```

Listing 25: Unitroller.sol

```
1 pendingAdmin = newPendingAdmin (#2119)
2 pendingGammatrollerImplementation = newPendingImplementation
    (#2499)
3 pendingAdmin = newPendingAdmin (#2550)
```

Risk Level:

Likelihood - 2

Impact - 2

Recommendation:

Although administrative restrictions are imposed to this function due to the role-based access controls (RBAC), it is recommended to add proper address validation when assigning user supplied input to a variable. This could be as simple as using the following statement:

Listing 26

```
1 require(address_input != 0, "Address is zero")
```

Remediation Plan:

RISK ACCEPTED: The [Planet Finance team](#) accepts the risk for this issue.

3.7 (HAL-07) EXPERIMENTAL FEATURES ENABLED - LOW

Description:

ABIEncoderV2 is enabled to be able to pass struct type into a function, both web3 and another contract. The use of experimental features could be dangerous on live deployments. The experimental ABI encoder does not handle non-integer values shorter than 32 bytes properly. This applies to bytesNN types, bool, enum and other types when they are part of an array or a struct and encoded directly from storage. This means these storage references have to be used directly inside abi.encode(...) as arguments in external function calls or in event data without prior assignment to a local variable. Using return does not trigger the bug. The types bytesNN and bool will result in corrupted data, while enum might lead to an invalid revert.

Furthermore, arrays with elements shorter than 32 bytes may not be handled correctly, even if the base type is an integer type. Encoding such arrays in the way described above can lead to other data in the encoding being overwritten if the number of elements encoded is not a multiple of the number of elements that fit a single slot. If nothing follows the array in the encoding (note that dynamically sized arrays are always encoded after statically sized arrays with statically sized content), or if only a single array is encoded, no other data is overwritten. There are known bugs that are publicly released while using this feature. However, the bug only manifests itself when all the following conditions are met:

Storage data involving arrays or structs is sent directly to an external function call, to abi.encode or to event data without prior assignment to a local (memory) variable.

There is an array that contains elements with size less than 32 bytes or a struct that has elements that share a storage slot or members of type bytesNN shorter than 32 bytes.

In addition to that, in the following situations, the code is NOT affected:

All the structs or arrays only use uint256 or int256 types.

If only using integer types (that may be shorter) and only encode at most one array at a time.

If only returning such data and do not use it in abi.encode, external calls or event data.

ABIEncoderV2 is enabled to be able to pass struct type into a function, both web3 and another contract. Naturally, any bug can have wildly varying consequences depending on the program control flow, but we expect that this is more likely to lead to malfunction than exploitability. The bug, when triggered, will under certain circumstances send corrupt parameters on method invocations to other contracts.

Code Location:

Listing 27

```
1 Gammatroller.sol#8: pragma experimental ABIEncoderV2;
2 UniswapConfig.sol#4: pragma experimental ABIEncoderV2;
3 GAMMA.sol#6: pragma experimental ABIEncoderV2;
4 UniswapAnchoredView.sol#4: pragma experimental ABIEncoderV2;
```

Risk Level:

Likelihood - 2

Impact - 2

Recommendation:

When possible, do not use experimental features in the final live deployment. Validate and check that all the conditions above are true for integers and arrays (i.e., all using uint256).

Reference:

[Solidity Optimizer and ABIEncoderV2 Bug](#)

FINDINGS & TECH DETAILS

Remediation Plan:

RISK ACCEPTED: The Planet Finance team accepts the risk for this issue.

3.8 (HAL-08) PRAGMA VERSION DEPRECATED - LOW

Description:

In the Green contracts, the current pragma version in use for the contracts is pragma `0.5.16`, `0.5.17`, and `0.5.8`. While this version is still functional, and some security issues safely implemented by mitigating contracts with other utility contracts such as SafeMath.sol, the risk to the long-term sustainability and integrity of the solidity code increases.

Code Location:

Listing 28

```
1 BNB_INTEREST_RATE_MODEL.sol:pragma solidity ^0.5.16;
2 GAMMA.sol:pragma solidity ^0.5.16;
3 Gammatroller.sol:pragma solidity ^0.5.16;
4 Maximillion.sol:pragma solidity ^0.5.8;
5 Maximillion.sol:pragma solidity ^0.5.16;
6 PlanetDiscountDelegate.sol:pragma solidity ^0.5.16;
7 PlanetDiscountDelegator.sol:pragma solidity ^0.5.16;
8 PlanetDiscountStorage.sol:pragma solidity ^0.5.17;
9 Reservoir.sol:pragma solidity ^0.5.17;
10 Unitroller.sol:pragma solidity ^0.5.8;
11 gAQUA_Delegate.sol:pragma solidity ^0.5.16;
12 gAQUA_Delegator.sol:pragma solidity ^0.5.16;
13 gBNB.sol:pragma solidity ^0.5.8;
14 gGAMMA_Delegate.sol:pragma solidity ^0.5.16;
15 gGAMMA_Delegator.sol:pragma solidity ^0.5.16;
16 gToken_Delegate.sol:pragma solidity ^0.5.16;
17 gToken_Delegator.sol:pragma solidity ^0.5.16;
18 interestRateModel.sol:pragma solidity ^0.5.16;
```

Risk Level:

Likelihood - 2

Impact - 2

Recommendation:

If possible, consider using the latest stable pragma version that has been thoroughly tested to prevent potential undiscovered vulnerabilities, such as pragma between `0.6.12 - 0.7.6`, or the latest pragma `0.8.9`. For example, after the `Solidity version 0.8.0` arithmetic operations revert to underflow and overflow by default, by using this version, utility contracts like `SafeMath.sol` would not be needed.

Remediation Plan:

RISK ACCEPTED: The `Planet Finance team` accepts the risk for this issue.

3.9 (HAL-09) MULTIPLE PRAGMA DEFINITION - LOW

Description:

Planet Finance contracts use different pragma versions. Contracts should be deployed with the same compiler version and flags that they have been tested with thoroughly.

Locking the pragma helps to ensure that contracts do not accidentally get deployed using another pragma, for example, either an outdated pragma version that might introduce bugs that affect the contract system negatively or a recently released pragma version which has not been extensively tested. The latest pragma version (0.8.9) was released in September 2021. Many pragma versions have been lately released, going from version 0.7.x to the recently released version 0.8.x. in just 8 months.

Reference: <https://github.com/ethereum/solidity/releases>

In the Solidity Github repository, there is a json file with all bugs finding in the different compiler versions. It should be noted that pragma 0.6.12 and 0.7.6 are widely used by Solidity developers and have been extensively tested in many security audits.

Reference: https://github.com/ethereum/solidity/blob/develop/docs/bugs_by_version.json

Recommendation:

Consider locking and using a single pragma version without known bugs for the compiler version. If possible, consider using the latest stable pragma version that has been thoroughly tested to prevent potential undiscovered vulnerabilities, such as pragma between 0.6.12 - 0.7.6, or the latest pragma 0.8.0 - 0.8.9. For example, after the Solidity version 0.8.0 arithmetic operations revert to underflow and overflow by default, by using this version, utility contracts like SafeMath.sol would not be needed.

FINDINGS & TECH DETAILS

Remediation Plan:

RISK ACCEPTED: The Planet Finance team accepts the risk for this issue.

3.10 (HAL-10) FLOATING PRAGMA - LOW

Description:

Contracts should be deployed with the same compiler version and flags used during development and testing. Locking the pragma helps to ensure that contracts do not accidentally get deployed using another pragma. For example, an outdated pragma version might introduce bugs that affect the contract system negatively or recently released pragma versions may have unknown security vulnerabilities.

Code Location:

The following contracts and pragma versions where not locked:

Listing 29

```
1 BNB_INTEREST_RATE_MODEL.sol:pragma solidity ^0.5.16;
2 GAMMA.sol:pragma solidity ^0.5.16;
3 Gammatroller.sol:pragma solidity ^0.5.16;
4 gAQUA_Delegate.sol:pragma solidity ^0.5.16;
5 gAQUA_Delegator.sol:pragma solidity ^0.5.16;
6 gBNB.sol:pragma solidity ^0.5.8;
7 gGAMMA_Delegate.sol:pragma solidity ^0.5.16;
8 gGAMMA_Delegator.sol:pragma solidity ^0.5.16;
9 gToken_Delegate.sol:pragma solidity ^0.5.16;
10 gToken_Delegator.sol:pragma solidity ^0.5.16;
11 interestRateModel.sol:pragma solidity ^0.5.16;
12 Maximillion.sol:pragma solidity ^0.5.16;
13 Maximillion.sol:pragma solidity ^0.5.8;
14 Ownable.sol:pragma solidity ^0.6.12;
15 PlanetDiscountDelegate.sol:pragma solidity ^0.5.16;
16 PlanetDiscountDelegator.sol:pragma solidity ^0.5.16;
17 PlanetDiscountStorage.sol:pragma solidity ^0.5.17;
18 Reservoir.sol:pragma solidity ^0.5.17;
19 UniswapAnchoredView.sol:pragma solidity ^0.6.12;
20 UniswapConfig.sol:pragma solidity ^0.6.12;
21 UniswapLib.sol:pragma solidity ^0.6.12;
22 Unitroller.sol:pragma solidity ^0.5.8;
```

Risk Level:

Likelihood - 2

Impact - 2

Recommendation:

Consider locking the pragma version as it is not recommended to use a floating pragma in production. Apart from just locking the pragma version in the code, the sign (`>=`) needs to be removed. It is possible to lock the pragma by fixing the version both in `truffle-config.js` for the Truffle framework or in `hardhat.config.js` for the HardHat framework.

Remediation Plan:

RISK ACCEPTED: The Planet Finance team accepts the risk for this issue.

3.11 (HAL-11) IGNORE RETURN VALUES - INFORMATIONAL

Description:

The return value of an external call is not stored in a local or state variable. In contract `Gammatroller.sol` and `gBNB.sol`, there are instances where external methods are being called, and return value are being ignored.

Code Location:

```
Listing 30: Gammatroller.sol (Line 4253)

4244     function _supportMarket(GToken gToken) external returns (uint)
4245     {
4246         if (msg.sender != admin) {
4247             return fail(Error.UNAUTHORIZED, FailureInfo.
4248                         SUPPORT_MARKET_OWNER_CHECK);
4249         }
4250         if (markets[address(gToken)].isListed) {
4251             return fail(Error.MARKET_ALREADY_LISTED, FailureInfo.
4252                         SUPPORT_MARKET_EXISTS);
4253         }
4254         gToken.isGToken(); // Sanity check to make sure its really
4255         // Note that isGammaed is not in active use anymore
4256         markets[address(gToken)] = Market({isListed: true,
4257                                         isGammaed: false, collateralFactorMantissa: 0});
4258         _addMarketInternal(address(gToken));
4259         emit MarketListed(gToken);
4260         return uint(Error.NO_ERROR);
4261     }
```

Listing 31: gBNB.sol (Line 1459)

```
1456     function accrueInterest() public returns (uint) {
1457
1458         if(isPriceUpdate)
1459             PriceOracle(gammatroller.oracle()).validate(address(this))
1460             ;
1461
1461     AccrueInterestLocalVars memory vars;
```

Risk Level:

Likelihood - 1

Impact - 2

Recommendation:

Add return value check to avoid unexpected crash of the contract. Return value check will help in handling the exceptions better way.

Remediation Plan:

ACKNOWLEDGED: The Planet Finance team acknowledged this issue. However, the team claims that when the external call Gtoken.isGToken() is false, it will automatically revert.

3.12 (HAL-12) MISUSE OF A BOOLEAN CONSTANT - INFORMATIONAL

Description:

It was observed that some in-scope contracts misuse the boolean constant `false` and are directly used in a conditional `if` statement. Boolean constants in code have only a few legitimate uses. Other purposes (in complex expressions, as conditionals) indicate an error or, most likely, the persistence of faulty code.

Code Location:

Listing 32

```
1 Gammatroller.sol-3579- // Shh - we don't ever want this hook to be
                           marked pure
2 Gammatroller.sol:3580: if (false) {
3 Gammatroller.sol-3581- maxAssets = maxAssets;
4 Gammatroller.sol-3582- }
5 --
6 Gammatroller.sol-3715- // Shh - we don't ever want this hook to be
                           marked pure
7 Gammatroller.sol:3716: if (false) {
8 Gammatroller.sol-3717- maxAssets = maxAssets;
9 Gammatroller.sol-3718- }
10 --
11 Gammatroller.sol-3771- // Shh - we don't ever want this hook to be
                           marked pure
12 Gammatroller.sol:3772: if (false) {
13 Gammatroller.sol-3773- maxAssets = maxAssets;
14 Gammatroller.sol-3774- }
15 --
16 Gammatroller.sol-3846- // Shh - we don't ever want this hook to be
                           marked pure
17 Gammatroller.sol:3847: if (false) {
18 Gammatroller.sol-3848- maxAssets = maxAssets;
19 Gammatroller.sol-3849- }
20 --
```

```
21 Gammatroller.sol:3909- // Shh - we don't ever want this hook to be
   marked pure
22 Gammatroller.sol:3910: if (false) {
23 Gammatroller.sol:3911- maxAssets = maxAssets;
24 Gammatroller.sol:3912- }
25 --
26 Gammatroller.sol:3956- // Shh - we don't ever want this hook to be
   marked pure
27 Gammatroller.sol:3957: if (false) {
28 Gammatroller.sol:3958- maxAssets = maxAssets;
29 Gammatroller.sol:3959- }
30 --
31 gAQUA_Delegate.sol:3175- // Shh -- we don't ever want this hook to
   be marked pure
32 gAQUA_Delegate.sol:3176: if (false) {
33 gAQUA_Delegate.sol:3177- implementation = address(0);
34 gAQUA_Delegate.sol:3178- }
35 --
36 gAQUA_Delegate.sol:3186- function _resignImplementation() public {
37 gAQUA_Delegate.sol:3187- // Shh -- we don't ever want this hook to
   be marked pure
38 gAQUA_Delegate.sol:3188: if (false) {
39 gAQUA_Delegate.sol:3189- implementation = address(0);
40 gAQUA_Delegate.sol:3190- }
41 --
42 gGAMMA_Delegate.sol:3044-
43 gGAMMA_Delegate.sol:3045- // Shh -- we don't ever want this hook
   to be marked pure
44 gGAMMA_Delegate.sol:3046: if (false) {
45 gGAMMA_Delegate.sol:3047- implementation = address(0);
46 gGAMMA_Delegate.sol:3048- }
47 --
48 gGAMMA_Delegate.sol:3056- function _resignImplementation() public
   {
49 gGAMMA_Delegate.sol:3057- // Shh -- we don't ever want this hook
   to be marked pure
50 gGAMMA_Delegate.sol:3058: if (false) {
51 gGAMMA_Delegate.sol:3059- implementation = address(0);
52 gGAMMA_Delegate.sol:3060- }
53 --
54 gToken_Delegate.sol:3175-
55 gToken_Delegate.sol:3176- // Shh -- we don't ever want this hook
   to be marked pure
56 gToken_Delegate.sol:3177: if (false) {
```

```
57 gToken_Delegate.sol-3178- implementation = address(0);
58 gToken_Delegate.sol-3179- }
59 --
60 gToken_Delegate.sol-3187- function _resignImplementation() public
  {
61 gToken_Delegate.sol-3188- // Shh -- we don't ever want this hook
    to be marked pure
62 gToken_Delegate.sol:3189: if (false) {
63 gToken_Delegate.sol-3190- implementation = address(0);
64 gToken_Delegate.sol-3191- }
```

Risk Level:

Likelihood - 1

Impact - 2

Recommendation:

Consider updating the logic and simplifying the conditional statement. Remove these conditional statements to increase the readability of the code.

Remediation Plan:

ACKNOWLEDGED: The Planet Finance team acknowledged this issue and claims that the team doesn't use these functions, so the team sets these statements to false so that code won't execute.

3.13 (HAL-13) MISSING ZERO VALUE CHECK MIGHT LEADS TO UNNECESSARY GAS CONSUMPTION - INFORMATIONAL

Description:

It was observed that in the contract `gToken_Delegate.sol` function `_addReserves` lack a zero value validation before an internal call to function `_addReservesInternal` against the valid caller supplied value to state variable `addAmount`. Likewise, function `_reduceReserves` also lacks a zero value validation. It leads to unnecessary internal function checks and calculations, in such cases, gas can be optimized by a prior ‘require’ check to an internal function call.

Code Location:

Zero Address Validation missing before address assignment to these state variables.

Listing 33: gToken_Delegate.sol (Lines 3014,3015)

```
3014     function _addReserves(uint addAmount) external returns (uint)
3015     {
3016         return _addReservesInternal(addAmount);
3017     }
```

Listing 34: gToken_Delegate.sol

```
2669     function _addReservesInternal(uint addAmount) internal
270         nonReentrant returns (uint) {
271         uint error = accrueInterest();
272         if (error != uint(Error.NO_ERROR)) {
273             // accrueInterest emits logs on errors, but on top of
274             // that we want to log the fact that an attempted
275             // reduce reserves failed.
276             return fail(Error(error), FailureInfo(
277                 ADD_RESERVES_ACCRUE_INTEREST_FAILED));
278         }
279     }
```

```
2675  
2676      // _addReservesFresh emits reserve-addition-specific logs  
2677      // on errors, so we don't need to.  
2678      (error, ) = _addReservesFresh(addAmount);  
2679      return error;  
2679 }
```

Listing 35: gToken_Delegate.sol (Lines 2732,2739)

```
2732  function _reduceReserves(uint reduceAmount) external  
2733      nonReentrant returns (uint) {  
2734          uint error = accrueInterest();  
2735          if (error != uint(Error.NO_ERROR)) {  
2736              // accrueInterest emits logs on errors, but on top of  
2737              // that we want to log the fact that an attempted  
2738              // reduce reserves failed.  
2739              return fail(Error(error), FailureInfo.  
2740                  REDUCE_RESERVES_ACCRUE_INTEREST_FAILED);  
2741          }  
2742          // _reduceReservesFresh emits reserve-reduction-specific  
2743          // logs on errors, so we don't need to.  
2744          return _reduceReservesFresh(reduceAmount);  
2745      }
```

Risk Level:

Likelihood - 1

Impact - 2

Recommendation:

It is recommended to add a zero value validation on caller supplied input to state variable. This could be as simple as using the following statement:

Listing 36

```
1 require(addAmount > 0, "Reserve Amount to add cannot be zero")
```

FINDINGS & TECH DETAILS

Remediation Plan:

ACKNOWLEDGED: The Planet Finance team acknowledged this issue.

3.14 (HAL-14) USE OF BLOCK.TIMESTAMP - INFORMATIONAL

Description:

During a manual review, it was observed the usage of `block.timestamp` in several smart contracts. The contract developers should be aware that `block.timestamp` does not mean current time. The value of `block.timestamp` can be influenced by miners to a certain degree, so the testers should be warned that this may have some risk if miners collude on time manipulation to influence the price of oracles. Miners can influence the timestamp by a tolerance of 900 seconds.

Code Location:

Contracts that make use of `block.timestamp` or `now`:

Listing 37: PlanetDiscountDelegate.sol

```
1 #867: _borrowDis.lastUpdated = block.timestamp;
2 #919: _dis.lastUpdated = block.timestamp;
3 #930: _dis.lastUpdated = block.timestamp;
4 #938: _dis.lastUpdated = block.timestamp;
```

Listing 38: Gammatroller.sol

```
1 #174: require(now <= expiry, "Gamma::delegateBySig: signature
expired");
```

Listing 39: GAMMA.sol

```
1 #172: require(now <= expiry, "Gamma::delegateBySig: signature
expired");
```

Listing 40: UniswapAnchoredView.sol

```
1 #115: oldObservations[symbolHash].timestamp = block.timestamp;
2 #116: newObservations[symbolHash].timestamp = block.timestamp;
3 #138: oldObservations[symbolHash].timestamp = block.timestamp;
4 #139: newObservations[symbolHash].timestamp = block.timestamp;
5 #419: (uint nowCumulativePrice, uint oldCumulativePrice, uint
       oldTimestamp) = pokeWindowValues(config);
6 #422: require(block.timestamp > oldTimestamp, "now must come after
       before");
7 #424: uint timeElapsed = block.timestamp - oldTimestamp;
8 #428: FixedPoint.uq112x112 memory priceAverage = FixedPoint.
       uq112x112(uint224((nowCumulativePrice - oldCumulativePrice) /
       timeElapsed));
9 #476: uint timeElapsed = block.timestamp - newObservation.
       timestamp;
10 #481: newObservations[symbolHash].timestamp = block.timestamp;
```

Risk Level:

Likelihood - 1

Impact - 2

Recommendation:

Use `block.number` instead of `block.timestamp` or `now` to reduce the risk of MEV attacks. Check if the timescale of the project occurs across years, days, and months rather than seconds. If possible, it is recommended to use Oracles.

Remediation Plan:

ACKNOWLEDGED: The Planet Finance team acknowledged this issue.

3.15 (HAL-15) MISSING EVENTS EMITTING - INFORMATIONAL

Description:

It has been observed that important functionality is missing emitting event for a function on the `Gammatroller.sol` contract. These functions should emit events. Events are a method of informing the transaction initiator about the actions taken by the called function. It logs its emitted parameters in a specific log history, which can be accessed outside the contract using some filter parameters. These functions should emit events.

Code Location:

```
function initialize
```

Listing 41: Gammatroller.sol

```
1 initialExchangeRateMantissa = initialExchangeRateMantissa_
 (#1897)
```

Risk Level:

Likelihood - 1

Impact - 2

Recommendation:

For best security practices, consider as much as possible, declaring events at the end of the function. Events can be used to detect the end of the operation.

FINDINGS & TECH DETAILS

Remediation Plan:

ACKNOWLEDGED: The issue was acknowledged by the Planet Finance team.

3.16 (HAL-16) LACK OF TEST COVERAGE AND DOCUMENTATION - INFORMATIONAL

Description:

It was observed that the smart contract set lacked test cases and documentation. Whilst this does not correspond to a direct security risk, it is always recommended to write thorough tests and documentation to ensure that the code is easily maintainable and behaves correctly.

Risk Level:

Likelihood - 1

Impact - 1

Recommendation:

Halborn recommends writing test cases to ensure that critical functionality of the smart contracts behave correctly and to write documentation to ensure that code is easily maintainable.

Remediation Plan:

ACKNOWLEDGED: The issue was acknowledged by the Planet Finance team.

3.17 (HAL-17) UNIMPLEMENTED FUNCTIONS - INFORMATIONAL

Description:

During the test, It has been observed that contract `Gammatroller.sol` and `Unitroller.sol` does not implement the below-listed function. As a result, the contract might not be properly compiled. All unimplemented functions must be implemented on a contract that is meant to be used.

Code Location:

```
Listing 42: Gammatroller.sol (Lines 3293,3299,3306)
3293     function getCashPrior() internal view returns (uint);
3294
3295     /**
3296      * @dev Performs a transfer in, reverting upon failure.
3297      *       Returns the amount actually transferred to the protocol,
3298      *       in case of a fee.
3299      * This may revert due to insufficient balance or
3300      *       insufficient allowance.
3301      */
3302     function doTransferIn(address from, uint amount) internal
3303         returns (uint);
3304
3305     /**
3306      * @dev Performs a transfer out, ideally returning an
3307      *       explanatory error code upon failure tather than reverting.
3308      * If caller has not called checked protocol's balance, may
3309      *       revert due to insufficient cash held in the contract.
3310      * If caller has checked protocol's balance, and verified it
3311      *       is >= amount, this should not revert in normal conditions.
3312      */
3313     function doTransferOut(address payable to, uint amount)
3314         internal;
```

Listing 43: Unitroller.sol (Lines 2352,2358,2365,2372)

```
2352     function getCashPrior() internal view returns (uint);
2353
2354     /**
2355      * @dev Checks whether or not there is sufficient allowance
2356      *       for this contract to move amount from `from` and
2357      *       whether or not `from` has a balance of at least `amount`.
2358      * Does NOT do a transfer.
2359
2360     */
2361     * @dev Performs a transfer in, ideally returning an
2362       explanatory error code upon failure rather than reverting.
2363     * If caller has not called `checkTransferIn`, this may
2364       revert due to insufficient balance or insufficient
2365       allowance.
2366     * If caller has called `checkTransferIn` successfully, this
2367       should not revert in normal conditions.
2368     */
2369     function doTransferIn(address from, uint amount) internal
2370       returns (Error);
2371
2372     /**
2373      * @dev Performs a transfer out, ideally returning an
2374       explanatory error code upon failure tather than reverting.
2375      * If caller has not called checked protocol's balance, may
2376       revert due to insufficient cash held in the contract.
2377      * If caller has checked protocol's balance, and verified it
2378       is >= amount, this should not revert in normal conditions.
2379     */
2380     function doTransferOut(address payable to, uint amount)
2381       internal returns (Error);
```

Risk Level:

Likelihood - 1

Impact - 1

FINDINGS & TECH DETAILS

Recommendation:

Unimplemented functions should either be implemented or deleted out from the contracts.

Remediation Plan:

ACKNOWLEDGED: The issue was acknowledged by the [Planet Finance team](#).

3.18 (HAL-18) REDUNDANT BOOLEAN COMPARISON - INFORMATIONAL

Description:

In the solidity language, Boolean constants can be used directly and do not need to be compared to `true` or `false`. In the `Gammatroller.sol` contract, boolean constants are compared with `true`.

Code Location:

Listing 44: Gammatroller.sol

```
1 marketToJoin.accountMembership[borrower] == true (#3460)
2 require(msg.sender == admin || state == true,only admin can
      unpause) (#4335)
3 require(msg.sender == admin || state == true,only admin can
      unpause) (#4345)
4 require(msg.sender == admin || state == true,only admin can
      unpause) (#4354)
5 require(msg.sender == admin || state == true,only admin can
      unpause) (#4363)
6 require(market.isListed == true,gammamarket is not listed) (#4399)
7 borrowers == true (#4560)
8 suppliers == true (#4567)
9 markets[address(gToken)].collateralFactorMantissa == 0 &&
      borrowGuardianPaused[address(gToken)] == true && gToken.
      reserveFactorMantissa() == 1e18 (#4657-4660)
```

Risk Level:

Likelihood - 1

Impact - 1

FINDINGS & TECH DETAILS

Recommendation:

It is recommended to compare boolean constants directly in the `require` or `if` statement.

Remediation Plan:

ACKNOWLEDGED: The issue was acknowledged by the `Planet Finance` team.

3.19 (HAL-19) UNUSED REDUNDANT CODE - INFORMATIONAL

Description:

During the test, It has been observed that some contract codes are not used. There are a few instances of unused code in the repository, each commented line references types/identifiers, but performs no action with them, so no code will be generated for such statements, and they can be removed. There might be more instances of unused code in the repository.

Code Location:

Listing 45

```
1 minter (Gammatroller.sol#3551)
2 mintAmount (Gammatroller.sol#3552)
3 gToken (Gammatroller.sol#3574)
4 minter (Gammatroller.sol#3575)
5 actualMintAmount (Gammatroller.sol#3576)
6 mintTokens (Gammatroller.sol#3577)
7 gToken (Gammatroller.sol#3636)
8 redeemer (Gammatroller.sol#3637)
9 gToken (Gammatroller.sol#3711)
10 borrower (Gammatroller.sol#3712)
11 borrowAmount (Gammatroller.sol#3713)
12 payer (Gammatroller.sol#3735)
13 borrower (Gammatroller.sol#3736)
14 repayAmount (Gammatroller.sol#3737)
15 gToken (Gammatroller.sol#3765)
16 payer (Gammatroller.sol#3766)
17 borrower (Gammatroller.sol#3767)
18 actualRepayAmount (Gammatroller.sol#3768)
19 borrowerIndex (Gammatroller.sol#3769)
20 liquidator (Gammatroller.sol#3792)
21 gTokenBorrowed (Gammatroller.sol#3839)
22 gTokenCollateral (Gammatroller.sol#3840)
23 liquidator (Gammatroller.sol#3841)
24 borrower (Gammatroller.sol#3842)
25 actualRepayAmount (Gammatroller.sol#3843)
```

```
26 seizeTokens (Gammatroller.sol#3844)
27 seizeTokens (Gammatroller.sol#3870)
28 gTokenCollateral (Gammatroller.sol#3903)
29 gTokenBorrowed (Gammatroller.sol#3904)
30 liquidator (Gammatroller.sol#3905)
31 borrower (Gammatroller.sol#3906)
32 seizeTokens (Gammatroller.sol#3907)
33 gToken (Gammatroller.sol#3951)
34 src (Gammatroller.sol#3952)
35 dst (Gammatroller.sol#3953)
36 transferTokens (Gammatroller.sol#3954)
37 minter (PlanetDiscountDelegate.sol#951)
38 exchangeRate (PlanetDiscountDelegate.sol#952)
```

Risk Level:

Likelihood - 1

Impact - 1

Recommendation:

Unused codes should be deleted from the repository. Consider updating the logic and remove redundant statements.

Remediation Plan:

ACKNOWLEDGED: The issue was acknowledged by the [Planet Finance team](#).

3.20 (HAL-20) USE OF INLINE ASSEMBLY - INFORMATIONAL

Description:

Inline assembly is a way to access the Ethereum Virtual Machine at a low level. This discards several important safety features in Solidity and could incur in some risks should they used incorrectly.

Code Location:

The following contracts make use of inline assembly:

Listing 46

```
1 GAMMA.sol:302: assembly { chainId := chainid() }
2 Gammatroller.sol:304: assembly { chainId := chainid() }
3 Gammatroller.sol:541: assembly {
4 gAQUA_Delegate.sol:3089: assembly {
5 gAQUA_Delegate.sol:3124: assembly {
6 gAQUA_Delegator.sol:1642: assembly {
7 gAQUA_Delegator.sol:1669: assembly {
8 gAQUA_Delegator.sol:1687: assembly {
9 gGAMMA_Delegate.sol:2959: assembly {
10 gGAMMA_Delegate.sol:2994: assembly {
11 gGAMMA_Delegator.sol:964: assembly {
12 gGAMMA_Delegator.sol:991: assembly {
13 gGAMMA_Delegator.sol:1009: assembly {
14 gToken_Delegate.sol:3090: assembly {
15 gToken_Delegate.sol:3125: assembly {
16 gToken_Delegator.sol:1642: assembly {
17 gToken_Delegator.sol:1669: assembly {
18 gToken_Delegator.sol:1687: assembly {
19 PlanetDiscountDelegator.sol:56: assembly {
20 Unitroller.sol:2595: assembly {
```

Risk Level:

Likelihood - 1

Impact - 1

Recommendation:

When possible, it is not recommended to use inline assembly because it allows access to the Ethereum Virtual Machine (EVM) at a low level. An attacker could bypass many important safety features of Solidity.

Remediation Plan:

ACKNOWLEDGED: The issue was acknowledged by the [Planet Finance](#) team.

3.21 (HAL-21) WRONG CODE NOMENCLATURE - INFORMATIONAL

Description:

During the manual review, it was identified that one function functionality did not behave accordingly to its name. The function was named `validateAquaAndGammaPrice` and its functionality allowed to change the token price.

Code Location:

`UniswapAnchoredView` function: `validateAquaAndGammaPrice`

Risk Level:

Likelihood - 1

Impact - 1

Recommendation:

It is recommended to rename the function to ensure better maintainability of the codebase.

Remediation Plan:

ACKNOWLEDGED: The issue was acknowledged by the `Planet Finance` team.

3.22 (HAL-22) POSSIBLE MISUSE OF PUBLIC FUNCTIONS - INFORMATIONAL

Description:

In public functions, array arguments are immediately copied to memory, while external functions can read directly from `calldata`. Reading `calldata` is cheaper than memory allocation. Public functions need to write the arguments to memory because public functions may be called internally. Internal calls are passed internally by pointers to memory. Thus, the function expects its arguments being located in memory when the compiler generates the code for an internal function.

Also, methods do not necessarily have to be public if they are only called within the contract-in such case they should be marked `internal`.

Code Location:

Below are smart contracts and their corresponding functions affected:

Gamma.sol:

```
_become claimGamma delegate delegateBySig enterMarkets getAccountLiquidity  
getAllMarkets getHypotheticalAccountLiquidity getPriorVotes  
_grantGamma initialize _setBorrowPaused _setContributorGammaSpeed  
_setGammaSpeed _setGammatroller _setMintPaused _setPauseGuardian  
_setPriceOracle _setSeizePaused _setTransferPaused
```

Gammatroller.sol:

```
_become claimGamma enterMarkets getAccountLiquidity getAllMarkets getHypotheticalAccountLiquidity  
_grantGamma initialize _setBorrowPaused _setContributorGammaSpeed  
_setGammaSpeed _setGammatroller _setMintPaused _setPauseGuardian  
_setPriceOracle _setSeizePaused _setTransferPaused
```

GToken.sol:

```
borrowBalanceStored exchangeRateStored getTokenConfigByGToken initialize  
reserveFactorMantissa _setBorrowPaused _setGammaSpeed _setInterestRateModel  
_setMintPaused _setWithdrawFeeAddress
```

Maximillion.sol:

```
repayBehalf(address)
```

PlanetDelegatorInterface.sol:

```
_setImplementation(address)
```

PlanetDiscount.sol:

```
changeAddress(address,address,address) deListMarket(address) deListMarkets(address[]) deprecateMarket(address) listMarket(address) listMarkets(address[]) _setAdmin(address)
```

PlanetDiscountDelegate.sol:

```
changeAddress(address,address,address) deListMarket(address) deListMarkets(address[]) listMarket(address) listMarkets(address[])
```

PlanetDiscountDelegator.sol:

```
_setAdmin(address)
```

Reservoir.sol:

```
drip()
```

Risk Level:**Likelihood - 1****Impact - 1****Recommendation:**

Consider, as much as possible, declaring external variables instead of public variables. As for best practice, you should use external if you expect that the function will only be called externally and use public if you need to call the function internally. To sum up, all can access to public functions, external functions only can be accessed externally, and internal functions can only be called within the contract.

FINDINGS & TECH DETAILS

Remediation Plan:

ACKNOWLEDGED: The issue was acknowledged by the Planet Finance team.

AUTOMATED TESTING

4.1 STATIC ANALYSIS REPORT

Description:

Halborn used automated testing techniques to enhance coverage of certain areas of the in-scope contracts. After Halborn verified all the contracts in the repository and was able to compile them correctly into their application binary interface (ABI) and binary formats, the tool Slither was used. Slither is a Solidity static analysis framework. This tool can statically verify mathematical relationships between Solidity variables to detect invalid or inconsistent usage of the contracts' application programming interfaces (APIs) across the entire code-base.

Results:

```
INFO:Detectors:  
PlanetDiscountDelegate (PlanetDiscountDelegate.sol#734-976) contract sets array length with a user-controlled value:  
  - usersWhoHaveBorrow[_market].push(borrower) (PlanetDiscountDelegate.sol#934)  
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#array-length-assignment  
  
INFO:Detectors:  
PlanetStorage.admin (PlanetStorage.sol#17) is never initialized. It is used in:  
  - PlanetDiscountDelegate.changeAddress(address,address,address) (PlanetDiscountDelegate.sol#736-751)  
  - PlanetDiscountDelegate.listMarket(address) (PlanetDiscountDelegate.sol#753-758)  
  - PlanetDiscountDelegate.listMarkets(address[]) (PlanetDiscountDelegate.sol#760-768)  
  - PlanetDiscountDelegate.delistMarket(address) (PlanetDiscountDelegate.sol#770-775)  
  - PlanetDiscountDelegate.delistMarkets(address[]) (PlanetDiscountDelegate.sol#777-785)  
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-state-variables  
  
INFO:Detectors:  
GERc20Delegator.delegateTo(address,bytes) (gToken_Delegator.sol#1640-1648) uses delegatecall to a input-controlled function id  
  - (success,returnData) = callee.delegatecall(data) (gToken_Delegator.sol#1641)  
GERc20Delegator.fallback() (gToken_Delegator.sol#1681-1695) uses delegatecall to a input-controlled function id  
  - (success) = implementation.delegatecall(msg.data) (gToken_Delegator.sol#1685)  
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#controlled-delegatecall  
  
INFO:Detectors:  
Gammatroller (Gammatroller.sol#3337-4676) contract sets array length with a user-controlled value:  
  - allMarkets.push(gToken(gToken)) (Gammatroller.sol#4269)  
Gammatroller (Gammatroller.sol#3337-4676) contract sets array length with a user-controlled value:  
  - accountAssets[borrower].push(gToken) (Gammatroller.sol#3471)  
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#array-length-assignment  
  
INFO:Detectors:  
Unitroller.fallback() (Unitroller.sol#537-549) uses delegatecall to a input-controlled function id  
  - (success) = implementation.delegatecall(msg.data) (Unitroller.sol#539)  
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#controlled-delegatecall  
  
INFO:Detectors:  
Gammatroller.grantGammaInternal(address,uint256) (Gammatroller.sol#4586-4594) ignores return value by gamma.transfer(user,amount) (Gammatroller.sol#4590)  
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unchecked-transfer  
  
INFO:Detectors:  
UnitrollerAdminStorage.implementation (UnitrollerAdminStorage.sol#397) is never initialized. It is used in:  
  - UnitrollerAdminStorage.adminOrInitializing() (UnitrollerAdminStorage.sol#4378-4380)
```

```

UniswapV2OracleLibrary.currentBlockTimestamp() (UniswapLib.sol#38-40) uses a weak PRNG: "uint32(block.timestamp % 2 ** 32) (UniswapLib.sol#39)"
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#weak-PRNG

INFO:Detectors:
Reservoir.drip() (Reservoir.sol#368-403) ignores return value by token_.transfer(gammaTroller,deltaDripGammaTroller_) (Reservoir.sol#396)
Reservoir.drip() (Reservoir.sol#368-403) ignores return value by token_.transfer(foundation,deltaDripFoundation_) (Reservoir.sol#397)
Reservoir.drip() (Reservoir.sol#368-403) ignores return value by token_.transfer(treasury,deltaDripTreasury_) (Reservoir.sol#398)
Reservoir.dripOnFarm(uint256) (Reservoir.sol#406-414) ignores return value by token.transfer(farmAddress,_amount) (Reservoir.sol#410)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unchecked-transfer

INFO:Detectors:
PlanetDiscountDelegator.fallback() (PlanetDiscountDelegator.sol#50-64) uses delegatecall to a input-controlled function id
  - (success) = implementation.delegatecall(msg.data) (PlanetDiscountDelegator.sol#54)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#controlled-delegatecall

INFO:Detectors:
GERc20Delegator.delegateTo(address,bytes) (gGAMMA_Delegator.sol#902-970) uses delegatecall to a input-controlled function id
  - (success,returnData) = callee.delegatecall(data) (gGAMMA_Delegator.sol#963)
GERc20Delegator.fallback() (gGAMMA_Delegator.sol#1003-1017) uses delegatecall to a input-controlled function id
  - (success) = implementation.delegatecall(msg.data) (gGAMMA_Delegator.sol#1007)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#controlled-delegatecall

INFO:Detectors:
Maximillion.repayBehalfExplicit(address,CEther) (Maximillion.sol#2750-2759) sends eth to arbitrary user
  Dangerous calls:
    - cEther_.repayBorrowBehalf.value(borrows)(borrower) (Maximillion.sol#2754)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#functions-that-send-ether-to-arbitrary-destinations

INFO:Detectors:
GERc20Delegator.delegateTo(address,bytes) (gAQUA_Delegator.sol#1640-1648) uses delegatecall to a input-controlled function id
  - (success,returnData) = callee.delegatecall(data) (gAQUA_Delegator.sol#1641)
GERc20Delegator.fallback() (gAQUA_Delegator.sol#1681-1695) uses delegatecall to a input-controlled function id
  - (success) = implementation.delegatecall(msg.data) (gAQUA_Delegator.sol#1685)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#controlled-delegatecall

INFO:Detectors:
UniswapV2OracleLibrary.currentBlockTimestamp() (UniswapLib.sol#38-40) uses a weak PRNG: "uint32(block.timestamp % 2 ** 32) (UniswapLib.sol#39)"
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#weak-PRNG

INFO:Detectors:
PlanetDiscountDelegator.changeBorrowDiscountAddress().vars (PlanetDiscountDelegator.sol#887) is a local variable never initialized
PlanetDiscountDelegator.returnBorrowerStakedAsset(address,address).vars (PlanetDiscountDelegator.sol#790) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables

INFO:Detectors:
EIP20NonStandardInterface (gToken_Delegator.sol#78-79) has incorrect ERC20 function interface EIP20NonStandardInterface.transfer(address,uint256) (gToken_Delegator.sol#754)
EIP20NonStandardInterface (gToken_Delegator.sol#78-79) has incorrect ERC20 function interface EIP20NonStandardInterface.transferFrom(address,address,uint256) (gToken_Delegator.sol#768)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-erc20-interface

INFO:Detectors:
Reentrancy in GERc20Delegator.setImplementation(address,bool,bytes) (gToken_Delegator.sol#1234-1247):
  External calls:
    - delegateToImplementationabi.encodeWithSignature(resignImplementation)) (gToken_Delegator.sol#1238)
      - (success,returnData) = callee.delegatecall(data) (gToken_Delegator.sol#1641)
    State variables written after call(s):
      - _setImplementation(implementation, (gToken_Delegator.sol#1242)
        - implementation = implementation, (gToken_Delegator.sol#1242)
      State variables written after call(s):
        - admin = admin, (gToken_Delegator.sol#1225)
        - admin = admin, (gToken_Delegator.sol#1225)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1

INFO:Detectors:
PlanetDiscountDelegator.returnBorrowerStakedAsset(address,address).vars (gToken_Delegator.sol#655) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variable

INFO:Detectors:
Gammatroller.mintVerify(address,address,uint256,uint256) (Gammatroller.sol#3572-3583) uses a Boolean constant improperly:
  - false (Gammatroller.sol#3580)
Gammatroller.borrowVerify(address,address,uint256) (Gammatroller.sol#3709-3710) uses a Boolean constant improperly:
  - false (Gammatroller.sol#3716)
Gammatroller.borrowBalanceStored(address,address,uint256,uint256) (Gammatroller.sol#3758-3775) uses a Boolean constant improperly:
  - false (Gammatroller.sol#3772)
Gammatroller.liquidateBorrowVerify(address,address,address,uint256,uint256) (Gammatroller.sol#3831-3850) uses a Boolean constant improperly:
  - false (Gammatroller.sol#3847)
Gammatroller.seizeVerify(address,address,address,uint256) (Gammatroller.sol#3896-3913) uses a Boolean constant improperly:
  - Gamma_.writeCheckpoint(address,uint32,int96,uint96) (Gammatroller.sol#268-279) uses a dangerous strict equality:
    - nCheckpoints > 0 && checkpoints[delegatee][nCheckpoints - 1].fromBlock == blockNumber (Gammatroller.sol#271)
GToken.accrueInterest() (Gammatroller.sol#2246-2324) uses a dangerous strict equality:
  - accrualBlockNumberPrior == currentBlockNumber (Gammatroller.sol#2252)
GToken.accrueInterest() (Gammatroller.sol#2246-2324) uses a dangerous strict equality:
  - require(bool,string)mathErr == MathError.NO_ERROR,could not calculate block delta) (Gammatroller.sol#2268)
GToken.balanceOfUnderlying(address) (Gammatroller.sol#2052-2057) uses a dangerous strict equality:
  - require(bool,string)err == MathError.NO_ERROR,balance could not be calculated) (Gammatroller.sol#2055)
GToken.borrowBalanceStored(address) (Gammatroller.sol#2133-2137) uses a dangerous strict equality:
  - require(bool,string)err == MathError.NO_ERROR,borrowBalanceStored: borrowBalanceStoredInternal failed) (Gammatroller.sol#2135)
CarefulMath.divUInt(uint256,uint256) (Gammatroller.sol#1000-1006) uses a dangerous strict equality:
  - b == 0 (Gammatroller.sol#1001)
GToken.exchangeRateStored() (Gammatroller.sol#2190-2194) uses a dangerous strict equality:
  - require(bool,string)err == MathError.NO_ERROR,exchangeRateStored: exchangeRateStoredInternal failed) (Gammatroller.sol#2192)
GToken.exchangeRateStoredInternal() (Gammatroller.sol#2201-2231) uses a dangerous strict equality:
  - _totalSupply == 0 (Gammatroller.sol#2203)
GToken.initialize(GammatrollerInterface,InterestRateModel,uint256,string,string,uint8) (Gammatroller.sol#1887-1918) uses a dangerous strict equality:
  - require(bool,string)err == 0 && borrowIndex == 0,market may only be initialized once) (Gammatroller.sol#1894)
GToken.initialize(GammatrollerInterface,InterestRateModel,uint256,string,string,uint8) (Gammatroller.sol#1887-1918) uses a dangerous strict equality:
  - require(bool,string)err == uint256(Error.NO_ERROR),setting gammatroller failed) (Gammatroller.sol#1902)
GToken.initialize(GammatrollerInterface,InterestRateModel,uint256,string,string,uint8) (Gammatroller.sol#1887-1918) uses a dangerous strict equality:
  - require(bool,string)err == uint256(Error.NO_ERROR),setting interest rate model failed) (Gammatroller.sol#1910)
GToken.liquidateBorrowFresh(address,address,uint256,GTokenInterface) (Gammatroller.sol#2817-2886) uses a dangerous strict equality:
  - require(bool,string)amountSeizeError == uint256(Error.NO_ERROR),LIQUIDATE_GAMMATROLLER_CALCULATE_AMOUNT_SEIZE_FAILED) (Gammatroller.sol#2862)
GToken.liquidateBorrowFresh(address,address,uint256,GTokenInterface) (Gammatroller.sol#2817-2886) uses a dangerous strict equality:
  - require(bool,string)seizeError == uint256(Error.NO_ERROR),token seizure failed) (Gammatroller.sol#2876)
GToken.mintFresh(address,uint256) (Gammatroller.sol#2359-2424) uses a dangerous strict equality:
  - require(bool,string)vars.mathErr == MathError.NO_ERROR,MINT_EXCHANGE_CALCULATION_FAILED) (Gammatroller.sol#2398)
GToken.mintFresh(address,uint256) (Gammatroller.sol#2359-2424) uses a dangerous strict equality:
  - require(bool,string)vars.mathErr == MathError.NO_ERROR,MINT_NEW_TOTAL_SUPPLY_CALCULATION_FAILED) (Gammatroller.sol#2406)
GToken.mintFresh(address,uint256) (Gammatroller.sol#2359-2424) uses a dangerous strict equality:
  - require(bool,string)vars.mathErr == MathError.NO_ERROR,MINT_NEW_ACCOUNT_BALANCE_CALCULATION_FAILED) (Gammatroller.sol#2409)
Exponential.mulExp(ExponentialNoError,Exp,ExponentialNoError,Exp) (Gammatroller.sol#1180-1200) uses a dangerous strict equality:
  - assert(bool)(err2 == MathError.NO_ERROR) (Gammatroller.sol#1197)
CarefulMath.mulInt(uint256,uint256) (Gammatroller.sol#983-995) uses a dangerous strict equality:
  - a == 0 (Gammatroller.sol#984)
ExponentialNoError.mul_(uint256,uint256,string) (Gammatroller.sol#911-918) uses a dangerous strict equality:
  - a == 0 || b == 0 (Gammatroller.sol#912)

```

```

Reentrancy in GToken.liquidateBorrowInternal(address,uint256,gTokenInterface) (Gammatroller.sol#2791-2806):
    External calls:
    - error = gTokenCollateral.accrueInterest() (Gammatroller.sol#2798)
    - liquidateBorrowFresh(msg.sender,borrower,repayAmount,gTokenCollateral) (Gammatroller.sol#2805)
        - allowed = gammatroller.repayBorrowAllowed(address(this),payer,borrower,repayAmount) (Gammatroller.sol#2716)
        - allowed = gammatroller.liquidateBorrowAllowed(address(this),address(gTokenCollateral),liquidator,borrower,repayAmount) (Gammatroller.sol#2819)
        - allowed = gammatroller.seizeAllowed(address(this),seizerToken,liquidator,borrower,seizeTokens) (Gammatroller.sol#2925)
        - seizeError = gTokenCollateral.seize(liquidator,borrower,seizeTokens) (Gammatroller.sol#2872)
    State variables written after the call(s):
    - liquidateBorrowFresh(msg.sender,borrower,repayAmount,gTokenCollateral) (Gammatroller.sol#2805)
        - totalBorrows = vars.totalBorrowsNew (Gammatroller.sol#2771)
    - liquidateBorrowFresh(msg.sender,borrower,repayAmount,gTokenCollateral) (Gammatroller.sol#2805)
        - totalReserves = vars.totalReservesNew (Gammatroller.sol#2968)
    Reentrancy in GToken.redeemFresh(address,uint256,uint256) (gammatroller.sol#2476-2570):
        External calls:
        - allowed = gammatroller.redeemAllowed(address(this),redeemer,vars.redeemTokens) (Gammatroller.sol#2516)
        State variables written after the call(s):
        - totalsupply = vars.totalSupplyNew (Gammatroller.sol#2559)
    Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1

INFO:Detectors:
GToken.repayBorrowFresh(address,address,uint256).vars (Gammatroller.sol#2726) is a local variable never initialized
GToken._addReservesFresh(uint256).actualAddAmount (Gammatroller.sol#3133) is a local variable never initialized
GToken.mintFresh(address,uint256).vars (Gammatroller.sol#2371) is a local variable never initialized
Gammatroller.borrowAllowed(address,address,uint256).err_scope_0 (Gammatroller.sol#3687) is a local variable never initialized
GToken.borrowFresh(address,uint256).vars (Gammatroller.sol#2616) is a local variable never initialized
GToken.seizeInternal(address,address,address,uint256).vars (Gammatroller.sol#2935) is a local variable never initialized
GToken.redeemFresh(address,uint256,uint256).vars (Gammatroller.sol#2479) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables

INFO:Detectors:
Gammatroller._supportMarket(GToken) (Gammatroller.sol#4244-4263) ignores return value by gToken.isGToken() (Gammatroller.sol#4253)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return

INFO:Detectors:
GERc20Delegate._becomeImplementation(bytes) (gAQUA_Delegate.sol#3171-3181) uses a Boolean constant improperly:
    - false (gAQUA_Delegate.sol#3176)
GERc20Delegate._resignImplementation() (gAQUA_Delegate.sol#3186-3193) uses a Boolean constant improperly:
    - false (gAQUA_Delegate.sol#3188)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#misuse-of-a-boolean-constant

INFO:Detectors:
Ether.updateDiscountForAll().exist_scope_1 (Maximillion.sol#4632) is a local variable never initialized
GToken.borrowFresh(address,uint256).vars (Maximillion.sol#1869) is a local variable never initialized
GToken.redeemFresh(address,uint256,uint256).vars (Maximillion.sol#1869) is a local variable never initialized
GToken._addReservesFresh(uint256).vars (Maximillion.sol#1869) is a local variable never initialized
GToken.mintFresh(address,uint256).vars (Maximillion.sol#1552) is a local variable never initialized
GToken.repayBorrowFresh(address,uint256).vars (Maximillion.sol#1993) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables

INFO:Detectors:
EIP20NonStandardInterface (gAQUA_Delegator.sol#728-798) has incorrect ERC20 function interface:EIP20NonStandardInterface.transfer(address,uint256) (gAQUA_Delegator.sol#75a)
EIP20NonStandardInterface (gAQUA_Delegator.sol#728-798) has incorrect ERC20 function interface:EIP20NonStandardInterface.transferFrom(address,address,uint256) (gAQUA_Delegator.sol#768)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-erc20-interface

INFO:Detectors:
Reentrancy in GERc20Delegator.setImplementation(address,bool,bytes) (gAQUA_Delegator.sol#1234-1247):
    External calls:
    - delegateToImplementation(abi.encodeWithSignature("resignImplementation()")) (gAQUA_Delegator.sol#1238)
        (success,returnData) = callee.delegatecall(data) (gAQUA_Delegator.sol#1641)
    State variables written after the call(s):
        implementation = implementation; (gAQUA_Delegator.sol#1242)
    Reentrancy in GERc20Delegator.constructor(address,GammatrollerInterface,InterestRateModel,uint256,string,string,uint8,address,bytes) (gAQUA_Delegator.sol#1198-1226):
        External calls:
        - delegateToImplementation_abi.encodeWithSignature(initialize(address,address,address,uint256,string,string,uint8),underlying_,gammatroller_,interestRateModel_,initialExchangeRateMantissa_,name_,symbol_,decimals_) (gAQUA_Delegator.sol#1212-1219)
            (success,returnData) = callee.delegatecall(data) (gAQUA_Delegator.sol#1642)
        - _setImplementation(implementation,False,becomeImplementationData) (gAQUA_Delegator.sol#1222)
            (success,returnData) = callee.delegatecall(data) (gAQUA_Delegator.sol#1641)
    State variables written after the call(s):
        - admin = admin; (gAQUA_Delegator.sol#1225)
    Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1

INFO:Detectors:
PlanetDiscount.returnBorrowerStakedAssets(address,address).vars (gAQUA_Delegator.sol#655) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables

INFO:Detectors:
BaseJumpRateModelV2.getSupplyRate(uint256,uint256,uint256) (InterestRateModel.sol#322-337) performs a multiplication on the result of a division:
    - rate = rate * (rate.mul((oneMinusInterestRateModel).div(100)).div(100)) * (rate.div(100)) * (rate.div(100)) (InterestRateModel.sol#338)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply

Reentrancy in GToken.borrowFresh(address,uint256) (Maximillion.sol#1852-1925):
    External calls:
    - allowed = gammatroller.borrowAllowed(address(this),borrower,borrowAmount) (Maximillion.sol#1854)
    - changeUserBorrowDiscountInternal(borrower) (Maximillion.sol#1872)
        - (accountBorrows[borrower].principal,accountBorrows[borrower].interestIndex,totalBorrows) = PlanetDiscount(discountLevel).changeUserBorrowDiscount(borrower) (Maximillion.sol#1839)
    State variables written after the call(s):
        - accountBorrows[borrower].principal = vars.accountBorrowsNew (Maximillion.sol#1910)
        - accountBorrows[borrower].interestIndex = borrowIndex (Maximillion.sol#1911)
        - totalBorrows = vars.totalBorrowsNew (Maximillion.sol#1912)
    Reentrancy in GToken.liquidateBorrowInternal(address,uint256,GToken) (Maximillion.sol#2074-2089):
        External calls:
        - error = gTokenCollateral.accrueInterest() (Maximillion.sol#2001)
        - liquidateBorrowFresh(msg.sender,borrower,repayAmount,gTokenCollateral) (Maximillion.sol#2088)
            - allowed = gammatroller.liquidateBorrowAllowed(address(this),address(gTokenCollateral),liquidator,borrower,repayAmount) (Maximillion.sol#2102)
            - PlanetDiscount(discountLevel).changeLastBorrowAmountDiscountGiven(borrower,newBorrow) (Maximillion.sol#1844)
            - allowed = gammatroller.repayBorrowAllowed(address(this),payer,borrower,repayAmount) (Maximillion.sol#1983)
            - (accountBorrows[borrower].principal,accountBorrows[borrower].interestIndex,totalBorrows) = PlanetDiscount(discountLevel).changeUserBorrowDiscount(borrower) (Maximillion.sol#1839)
            - seizeError = gTokenCollateral.seize(liquidator,borrower,seizeTokens) (Maximillion.sol#2150)
            - gammatroller.liquidateBorrowVerify(address(this),address(gTokenCollateral),liquidator,borrower,repayAmount,seizeTokens) (Maximillion.sol#2157)
            - gammatroller.repayBorrowVerify(address(this),payer,borrower,vars.repayAmount,vars.borrowerIndex) (Maximillion.sol#2061)
        State variables written after the call(s):
            - liquidateBorrowFresh(msg.sender,borrower,repayAmount,gTokenCollateral) (Maximillion.sol#2088)
            - accrualBlockNumber = vars.currentBlockNumber (Maximillion.sol#1486)
            - liquidateBorrowFresh(msg.sender,borrower,repayAmount,gTokenCollateral) (Maximillion.sol#2088)
            - liquidateBorrowFresh(msg.sender,borrower,repayAmount,gTokenCollateral) (Maximillion.sol#2088)
            - (accountBorrows[borrower].principal,accountBorrows[borrower].interestIndex,totalBorrows) = PlanetDiscount(discountLevel).changeUserBorrowDiscount(borrower) (Maximillion.sol#1839)
            - totalBorrows = vars.totalBorrowsNew (Maximillion.sol#1848)
            - totalBorrows = vars.totalBorrowsNew (Maximillion.sol#2049)
            - liquidateBorrowFresh(msg.sender,borrower,repayAmount,gTokenCollateral) (Maximillion.sol#2088)
            - totalReserves = vars.totalReservesNew (Maximillion.sol#1489)
    Reentrancy in GToken.redeemFresh(address,uint256,uint256) (Maximillion.sol#1682-1786):
        External calls:
        - changeUserSupplyDiscountInternal(redememer) (Maximillion.sol#1686)
            - (totalsupply,accountTokens[minter]) = PlanetDiscount(discountLevel).changeUserSupplyDiscount(minter) (Maximillion.sol#1530)
        - allowed = gammatroller.redeemAllowed(address(this),redeemer,vars.redeemTokens) (Maximillion.sol#1726)
    State variables written after the call(s):
        - totalsupply = vars.totalSupplyNew (Maximillion.sol#1771)
    Reentrancy in GToken.repayBorrowFresh(address,address,uint256) (Maximillion.sol#1975-2064):
        External calls:
        - changeUserBorrowDiscountInternal(borrower) (Maximillion.sol#1988)
            - (accountBorrows[borrower].principal,accountBorrows[borrower].interestIndex,totalBorrows) = PlanetDiscount(discountLevel).changeUserBorrowDiscount(borrower) (Maximillion.sol#1839)
        - allowed = gammatroller.repayBorrowAllowed(address(this),payer,borrower,repayAmount) (Maximillion.sol#1983)
    State variables written after the call(s):
        - accountBorrows[borrower].principal = vars.accountBorrowsNew (Maximillion.sol#2047)
        - accountBorrows[borrower].interestIndex = borrowIndex (Maximillion.sol#2048)
        - totalBorrows = vars.totalBorrowsNew (Maximillion.sol#2049)

```

```

GToken._addReservesFresh(uint256) callsite[decom] (gToken_Delegate.sol#900) is a local variable never initialized
GToken._redeemFresh(address,uint256) callsite[decom] (gToken_Delegate.sol#2277) is a local variable never initialized
GToken._redeemFresh(address,uint256,uint256).vars (gToken_Delegate.sol#1969) is a local variable never initialized
GToken._minFresh(address,uint256) callsite[decom] (gToken_Delegate.sol#2278) is a local variable never initialized
GToken._maxFresh(address,uint256) callsite[decom] (gToken_Delegate.sol#2258) is a local variable never initialized
GToken._seizeInternal(address,address,address,uint256).vars (gToken_Delegate.sol#2692) is a local variable never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#uninitialized-local-variables

INFODetectors:
GToken.accurseInterest() (gToken_Delegate.sol#1780-1785) ignores return value by gammatroller.oracle().validate(address(this)) (gToken_Delegate.sol#1787)
ERC20.initialize(address,GammatrollerInterface,InterestRateModel,uint256,string,string,uint8) (gToken_Delegate.sol#2988-2991) ignores return value by ERC20Interface(underlying).totalsupply() (gToken_Delegate.sol#2920)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unhandled-return

INFODetectors:
ERC20NonStandardInterface (gAMMA_Delegator.sol#1-76) has incorrect ERC20 function interface:ERC20NonStandardInterface.transfer(address,uint256) (gAMMA_Delegator.sol#84)
ERC20NonStandardInterface (gAMMA_Delegator.sol#1-76) has incorrect ERC20 function interface:ERC20NonStandardInterface.transferFrom(address,address,uint256) (gAMMA_Delegator.sol#54)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-erc20-interface

INFODetectors:
Reentrancy in GERC20Delegator._setImplementation(address,bytes) (gAMMA_Delegator.sol#502-505):
External calls:
- _delegateImplementationabi.encodeWithSignature(_resignImplementation()) (gAMMA_Delegator.sol#886)
  - (success,returnData) = call{gas,delegatecall}(data) (gAMMA_Delegator.sol#1963)
  State variables written after the call(s):
    - implementation = implementation (gAMMA_Delegator.sol#598)
Reentrancy in GERC20Delegator.constructor(address,CommitterInterface,InterestRateModel,uint256,string,string,uint8,address,bytes) (gAMMA_Delegator.sol#540-574):
External calls:
- delegateImplementationabi.encodeWithSignature(initialize(address,address,address,uint256,string,string,uint8),underlying_gammatroller_,interestRateModel_,initialExchangeRateMantissa_,name_,symbol_,decimals_) (gAMMA_Delegator.sol#560-567)
  - (success,returnData) = call{gas,delegatecall}(data) (gAMMA_Delegator.sol#1963)
  - _setImplementationabi.encodeWithSignature(resignImplementation()) (gAMMA_Delegator.sol#570)
  State variables written after the call(s):
    - admin = admin (gAMMA_Delegator.sol#598)
  State variables written after the call(s):
    - admin = admin (gAMMA_Delegator.sol#598)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1

INFODetectors:
ERC20NonStandardInterface (Maxmillion.sol#673-734) has incorrect ERC20 function interface:ERC20NonStandardInterface.transfer(address,uint256) (Maxmillion.sol#690)
ERC20NonStandardInterface (Maxmillion.sol#672-734) has incorrect ERC20 function interface:ERC20NonStandardInterface.transferFrom(address,address,uint256) (Maxmillion.sol#712)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-erc20-interface

Reentrancy in GToken._addReservesInternal(uint256) (gToken_Delegate.sol#2669-2679):
External calls:
- error = accrueInterest() (gToken_Delegate.sol#2670)
  - gammatroller.oracle().validate(address(this)) (gToken_Delegate.sol#1707)
  State variables written after the call(s):
    - (error,None) = _addReservesFresh(addAmount) (gToken_Delegate.sol#2677)
      - totalReserves = totalReservesNew (gToken_Delegate.sol#2717)
    - _reduceReservesFresh(reduceAmount) (gToken_Delegate.sol#2739)
      - totalReserves = totalReservesNew (gToken_Delegate.sol#2787)
  Reentrancy in GToken._reduceReserves(uint256) (gToken_Delegate.sol#2732-2740):
External calls:
- error = accrueInterest() (gToken_Delegate.sol#2733)
  - gammatroller.oracle().validate(address(this)) (gToken_Delegate.sol#1707)
  State variables written after the call(s):
    - _reduceReservesFresh(reduceAmount) (gToken_Delegate.sol#2739)
      - totalReserves = totalReservesNew (gToken_Delegate.sol#2787)
  Reentrancy in GToken._setInterestRateModel(InterestRateModel) (gToken_Delegate.sol#2802-2810):
External calls:
- error = accrueInterest() (gToken_Delegate.sol#2803)
  - gammatroller.oracle().validate(address(this)) (gToken_Delegate.sol#1707)
  State variables written after the call(s):
    - _setInterestRateModelFresh(NewInterestRateModel) (gToken_Delegate.sol#2809)
      - interestRateModel = NewInterestRateModel (gToken_Delegate.sol#2840)
  Reentrancy in GToken._setReserveFactor(uint256) (gToken_Delegate.sol#2625-2633):
External calls:
- error = accrueInterest() (gToken_Delegate.sol#2626)
  - gammatroller.oracle().validate(address(this)) (gToken_Delegate.sol#1707)
  State variables written after the call(s):
    - _setReserveFactorFresh(newReserveFactorMantissa) (gToken_Delegate.sol#2632)
      - reserveFactorMantissa = newReserveFactorMantissa (gToken_Delegate.sol#2657)
  Reentrancy in GToken.borrowFresh(address,uint256) (gToken_Delegate.sol#2132-2206):
External calls:
- allowed = gammatroller.borrowAllowed(address(this),borrower,borrowAmount) (gToken_Delegate.sol#2134)
- changeUserBorrowDiscountInternal(borrower) (gToken_Delegate.sol#2153)
  - accountBorrows[borrower].principal = accountBorrows[borrower].interestIndex,accountBorrows[borrower].totalBorrows = PlanetDiscount(discountLevel).changeUserBorrowDiscount(borrower) (gToken_Delegate.sol#2118)
  State variables written after the call(s):
    - accountBorrows[borrower].principal = vars.accountBorrowsNew (gToken_Delegate.sol#2190)
    - accountBorrows[borrower].interestIndex = borrowIndex (gToken_Delegate.sol#2191)
    - totalBorrows = vars.totalBorrowsNew (gToken_Delegate.sol#2192)
  Reentrancy in GToken.borrowInternal(uint256) (gToken_Delegate.sol#2072-2080):
External calls:
- error = accrueInterest() (gToken_Delegate.sol#2073)
  - gammatroller.oracle().validate(address(this)) (gToken_Delegate.sol#1707)
  - borrowFresh(msg.sender,borrowAmount) (gToken_Delegate.sol#2079)
    - gammatroller.oracle().validate(address(this)) (gToken_Delegate.sol#1707)
    - _borrowFreshInternal() (gToken_Delegate.sol#2079)
  Reentrancy in PlanetDiscountDelegate.changeAddress(address,address,address) (PlanetDiscountDelegate.sol#736):
External calls:
- oracle = gammatroller.oracle().validate(address(this)) (gToken_Delegate.sol#1707)
  - oracle = newOracle (PlanetDiscountDelegate.sol#736)
  State variables written after the call(s):
    - newOracle = newOracle (PlanetDiscountDelegate.sol#736)
  Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation

INFODetectors:
Exponential.divScaledByExpTruncate(uint256,ExponentialNoError.Exp).fraction (gToken_Delegate.sol#198) shadows:
  ExponentialNoError.fraction(uint256,uint256) (gToken_Delegate.sol#193-193) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing

INFODetectors:
PlanetDiscount.changeAddress(address,address,address) (gToken_Delegate.sol#631) lacks a zero-check on :
  - gammatroller.gammaddress = newGammaddress (gToken_Delegate.sol#636)
PlanetDiscount.changeAddress(address,address,address) (gammatroller.gammaddress) lacks a zero-check on :
  - gammatroller.gammaddress = newGammaddress (gToken_Delegate.sol#631)
PlanetDiscount.changeAddress(address,address,address) (newOracle) (gToken_Delegate.sol#633) lacks a zero-check on :
  - oracle = newOracle (gToken_Delegate.sol#638)
GERC20Delegator.constructor(address,GammatrollerInterface,InterestRateModel,uint256,string,string,uint8,address,bytes).admin (gToken_Delegate.sol#1205) lacks a zero-check on :
  - implementation = implementation (gToken_Delegate.sol#1242)
  Event emitted after the call(s):
    - NewImplementation(implementation) (gToken_Delegate.sol#1246)
  Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation

INFODetectors:
Reentrancy in GERC20Delegator._setImplementation(address,bytes) (gToken_Delegate.sol#1234-1247):
External calls:
- _delegateImplementationabi.encodeWithSignature(_resignImplementation()) (gToken_Delegate.sol#1238)
  - (success,returnData) = call{gas,delegatecall}(data) (gToken_Delegate.sol#1643)
  - _delegateImplementationabi.encodeWithSignature(_becomeImplementation(bytes)) (gToken_Delegate.sol#1244)
  State variables written after the call(s):
    - NewImplementation(implementation) (gToken_Delegate.sol#1246)
  Event emitted after the call(s):
    - NewImplementation(implementation) (gToken_Delegate.sol#1246)
  Reentrancy in GERC20Delegator.constructor(address,GammatrollerInterface,InterestRateModel,uint256,string,string,uint8,address,bytes) (gToken_Delegate.sol#1190-1226):
External calls:
- _delegateImplementationabi.encodeWithSignature(initialize(address,address,address,uint256,string,string,uint8),underlying_gammatroller_,interestRateModel_,initialExchangeRateMantissa_,name_,symbol_,decimals_) (gToken_Delegate.sol#1212-1219)
  - (success,returnData) = call{gas,delegatecall}(data) (gToken_Delegate.sol#1222)
  - _setImplementationabi.encodeWithSignature(resignImplementation(bytes)) (gToken_Delegate.sol#1222)
  - (success,returnData) = call{gas,delegatecall}(data) (gToken_Delegate.sol#1643)
  State variables written after the call(s):
    - NewImplementation(implementation) (gToken_Delegate.sol#1246)
    - _setImplementation(implementation,bytes) (gToken_Delegate.sol#1246)
  Event emitted after the call(s):
    - NewImplementation(implementation) (gToken_Delegate.sol#1246)
  Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3

INFODetectors:
Exponential.divScaledByExpTruncate(uint256,ExponentialNoError.Exp).fraction (Gammatroller.sol#1169) shadows:
  - ExponentialNoError.fraction(uint256,uint256) (Gammatroller.sol#1953-1955) (function)
  Gammatroller.updateGammaSupply(index,address).gammaAccrued (Gammatroller.sol#1433) shadows:

```

```

Unitroller._setPendingAdmin(address).newPendingAdmin (Gammatroller.sol#607) lacks a zero-check on :
    - pendingAdmin != _newPendingAdmin (Gammatroller.sol#407)
GToken._setPendingAdmin(address).newPendingAdmin (Gammatroller.sol#2094) lacks a zero-check on :
    - pendingAdmin != _newPendingAdmin (Gammatroller.sol#407)
Gammatroller._setBorrowCapGuardian(address).newBorrowCapGuardian (Gammatroller.sol#420) lacks a zero-check on :
    - borrowCapGuardian != _newBorrowCapGuardian (Gammatroller.sol#406)
Gammatroller._setPauseGuardian(address).newPauseGuardian (Gammatroller.sol#422) lacks a zero-check on :
    - pauseGuardian != _newPauseGuardian (Gammatroller.sol#422)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation

INFO:Dectors:
Gammatroller.getHypotheticalAccountLiquidityInternal(address,token,uint256) (Gammatroller.sol#403-4091) has external calls inside a loop: (err,vars.gTokenBalance,vars.borrowBalance,vars.exchangeRateMantissa) = asset.getAccountSnapshot(account) (Gammatroller.sol#1449)
Gammatroller.getHypotheticalAccountLiquidityInternal(address,token,uint256,uint256) (Gammatroller.sol#403-4091) has external calls inside a loop: vars.oraclePriceMantissa = oracle.getUnderlyingPrice(asset) (Gammatroller.sol#4058)
Gammatroller.claimGains(address[],bool,bool) (Gammatroller.sol#4556) has external calls inside a loop: borrowIndex = Exp(token.borrowIndex()) (Gammatroller.sol#4561)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#pre-declaration-usage-of-local-variables

INFO:Dectors:
variable GToken.getBalance(address,address,uint256).err (Gammatroller.sol#3665) in Gammatroller.borrowAllowed(address,address,uint256) (Gammatroller.sol#3962-3981) potentially used before declaration: (err,shortfall) = getHypotheticalAccountLiquidityInternal(borrower,token)
External calls:
    - borrowCapGuardian.borrowAllowed(address,address,uint256) (Gammatroller.sol#3687)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#pre-declaration-usage-of-local-variables

INFO:Dectors:
Reentrancy in GToken.borrowAllowed(address,uint256) (Gammatroller.sol#2599-2663):
    State variables written after the call(s):
        - accountBorrowers[borrower].principal += vars.accountBorrowNew (Gammatroller.sol#2651)
        - accountBorrowers[borrower].timestamp = borrowIndex (Gammatroller.sol#2652)
        - totalBorrow = vars.totalBorrowNew (Gammatroller.sol#2653)
Reentrancy in GToken.redeemAllowed(address,uint256) (Gammatroller.sol#2359-2424):
    External calls:
        - allowed = gammatroller.redeemAllowed(address(this),minter,minAmount) (Gammatroller.sol#2361)
    State variables written after the call(s):
        - accountBorrowers[borrower].principal -= vars.accountBorrowNew (Gammatroller.sol#2413)
        - totalSupply = vars.totalSupplyNew (Gammatroller.sol#2424)
Reentrancy in GToken.redeemFresh(address,uint256) (Gammatroller.sol#2476-2570):
    External calls:
        - allowed = gammatroller.redeemAllowed(address(this),redeemer,vars.redenTokens) (Gammatroller.sol#2516)
    State variables written after the call(s):
        - accountBorrowers[borrower].principal -= vars.accountBorrowNew (Gammatroller.sol#2568)
Reentrancy in GToken.repayBorrowAllowed(address(this),payer,borrower,repayAmount) (Gammatroller.sol#2716):
    State variables written after the call(s):
        - accountBorrowers[borrower].principal -= vars.accountBorrowNew (Gammatroller.sol#2769)
        - accountBorrowers[payer].principal += vars.accountBorrowNew (Gammatroller.sol#2770)
        - totalSupply = vars.totalSupplyNew (Gammatroller.sol#2771)
        - repayAmount = vars.repayAmountNew (Gammatroller.sol#2772)
ExponentialDecaySupplyOpportunity(uint256,ExponentialDecayOpportunity).fraction (gAQUA_Delegate.sol#664) shadows:
    - ExponentialDecayOpportunity.fraction(uint256,uint256) (gAQUA_Delegate.sol#47-47) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing

INFO:Dectors:
Gerc20.initialize(address,GammatrollerInterface.InterestRateModel,uint256,string,uint8).underlying_ (gAQUA_Delegate.sol#2907) lacks a zero-check on :
    - pendingAdmin != _newPendingAdmin (gAQUA_Delegate.sol#2560)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation

INFO:Dectors:
Variable PlanetDiscount.discountForAll().exist (gAQUA_Delegate.sol#3046) in Gerc20.updateDiscountForAll() (gAQUA_Delegate.sol#3039-3059) potentially used before declaration: (exist) = PlanetDiscount(discountLevel).supplyDiscountSnap(address(this),usersWhoHaveSupply[1...sc.length-1])
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#pre-declaration-usage-of-local-variables

INFO:Dectors:
Reentrancy in GToken.mintFreshAddress(uint256) (gAQUA_Delegate.sol#1839-1904):
    External calls:
        - changeSupplySupplyBisectInternal(minter) (gAQUA_Delegate.sol#1843)
        - (totalSupply,accountTokens[minter]) = PlanetDiscount(discountLevel).changeUserSupplyDiscount(minter) (gAQUA_Delegate.sol#1820)
    State variables written after the call(s):
        - changeSupplySupplyBisectInternal(minter) (gAQUA_Delegate.sol#1843)
        - (totalSupply,accountTokens[minter]) = PlanetDiscount(discountLevel).changeUserSupplyDiscount(minter) (gAQUA_Delegate.sol#1820)
        - accountTokens[minter] = PlanetDiscount(discountLevel).getSupply(minter) (gAQUA_Delegate.sol#1889)
        - changeSupplySupplyBisectInternal(minter) (gAQUA_Delegate.sol#1843)
        - (totalSupply,accountTokens[minter]) = PlanetDiscount(discountLevel).changeUserSupplyDiscount(minter) (gAQUA_Delegate.sol#1820)
        - totalSupply = vars.totalSupplyNew (gAQUA_Delegate.sol#1844)
Reentrancy in GToken.burnInternal(address,address,address,uint256) (gAQUA_Delegate.sol#2479-2539):
    External calls:
        - changeSupplySupplyBisectInternal(redeemer) (gAQUA_Delegate.sol#1843)
        - (totalSupply,accountTokens[minter]) = PlanetDiscount(discountLevel).changeUserSupplyDiscount(minter) (gAQUA_Delegate.sol#1820)
    State variables written after the call(s):
        - accountTokens[borrower] = vars.borrowerTokensNew (gAQUA_Delegate.sol#2526)
        - (totalReserves,vars.totalReservesNew) = PlanetDiscount(discountLevel).getSupply(minter) (gAQUA_Delegate.sol#2527)
        - totalReserves = vars.totalReservesNew (gAQUA_Delegate.sol#2525)
        - totalSupply = vars.totalSupplyNew (gAQUA_Delegate.sol#1844)
Reentrancy in gAQUADelegate.transferTokens(address,address,address,uint256) (gAQUA_Delegate.sol#1388-1448):
    External calls:
        - allowed = gammatroller.transferAllowed(address(this),src,dst,tokens) (gAQUA_Delegate.sol#1398)
    State variables written after the call(s):
        - 

UniswapAnchoredView.validate(address) (UniswapAnchoredView.sol#242-306) uses timestamp for comparisons
    Dangerous comparisons:
        - error2 != Error.NO_ERROR (UniswapAnchoredView.sol#255)
        - error3 != Error.NO_ERROR (UniswapAnchoredView.sol#262)
        - anchorPrice >= 2 ** 248 (UniswapAnchoredView.sol#270)
        - error4 != Error.NO_ERROR (UniswapAnchoredView.sol#282)
        - reportedPrice >= 2 ** 248 (UniswapAnchoredView.sol#289)
UniswapAnchoredView.pokeFailedOverPrice(bytes32) (UniswapAnchoredView.sol#314-326) uses timestamp for comparisons
    Dangerous comparisons:
        - error != Error.NO_ERROR (UniswapAnchoredView.sol#319)
        - require(bool,string)(anchorPrice < 2 ** 248,Anchor price too large) (UniswapAnchoredView.sol#322)
UniswapAnchoredView.calculateAnchorPriceFromEthPrice(UniswapConfig.TokenConfig) (UniswapAnchoredView.sol#333-350) uses timestamp for comparisons
    Dangerous comparisons:
        - error != Error.NO_ERROR (UniswapAnchoredView.sol#336)
        - error != Error.NO_ERROR (UniswapAnchoredView.sol#346)
UniswapAnchoredView.convertReportedPrice(UniswapConfig.TokenConfig,int256) (UniswapAnchoredView.sol#358-375) uses timestamp for comparisons
    Dangerous comparisons:
        - err != Error.ERROR (UniswapAnchoredView.sol#369)
UniswapAnchoredView.isWithinAnchor(uint256,uint256) (UniswapAnchoredView.sol#378-392) uses timestamp for comparisons
    Dangerous comparisons:
        - reporterPrice > 0 (UniswapAnchoredView.sol#379)
        - err != Error.NO_ERROR (UniswapAnchoredView.sol#383)
        - (Error.NO_ERROR,anchorRatio <= upperBoundAnchorRatio && anchorRatio >= lowerBoundAnchorRatio) (UniswapAnchoredView.sol#389)
UniswapAnchoredView.fetchAnchorPrice(bytes32,UniswapConfig.TokenConfig,uint256) (UniswapAnchoredView.sol#418-463) uses timestamp for comparisons
    Dangerous comparisons:
        - require(bool,string)(block.timestamp > oldTimestamp,now must come after before) (UniswapAnchoredView.sol#422)
        - error1 != Error.NO_ERROR (UniswapAnchoredView.sol#431)
        - error2 != Error.NO_ERROR (UniswapAnchoredView.sol#451)
UniswapAnchoredView.pokeWindowValues(UniswapConfig.TokenConfig) (UniswapAnchoredView.sol#469-486) uses timestamp for comparisons
    Dangerous comparisons:
        - timeElapsed >= anchorPeriod (UniswapAnchoredView.sol#477)
UniswapAnchoredView.mul(uint256,uint256) (UniswapAnchoredView.sol#510-519) uses timestamp for comparisons
    Dangerous comparisons:
        - a == 0 (UniswapAnchoredView.sol#511)
        - c / a != b (UniswapAnchoredView.sol#514)
UniswapV2OracleLibrary.currentCumulativePrices(address) (UniswapLib.sol#43-61) uses timestamp for comparisons
    Dangerous comparisons:
        - blockTimestampLast != blockTimestamp (UniswapLib.sol#52)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp

INFO:Dectors:
Reservoir.changeFoundationAddress(address)._newFoundation (Reservoir.sol#416) lacks a zero-check on :
    - foundation = _newFoundation (Reservoir.sol#420)
Reservoir.changeTreasuryAddress(address)._newTreasury (Reservoir.sol#426) lacks a zero-check on :
    - treasury = _newTreasury (Reservoir.sol#430)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation

```

```

INFO:Detectors:
Redundant expression "minter (PlanetDiscountDelegate.sol#951)" inPlanetDiscountDelegate (PlanetDiscountDelegate.sol#734-976)
Redundant expression "exchangeRate (PlanetDiscountDelegate.sol#952)" inPlanetDiscountDelegate (PlanetDiscountDelegate.sol#734-976)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements

INFO:Detectors:
Low level call in GERC20Delegator.delegateTo(address,bytes) (gToken_Delegate.sol#1640-1648):
- (success,returnData) = callee.delegatecall(data) (gToken_Delegate.sol#1641)
Low level call in GERC20Delegator.delegateToViewImplementation(bytes) (gToken_Delegate.sol#1667-1675):
- (success,returnData) = address(this).staticcallabi.encodeWithSignature(delegateToImplementation(bytes),data)) (gToken_Delegate.sol#1668)
Low level call in GERC20Delegator.fallback() (gToken_Delegate.sol#1681-1695):
- (success) = implementation.delegatecall(msg.data) (gToken_Delegate.sol#1685)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

INFO:Detectors:
Redundant expression "this (gToken_Delegate.sol#544)" inContext (gToken_Delegate.sol#538-547)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements

INFO:Detectors:
Gammatroller.addToMarketInternal(GToken,address) (Gammatroller.sol#3452-3476) compares to a boolean constant:
- marketJoin.accountMembership[borrower] == true (Gammatroller.sol#3460)
Gammatroller._setMinPaused(GToken,bool) (Gammatroller.sol#4332-4340) compares to a boolean constant:
- require(bool,string)msg.sender == admin || state == true,only admin can unpause (Gammatroller.sol#4335)
Gammatroller._setBorrowPaused(GToken,bool) (Gammatroller.sol#4342-4350) compares to a boolean constant:
- require(bool,string)msg.sender == admin || state == true,only admin can unpause (Gammatroller.sol#4345)
Gammatroller._setTransferPaused(bool) (Gammatroller.sol#4352-4359) compares to a boolean constant:
- require(bool,string)msg.sender == admin || state == true,only admin can unpause (Gammatroller.sol#4354)
Gammatroller._setSeizePaused(bool) (Gammatroller.sol#4361-4368) compares to a boolean constant:
- require(bool,string)msg.sender == admin || state == true,only admin can unpause (Gammatroller.sol#4363)
Gammatroller.setGammaSpeedInternal(GToken,uint256) (Gammatroller.sol#4389-4420) compares to a boolean constant:
- require(bool,string)market.islisted == true,gammamarket is not listed (Gammatroller.sol#4399)
Gammatroller.claimGammat(address[],GToken[],bool,bool) (Gammatroller.sol#4556-4577) compares to a boolean constant:
- suppliers == true (Gammatroller.sol#4567)
Gammatroller.claimGammat(address[],GToken[],bool,bool) (Gammatroller.sol#4556-4577) compares to a boolean constant:
- borrowers == true (Gammatroller.sol#4568)
Gammatroller.isDeprecated(GToken) (Gammatroller.sol#4656-4662) compares to a boolean constant:
- markets[address(gToken)].collateralFactorMantissa == 0 66 borrowGuardianPaused[address(gToken)] == true && gToken.reserveFactorMantissa() == 1e18 (Gammatroller.sol#4657-4660)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#boolean-equality

INFO:Detectors:
Low level call in Unitroller.fallback() (Unitroller.sol#537-540):
- (success) = implementation.delegatecall(msg.data) (Gammatroller.sol#539)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

GToken (Gammatroller.sol#1877-3320) does not implement functions:

- GToken.doTransferIn(address,uint256) (Gammatroller.sol#3299)
- GToken.doTransferOut(address,uint256) (Gammatroller.sol#3306)
- GToken.getCashPrior() (Gammatroller.sol#3293)


Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unimplemented-functions

INFO:Detectors:
Redundant expression "this (gAQUA_Delegate.sol#820)" inContext (gAQUA_Delegate.sol#814-823)
Redundant expression "data (gAQUA_Delegate.sol#3173)" inGErc20Delegate (gAQUA_Delegate.sol#3161-3194)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements

INFO:Detectors:
Low level call in Unitroller.fallback() (Unitroller.sol#2590-2603):
- (success) = implementation.delegatecall(msg.data) (Unitroller.sol#2592)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

INFO:Detectors:
GToken (Unitroller.sol#817-2373) does not implement functions:

- GToken.checkTransferIn(address,uint256) (Unitroller.sol#2358)
- GToken.doTransferIn(address,uint256) (Unitroller.sol#2365)
- GToken.doTransferOut(address,uint256) (Unitroller.sol#2372)
- GToken.getCashPrior() (Unitroller.sol#2352)


Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unimplemented-functions

INFO:Detectors:
Redundant expression "data (gGAMMA_Delegate.sol#3043)" inGErc20Delegate (gGAMMA_Delegate.sol#3031-3064)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements

INFO:Detectors:
Gamma.slitherConstructorConstantVariables() (GAMMA.sol#8-305) uses literals with too many digits:


- totalSupply = 100000000e18 (GAMMA.sol#19)


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

INFO:Detectors:
Redundant expression "this (Reservoir.sol#22)" inContext (Reservoir.sol#16-25)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements

INFO:Detectors:
Reservoir.slitherConstructorConstantVariables() (Reservoir.sol#262-562) uses literals with too many digits:


- maxGammaDrippedPerDay = 100000 (Reservoir.sol#303)


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

INFO:Detectors:
Low level call in PlanetDiscountDelegator.fallback() (PlanetDiscountDelegator.sol#50-64):


- (success) = implementation.delegatecall(msg.data) (PlanetDiscountDelegator.sol#54)


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

```

According to the test results, some findings found by these tools were considered as false positives, while some of these findings were real security concerns. All relevant findings were reviewed by the auditors and addressed in the report as security concerns.

4.2 AUTOMATED SECURITY SCAN

Description:

Halborn used automated security scanners to assist with detection of well-known security issues, and to identify low-hanging fruits on the target contracts. Among the tools used was MythX, a security analysis service for Ethereum smart contracts. MythX performed a scan on the testers machine and sent the compiled results to the analyzers to locate any vulnerabilities. Only security-related findings are shown below.

Results:

Report for PlanetDiscountDelegate.sol
<https://dashboard.mythx.io/#/console/analyses/53207a63-cce7-4ec4-bfc5-121024beb9c1>

Line	SWC Title	Severity	Short Description
1	(SWC-103) Floating Pragma	Low	A floating pragma is set.
651	(SWC-123) Requirement Violation	Low	Requirement violation.
712	(SWC-123) Requirement Violation	Low	Requirement violation.
715	(SWC-113) DoS with Failed Call	Low	Multiple calls are executed in the same transaction.

Report for PlanetDiscountStorage.sol
<https://dashboard.mythx.io/#/console/analyses/a956871e-fff2-4e7b-9221-7104dbe13a50>

Line	SWC Title	Severity	Short Description
1	(SWC-103) Floating Pragma	Low	A floating pragma is set.

Report for Reservoir.sol
<https://dashboard.mythx.io/#/console/analyses/c861a5e1-7850-452d-9484-8ee14a86e708>

Line	SWC Title	Severity	Short Description
7	(SWC-103) Floating Pragma	Low	A floating pragma is set.
341	(SWC-120) Weak Sources of Randomness from Chain Attributes	Low	Potential use of "block.number" as source of randomness.
372	(SWC-120) Weak Sources of Randomness from Chain Attributes	Low	Potential use of "block.number" as source of randomness.

Report for PlanetDiscountDelegator.sol
<https://dashboard.mythx.io/#/console/analyses/f54bfed0-84a3-4dae-acac-b816b1c2cd53>

Line	SWC Title	Severity	Short Description
1	(SWC-103) Floating Pragma	Low	A floating pragma is set.
37	(SWC-112) Delegatecall to Untrusted Callee	High	The contract delegates execution to another contract with a user-supplied address.

Report for GAMMA.sol

Line	SWC Title	Severity	Short Description
194	(SWC-120) Weak Sources of Randomness from Chain Attributes	Low	Potential use of "block.number" as source of randomness.
267	(SWC-120) Weak Sources of Randomness from Chain Attributes	Low	Potential use of "block.number" as source of randomness.

Report for PlanetDiscountDelegate.sol

Line	SWC Title	Severity	Short Description
651	(SWC-123) Requirement Violation	Low	Requirement violation.
712	(SWC-123) Requirement Violation	Low	Requirement violation.
715	(SWC-113) DoS with Failed Call	Low	Multiple calls are executed in the same transaction.

Report for Maximillion.sol

Line	SWC Title	Severity	Short Description
1237	(SWC-120) Weak Sources of Randomness from Chain Attributes	Low	Potential use of "block.number" as source of randomness.

Report for PlanetDiscountDelegator.sol

Line	SWC Title	Severity	Short Description
37	(SWC-112) Delegatecall to Untrusted Callee	High	The contract delegates execution to another contract with a user-supplied address.

Report for Unitroller.sol

Line	SWC Title	Severity	Short Description
1204	(SWC-120) Weak Sources of Randomness from Chain Attributes	Low	Potential use of "block.number" as source of randomness.

Report for priceOracle.sol

Line	SWC Title	Severity	Short Description
1062	(SWC-120) Weak Sources of Randomness from Chain Attributes	Low	Potential use of "block.number" as source of randomness.
2287	(SWC-123) Requirement Violation	Low	Requirement violation.
2434	(SWC-123) Requirement Violation	Low	Requirement violation.

Report for gErc20_delegate.sol

Line	SWC Title	Severity	Short Description
1469	(SWC-120) Weak Sources of Randomness from Chain Attributes	Low	Potential use of "block.number" as source of randomness.

Report for Gammatroller.sol

Line	SWC Title	Severity	Short Description
196	(SWC-120) Weak Sources of Randomness from Chain Attributes	Low	Potential use of "block.number" as source of randomness.
269	(SWC-120) Weak Sources of Randomness from Chain Attributes	Low	Potential use of "block.number" as source of randomness.
2090	(SWC-120) Weak Sources of Randomness from Chain Attributes	Low	Potential use of "block.number" as source of randomness.
4665	(SWC-120) Weak Sources of Randomness from Chain Attributes	Low	Potential use of "block.number" as source of randomness.

Report for gErc20Delegate_Discount.sol

Line	SWC Title	Severity	Short Description
1543	(SWC-120) Weak Sources of Randomness from Chain Attributes	Low	Potential use of "block.number" as source of randomness.

Report for gEther_Discount.sol

Line	SWC Title	Severity	Short Description
1237	(SWC-120) Weak Sources of Randomness from Chain Attributes	Low	Potential use of "block.number" as source of randomness.

Report for Reservoir.sol

Line	SWC Title	Severity	Short Description
341	(SWC-120) Weak Sources of Randomness from Chain Attributes	Low	Potential use of "block.number" as source of randomness.
372	(SWC-120) Weak Sources of Randomness from Chain Attributes	Low	Potential use of "block.number" as source of randomness.

All relevant valid findings were founded in the manual code review.

THANK YOU FOR CHOOSING
 HALBORN