# Multiplier.Finance MCL Code Review

## Multiplier.Finance

25 May 2021
Version: 1.0

Presented by:
Kudelski Security Research Team
Kudelski Security – Nagravision SA

Corporate Headquarters
Kudelski Security – Nagravision SA
Route de Genève, 22-24
1033 Cheseaux sur Lausanne
Switzerland

## DOCUMENT PROPERTIES

| | |
|---|---|
| Version: | 1.0 |
| File Name: | 0525 Findings_Multiplier_Finance_Final_Report |
| Publication Date: | 25 May 2021 |
| Confidentiality Level: | For Public Disclosure |
| Document Owner: | Scott Carlson |
| Document Recipient: | Multiplier Finance |
| Document Status: | Approved |

# TABLE OF CONTENTS

# TABLE OF FIGURES

# 1. EXECUTIVE SUMMARY

Kudelski Security (“Kudelski”), the cybersecurity division of the Kudelski Group, was engaged by Multiplier.Finance. (“Multipler.Finance” or “Client”) to conduct an external security assessment in the form of a Code Review of the Multi Chain Lending application.

The assessment was conducted remotely by the Kudelski Security team from our secure lab environment. The tests took place from Feb 20, 2021 to Apr 7, 2021 and focused on the following objectives:

1. To help the Client better understand the security posture of the application.
2. To provide a professional opinion on the maturity, adequacy, and efficiency of the security measures that are in place.
3. To identify potential issues and include improvement recommendations.

This report summarizes the tests performed and findings in terms of strengths and weaknesses. It also contains detailed descriptions of the discovered vulnerabilities, steps the Kudelski Security team took to exploit each vulnerability, and recommendations for remediation.

As a separate document, we have provided the full threat model conducted against each component within the system and have described all possible threat scenarios.

## 1.1 Engagement Limitations

The engagement was cut into two parts where the first engagement, timeboxed to four days, verified that the Smart Contracts to be deployed were production ready.

**Contracts Reviewed (BSC) During Phase One**

| FILE | STATUS |
|---|---|
| src/bsc/bMxx.sol | OK |
| src/bsc/DeedToken.sol | OK |
| src/bsc/Migrations.sol | OK |
| src/bsc/MxxConverterRedemption.sol | OK |

**Contracts Reviewed (ETH) During Phase One**

| FILE | STATUS |
|---|---|
| src/eth/Migrations.sol | OK |
| src/eth/MockupMXX.sol | OK |
| src/eth/MxxConverter.sol | OK |

The second engagement, also timeboxed, was completed much later and was based on the last version of the MCL application.

The architecture and code review are based on the documentation and code provided by Multiplier.Finance. The code resides in a repository at https://github.com/Multiplier-Finance/MCL-SmartContracts.

The reviews are based on the commit hash:

MCL-SmartContracts: cff17d6e07b51e7468a4aba72ae83b309b98d561

All third-party libraries were deemed out-of-scope for this review and are expected to work as designed. Based on the criticality of the dependency, we looked at the current state of the third-party libraries included when necessary.

## 1.2 Engagement Analysis

This engagement was comprised of a code review including reviewing how the architecture has been implemented as well as any security issues. The architecture implementation review was based on the documentation and the information retrieved through communication between the Multiplier.Finance team and the Kudelski Security team. The implementation review concluded that the application implementation is as good as expected.

The code review was conducted by the Kudelski Security team on the code provided by Multiplier.Finance, in the form of a Github repository. The code review focused on the handling of secure and private information handling in the code of the Smart Contracts. As a result of our work, we identified **0 High**, **0 Medium**, **3 Low**, and **15 Informational** findings.

The findings referred to in the Findings section are such as they would improve the functionality and performance of the application and secure it further.

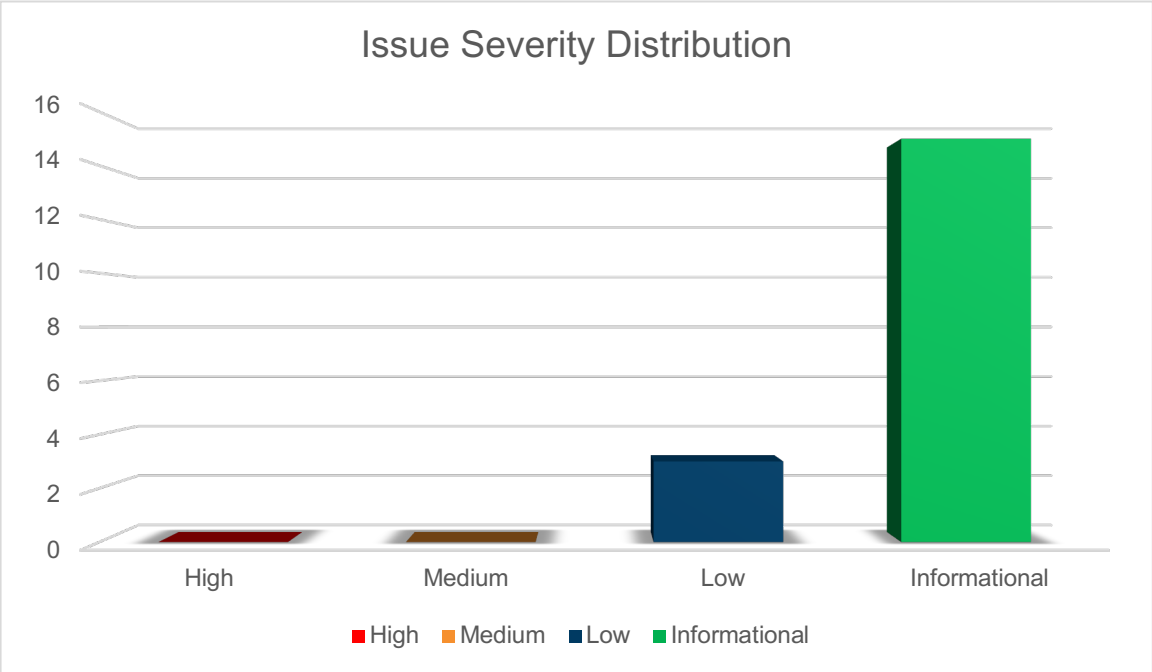

Figure 1 Issue Severity Distribution

## 1.3 Observations

The code is generally well written and for the most part documented. This facilitates reading of the execution flow.

The engagement concluded that the code is fit for the purpose for which it has been designed. All issues raised have been remedied during the re-review process.

## 1.4 Issue Summary List

| ID | SEVERITY | RESOLUTION | FINDING |
|---|---|---|---|
| KS-MCL-F-01 | Low | Resolved | Outdated versions of Solidity in use, with dependencies |
| KS-MCL-F-02 | Low | Resolved | High setting allowed that fee could be hiked and hurt the lenders |
| KS-MCL-F-03 | Low | Resolved | Arithmetic calculation of token value |
| KS-MCL-F-04 | Informational | Resolved | Difference in language modifiers |
| KS-MCL-F-05 | Informational | Resolved | Using a cool-down approach to resolve timing issues on nodes |
| KS-MCL-F-06 | Informational | Resolved | Addresses used need to be verified |
| KS-MCL-F-07 | Informational | Resolved | Constants need to be verified |
| KS-MCL-F-08 | Informational | Resolved | Tests do not compile according to documentation |
| KS-MCL-F-09 | Informational | Resolved | Out-commented code |
| KS-MCL-F-10 | Informational | Resolved | Unnecessary gas consumption |
| KS-MCL-F-11 | Informational | Resolved | Redundant fallback function |
| KS-MCL-F-12 | Informational | Resolved | Hardcoded WAD values |
| KS-MCL-F-13 | Informational | Resolved | Hardcoded RAY values |
| KS-MCL-F-14 | Informational | Resolved | Hardcoded RAY values |
| KS-MCL-F-15 | Informational | Resolved | Incorrect comment |
| KS-MCL-F-16 | Informational | Resolved | Unnecessary gas consumption |

| ID | SEVERITY | RESOLUTION | FINDING |
|---|---|---|---|
| KS-MCL-F-17 | Informational | Resolved | Unnecessary nesting of conditionals |
| KS-MCL-F-18 | Informational | Resolved | User allowed to borrow too much for a stable asset |

# 2. METHODOLOGY

Kudelski Security uses the following high-level methodology when approaching engagements. They are broken up into the following phases.


Figure 2 Methodology Flow

## 2.1 Kickoff

A kickoff meeting was conducted where project stakeholders were gathered to discuss the project and the responsibilities of the participants. During the kickoff meeting, the scope and activities were verified. The kickoff meeting validated and ensured communication channels were in place so that when the engagement began the following information was understood by all participants and stakeholders.

- Designated points of contact
- Communication methods and frequency
- Shared documentation
- Code and/or any other artifacts necessary for project success
- Follow-up meeting schedule, such as a technical walkthrough
- Understanding of timeline and duration

## 2.2 Ramp-up

Ramp-up consisted of the activities necessary to gain proficiency on the project. This included the steps needed for familiarity with the codebase and technological innovation utilized. This included, but is not limited to:

- Reviewing previous work in the area including academic papers
- Reviewing programming language constructs for specific languages
- Researching common flaws and recent technological advancements

## 2.3 Review

The review phase is where the majority of the work on the engagement was completed. We analyzed the project for flaws and issues that could impact the security posture.

In this code audit, we performed the following tasks:

1. Security analysis and architecture review
2. Review of the code written for the project
3. Compliance of the code with the provided technical documentation

The review for this project was performed using manual methods, utilizing the experience of the reviewer. No dynamic testing was performed. Only the use of custom-built scripts and tools were used to assist the reviewer during the testing. Our methodology is discussed in more detail in the following sections.

### Code Safety

We analyzed the provided code, checking for issues related to the following categories:

- General code safety and susceptibility to known issues
- Poor coding practices and unsafe behavior
- Leakage of secrets or other sensitive data through memory mismanagement
- Susceptibility to misuse and system errors
- Error management and logging

This is a general list and not comprehensive. It is meant only to give an understanding of the issues for which we were looking.

### Technical Specification Matching

We analyzed the provided documentation and checked that the code matches the specification. We checked for items such as:

- Proper implementation of the documented protocol phases
- Proper error handling
- Adherence to the protocol logical description

## 2.4 Reporting

Kudelski Security delivered a preliminary report in PDF format that contained an executive summary, technical details, and observations about the project.

The executive summary contained an overview of the engagement including the number of findings as well as a statement about our general risk assessment of the project. We may conclude that the overall risk is low, but depending on what was assessed we may conclude that more scrutiny of the project is needed.

We not only reported security issues identified but also informational findings for improvement categorized into several buckets:

- High
- Medium
- Low
- Informational

The technical details are aimed more at developers, describing the issues, the severity ranking and recommendations for mitigation.

As we performed the audit, we identified issues that are not security related, but are general best practices and steps, that can be taken to lower the attack surface of the project. We called those out as we encounter them and as time permitted.

As an optional step, we can agree on the creation of a public report that can be shared and distributed with a larger audience.

## 2.5 Verify

After the preliminary findings were delivered (in the form of the approved communication channel or delivery of the draft report) we verified any fixes within the specified timeframe agreed upon as part of the project. After the fixes have been verified, we changed the status of the finding in the report from open to remediated.

The output of this phase was a final report with any mitigated findings noted.

## 2.6 Additional Note

It is important to note that, although we do our best in our analysis, no code audit or assessment is a guarantee of the absence of flaws. Our effort was constrained by resource and time limits along with the scope of the agreement.

While assessing the severity of the findings, we considered the impact, ease of exploitability, and the probability of attack. These is a solid baseline for severity determination. Information about the severity ratings can be found in **Appendix B** of this document.

# 3. TECHNICAL DETAILS

## 3.1 Outdated versions of Solidity in use

Finding ID: Finding ID: KS-ML-F-01

Severity: [**Low**]

Status: [**Remediated**]

**Description**

The contracts used old versions of Solidity: down to ^0.4.6 and up to 0.8.0. As a result, functionality available in the latest versions of Solidity will not be used. This would create a less secure environment due to a lack of more robust language constructions.

**Proof of Issue**

**Filename:** lending/contracts/lendingpool.sol

**Beginning Line Number:** 1

```
pragma Solidity ^0.5.0;
```

**Severity and Impact Summary**

Bugfixes and new language features were not included in the contracts used. This has been concluded to be part of a dependency of the AAVE protocols use of old OpenZeppelin contracts. Should the Solidity version be higher the contracts would break.

**Recommendation**

It is easy for projects to lose track of bug fixes and third-party dependencies when using “outdated” versions of core libraries.

Due to the up-stream dependency of this version of solidity by Aave and OpenZeppelin, this finding has been accepted by the project as a requirement.

**References**

- N/A

## 3.2 Fee could be hiked and hurt the lenders

Finding ID: Finding ID: KS-ML-F-02

Severity: [**Low**]

Status: [**Remediated**]

**Description**

The variable feePcnt was defined as 1000000 which is a noticeably high value.

The owner can change the feePcnt through the setFee function, but there is no limit to what to set which could lead to misuse.

**Proof of Issue**

**Filename:**

**Beginning Line Number:** 109

```
/**
     * @dev - The fee percent for each conversion. Default is 1_000_000 (ie 1%). 1e8 is 100%
     */
    uint256 public feePcnt = 1_000_000;
```

**Severity and Impact Summary**

The misuse of parameters and an understanding of compensating controls to stop changing of these values inappropriately could create an unsound interest rate.

The setFee function is defined as onlyOwner(), but there are no limits to the amount the fee can set which can lead to misuse by the owner.

**Recommendation**

Create an acceptable upper limit on the interest rates so that a malicious owner of a contract could not drain accounts.

Tying the interest rate to an oracle or adding additional sanity checks may provide additional safety.

**References**

- N/A

## 3.3 Arithmetic calculation of token value leads to rounding errors

Finding ID: KS-MCL-F-03

Severity: [**Low**]

Status: [**Remediated**]

**Description**

Since Solidity does not handle floating point numbers, and the code is working with integers, multiplication after division should be avoided as a best practice to limit possible rounding errors.

**Proof of Issue**

**Filename:** lending/contracts/lendingpool/LendingPoolLiquidationManager.sol

**Beginning Line Number:** 378, 385

```
vars.maxAmountCollateralToLiquidate = vars
   .principalCurrencyPrice
   .mul(_purchaseAmount)
   .mul(10**vars.collateralDecimals)
   .div(vars.collateralPrice.mul(10**vars.principalDecimals))
   .mul(vars.liquidationBonus)
   .div(100);
```

**Severity and Impact Summary**

A rounding error can have drastic consequences if the assigned variable is used in anything remotely important, such as economic decisions.

There is no impact on the project with this finding as the token length is 18 digits and this rounding error could impact lengths of 24 digits, thus is informational only for thoroughness.

**Recommendation**

Do not perform multiplication after division.

**References**

- https://blog.sigmaprime.io/Solidity-security.html#SP-15
- https://medium.com/@Soliditydeveloper.com/Solidity-design-patterns-multiply-before-dividing-407980646f7

## 3.4 Difference in language modifiers

Finding ID: KS-ML-F-04

Severity: [**Informational**]

Status: [**Remediated**]

**Description**

The Solidity language on both sides reflects the same usage, but several modifiers are defined in BSC, which are not needed in ETH.

**Proof of Issue**

**Filename:** converter/bsc/contracts/Migrations.sol

**Beginning Line Number:** 4

```
contract Migrations {
  address public owner = msg.sender;
  uint public last_completed_migration;

  modifier restricted() {
    require(
      msg.sender == owner,
```

**Severity and Impact Summary**

When maintaining the contracts, confusion about what modifiers are needed on which side may be an issue.

**Recommendation**

This should not pose a high risk but should be monitored for consistency.

**References**

- N/A

## 3.5 Using a cool-down approach to resolve timing issues on nodes

Finding ID: KS-ML-F-05

Severity: [**Informational**]

Status: [**Remediated**]

**Description**

A constant COOL_DOWN is defined as 15 minutes to reduce the possibility to game the conversion if servers are compromised, which will ensure good consistency. Future verification must review “low and slow” methods to impact the system from this vector for manipulation.

**Proof of Issue**

**Filename:** MxxConverterRedemption.sol

**Beginning Line Number:** 103

```
/**
   * @dev - A cool-down period of 15 minutes.
   */
  uint256 public constant COOL_DOWN = 15 minutes;
```

**Severity and Impact Summary**

Check for the cool-down period to have passed, keeping in mind every check during the cool-down period consumes gas. This is as good as you can get in Solidity unless you want to create an Oracle or any other off-chain solution to serve the cool-down period's timing.

**Recommendation**

May introduce re-deployment needs if these variables ever need to be changed, contract must be updated and re-deployed.

**References**

- Check Ethereum Alarm clock for an example (https://www.ethereum-alarm-clock.com/)

## 3.6 Addresses used need to be verified

Finding ID: Finding ID: KS-ML-F-06

Severity: [**Informational**]

Status: [**Remediated**]

**Description**

Public MXX_ADDRESS is as 0x8F1E37Afacdc033dC2c0BF2BD116eaa558eFCdE8

Public BURN_ADDRESS is as 0x1f31541E77cAC6BFAE2FBdA66a2085e6C137871e

**Proof of Issue**

**Filename:** converter/eth/contracts/MxxConverter.sol

**Beginning Line Number:** 119

```
constructor(address mxx, address burnAddress) public Ownable() {
      availableMxxAmt = MAX_CONVERTABLE;
      MXX_ADDRESS = mxx;
      BURN_ADDRESS = burnAddress;
  }
```

**Severity and Impact Summary**

We see no transactions recorded at this time at MXX_ADDRESS.

We see considerable activity happening at BURN_ADDRESS.

**Recommendation**

These addresses have been verified as the public addresses accessible for everyone looking at the deployed smart contracts.

**References**

- N/A

## 3.7 Constants need to be verified

Finding ID: Finding ID: KS-ML-F-07

Severity: [**Informational**]

Status: [**Remediated**]

**Description**

The Constant MAX_CONVERTABLE is defined as 415_000_000e8 (41,500,000,000,000,000) to put a limit on conversions

**Proof of Issue**

**Filename:** converter/eth/contracts/MxxConverter.sol

**Beginning Line Number:** 83

```
/**
 * @dev - The grand total number of Mxx that can be converted to BSC.
 */
uint256 internal constant MAX_CONVERTABLE = 415_000_000e8;
```

**Severity and Impact Summary**

The defined value differs considerably from MAX_SUPPLY on the BSC side, which is 4_150_000e18 (1E9), but we do not have enough information to determine if this is a risk due to its size or a simple math miscalculation.

**Recommendation**

This has been verified to be correct.

**References**

- N/A

## 3.8 Tests do not compile according to documentation

Finding ID: KS-MCL-F-08

Severity: [**Informational**]

Status: [**Remediated**]

**Description**

Documentation is provided to run the tests for *lending*; however, following the instructions leads to compilation issues.

**Proof of Issue**

**Filename:** -

**Beginning Line Number:** -

```
$ cd lending/
$ yarn
$ npx hardhat compile
An unexpected error has occurred.

Helpers/contracts-helpers.ts - error TS2307: Cannot find module '../types/MintableERC20'
```

**Severity and Impact Summary**

Tests cannot be executed. Coverage cannot be determined. Edge cases in the code to be audited cannot be explored.

Changes in the code could lead to unintended errors elsewhere in the contracts.

**Recommendation**

Starting from a newly installed machine with access to the repository provide clear, step by step instructions on how to setup an environment where the tests can be executed.

**References**

- N/A

## 3.9 Out-commented code

Finding ID: KS-MCL-F-09

Severity: [**Informational**]

Status: [**Remediated**]

**Description**

Code that is commented out adds confusion, since it is unknown if the code is forgotten to be included, or if it should in fact be removed.

**Proof of Issue**

**Filename:**

lending/contracts/lendingpool/LendingPool.sol,

lending/contracts/fees/FeeProvider.sol

**Beginning Line Number:** 674, 23

```
// vars.totalLiquidity = core.getReserveTotalLiquidity(_reserve);


// uint256 private safetyModuleRewardRate; // The remaining of 1 -
supplierRewardRate - governanceRewardRate
```

**Severity and Impact Summary**

N/A

**Recommendation**

Remove the line if it should be removed.

**References**

- N/A

## 3.10 Unnecessary gas consumption

Finding ID: KS-MCL-F-10

Severity: [**Informational**]

Status: [**Remediated**]

**Description**

Re-calculating .length on a variable in a loop is gas-expensive and should be avoided if possible.

**Proof of Issue**

Example of finding. Not every occurrence is mentioned.

**Filename:** lending/contracts/misc/ChainlinkProxyProvider.sol

**Beginning Line Number:** 113

```
uint256[] memory prices = new uint256[](_assets.length);
for (uint256 i = 0; i < _assets.length; i++) {
    prices[i] = getAssetPrice(_assets[i]);
}
```

**Severity and Impact Summary**

N/A

**Recommendation**

Store the length of the iterable in a local variable.

**References**

- N/A

## 3.11 Redundant fallback function

Finding ID: KS-MCL-F-11

Severity: [**Informational**]

Status: [**Remediated**]

**Description**

Starting from Solidity 0.4.0, contracts without a fallback function automatically revert payments, making the code redundant.

**Proof of Issue**

**Filename:** lending/contracts/misc/WallerBallanceProvider.sol

**Beginning Line Number:** 29

```
/**
@dev Fallback function, don't accept any ETH
**/
function() external payable {
        revert("WalletBalanceProvider does not accept payments");
`
```

**Severity and Impact Summary**

N/A

**Recommendation**

As the intention is met, there is no remediation recommendation as the project has chosen to not remove the fallback function as it is included for clarity.

**References**

- N/A

## 3.12 Hardcoded WAD values

Finding ID: KS-MCL-F-12

Severity: [**Informational**]

Status: [**Remediated**]

**Description**

Hardcoded WAD values (18 digit precision) are used to declare certain constants and during certain calculations. If the WAD precision needs to be modified, all these values need to be modified as well.

**Proof of Issue**

Example of finding. Not every occurrence is mentioned.

**Filename:** lending/contracts/fees/FeeProvider.sol

**Beginning Line Number:** 17

```
// percentage of the fee to be calculated on the loan amount
uint256 private constant originationFeeRate = 0.00001 * 1e18; // 100% = 1e18;
uint256 private constant flashloanFeeRate = 0.0006 * 1e18;


// 70/20/10% reward rates //
uint256 private constant supplierRewardRate = 0.7 * 1e18;
uint256 private constant governanceRewardRate = 0.2 * 1e18;
// uint256 private safetyModuleRewardRate; // The remaining of 1 - supplierRewardRate -
governanceRewardRate
```

**Severity and Impact Summary**

N/A

**Recommendation**

For clarity and for easier modifying, replace occurrences of 1e18 with a constant, preferably one that is defined in WadRayMath.sol.

**References**

- N/A

## 3.13 Hardcoded RAY values

Finding ID: KS-MCL-F-13

Severity: [**Informational**]

Status: [**Remediated**]

**Description**

Hardcoded RAY values (1e27) are used to declare certain constants and during certain calculations. If the RAY precision needs to be modified, all these values need to be modified as well.

**Proof of Issue**

**Filename:** lending/contracts/configuration/LendingPoolParametersProvider.sol

**Beginning Line Number:** 15

```
uint256 private constant REBALANCE_DOWN_RATE_DELTA = (1e27)/5;
```

**Severity and Impact Summary**

N/A

**Recommendation**

For clarity and for easier modifying, replace occurrences of 1e27 literals with a constant, preferably one that is defined in WadRayMath.sol.

**References**

- N/A

## 3.14 Hardcoded RAY values

Finding ID: KS-MCL-F-14

Severity: [**Informational**]

Status: [**Remediated**]

**Description**

Hardcoded RAY values (1e27) are used to declare certain constants and during certain calculations. If the RAY precision needs to be modified, all these values need to be modified as well.

**Proof of Issue**

**Filename:** lending/contracts/lendingpool/DefaultReserveInterestRateStrategy.sol

**Beginning Line Number:** 78

```
excessUtilizationRate = uint256(1e27).sub(_optimalUtilizationRate);
```

**Severity and Impact Summary**

N/A

**Recommendation**

For clarity and for easier modifying, replace occurrences of 1e27 literals with a constant, preferably one that is defined in WadRayMath.sol.

**References**

- N/A

## 3.15 Incorrect comment

Finding ID: KS-MCL-F-15

Severity: [**Informational**]

Status: [**Remediated**]

**Description**

Comment is no longer correct as the value is no longer a constant.

**Proof of Issue**

**Filename:** lending/contracts/lendingpool/DefaultReserveInterestRateStrategy.sol

**Beginning Line Number:** 31

```
/**
 * @dev this value represents the excess utilization rate above the
 * optimal. It's always equal to
 * 1-optimal utilization rate. Added as a constant here for gas optimizations
 * expressed in ray (e27)
 **/
uint256 public  excessUtilizationRate;
```

**Severity and Impact Summary**

N/A

**Recommendation**

Remove the following sentence from the comment: *Added as a constant here for gas optimizations*.

Or, if possible, make the value a constant if used as such.

**References**

- N/A

## 3.16 Unnecessary gas consumption

Finding ID: KS-MCL-F-16

Severity: [**Informational**]

Status: [**Remediated**]

**Description**

Performing a SafeMath subtraction is not necessary if there is no risk of underflow. In this case, it is verified by the if-statement that *utilizationRate* is larger than *optimalUtilizationRate*, making the subtraction safe.

**Proof of Issue**

**Filename:** lending/contracts/lendingpool/DefaultReserveInterestRateStrategy.sol

**Beginning Line Number:** 148

```
if (utilizationRate > optimalUtilizationRate) {

            uint256 excessUtilizationRateRatio =

                        utilizationRate.sub(optimalUtilizationRate).rayDiv(

                                    excessUtilizationRate

                        );
```

**Severity and Impact Summary**

N/A

**Recommendation**

Replace the *SafeMath* subtraction with a regular subtraction.

**References**

- N/A

## 3.17 Unnecessary nesting of conditionals

Finding ID: KS-MCL-F-17

Severity: [**Informational**]

Status: [**Remediated**]

**Description**

Description of the finding goes here.

**Proof of Issue**

**Filename:** lending/contracts/misc/ChainlinkProxyPriceProvider.sol

**Beginning Line Number:** 85

```
if (_asset == BscAddressLib.bnbAddress()) {
    return 1 ether;
} else {
    // If there is no registered source for the asset, call the fallbackOracle
    if (address(source) == address(0)) {
        return IPriceOracleGetter(fallbackOracle).getAssetPrice(_asset);
    } else {
        int256 _price = IChainlinkAggregator(source).latestAnswer();
        if (_price > 0) {
            return uint256(_price);
        } else {
            return
                IPriceOracleGetter(fallbackOracle).getAssetPrice(
                    _asset
                );
        }
    }
}
```

**Severity and Impact Summary**

N/A

**Recommendation**

Reduce the visible complexity by removing the unnecessary nesting of conditionals.

If one branch always exits the function, having an else-case is redundant. Place the code inside the else-block outside the if-statement and remove the else-block.

**References**

- N/A

## 3.18 User allowed to borrow too much for a stable asset

Finding ID: KS-MCL-F-18

Severity: [**Informational**]

Status: [**Remediated**]

### Description

The documentation (litepaper) specifies that a stable asset can be borrowed up to a ratio of the user's collateral; however, the external function which checks if a user can borrow a certain amount from a reserve does not take the ratio into account.

Another function that makes use of this check takes the collateral ratio into account; however, because the function is external, the user may have a misunderstanding of the amount they are allowed to borrow.

### Proof of Issue

**Filename:** lending/contracts/lendingpool/LendingPoolCore.sol

**Beginning Line Number:** 674

```
function isUserAllowedToBorrowAtStable(
    address _reserve,
    address _user,
    uint256 _amount
) external view returns(bool)
```

### Severity and Impact Summary

A user may think they can borrow more than is described in the documentation.

### Recommendation

Update the documentation or ensure that the GUI/interface specifically match to avoid misunderstanding.  The underlying code enforces the proper ratios.

### References

- N/A

## APPENDIX A: ABOUT KUDELSKI SECURITY

Kudelski Security is an innovative, independent Swiss provider of tailored cyber and media security solutions to enterprises and public sector institutions. Our team of security experts delivers end-to-end consulting, technology, managed services, and threat intelligence to help organizations build and run successful security programs. Our global reach and cyber solutions focus is reinforced by key international partnerships.

Kudelski Security is a division of Kudelski Group. For more information, please visit https://www.kudelskisecurity.com.

**Kudelski Security**

route de Genève, 22-24
1033 Cheseaux-sur-Lausanne
Switzerland

**Kudelski Security**

5090 North 40th Street, Suite 450
Phoenix, Arizona 85018

## APPENDIX B: SEVERITY RATING DEFINITIONS

Kudelski Security uses a custom approach when determining criticality of identified issues. This is meant to be simple and fast, providing client with a quick at a glance view of the risk an issue poses to the system. As with anything risk related, these findings are situational. We consider multiple factors when assigning a severity level to an identified vulnerability. A few of these include:

- Impact of exploitation
- Ease of exploitation
- Likelihood of attack
- Exposure of attack surface
- Number of instances of identified vulnerability
- Availability of tools and exploits

| SEVERITY | DEFINITION |
|---|---|
| **High** | The identified issue may be directly exploitable causing an immediate negative impact on the users, data, and availability of the system for multiple users. |
| **Medium** | The identified issue is not directly exploitable but combined with other vulnerabilities may allow for exploitation of the system or exploitation may affect singular users. These findings may also increase in severity in the future as techniques evolve. |
| **Low** | The identified issue is not directly exploitable but raises the attack surface of the system. This may be through leaking information that an attacker can use to increase the accuracy of their attacks. |
| **Informational** | Informational findings are best practice steps that can be used to harden the application and improve processes. |

## APPENDIX C: SYSTEM & APPLICATION THREAT MODEL REPORT

The System & Application Threat Model has been included as a separate document.

## APPENDIX D: CONTACTS

| NAME | POSITION | CONTACT INFORMATION |
|---|---|---|
| Scott Carlson | Director – Security Architecture | scott.carlson@kudelskisecurity.com |
| Ryan Spanier | VP – Global Innovation | ryan.spanier@kudelskisecurity.com |
| Amy Fleischer | Program Manager | amy.fleischer@kudelskisecurity.com |
| Ken Toler | Principal Application Security Manager | ken.toler@kudelskisecurity.com |