QuillAudits

Audit Report
June, 2022

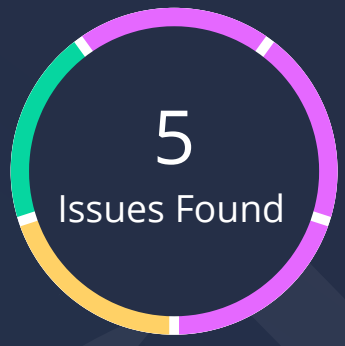
For

ØSPACE

Table of Content

Executive Summary

Project Name	SpaceFi
Timeline	26th May, 2022 to 9th June, 2022
Method	Manual Review, Functional Testing, Automated Testing etc.
Scope of Audit	The scope of this audit was to analyse SpaceFi codebase for quality, security, and correctness.
Git Repo link	*https://github.com/SpaceFinance/space-contract*
Git Branch	*https://github.com/SpaceFinance/space-contract/tree/main*
Commit Hash	e0fbb882a959f28f135cbf36a63ff8953beb026a
Fixed In	7db1cb73997cc593b203dfbe9aff620b25743d73



High
Medium
Low
Informational

	High	Medium	Low	Informational
Open Issues	0	0	0	0
Acknowledged Issues	0	0	0	0
Partially Resolved Issues	0	0	0	0
Resolved Issues	0	1	1	3

Types of Severities

High

A high severity issue or vulnerability means that your smart contract can be exploited. Issues on this level are critical to the smart contract's performance or functionality, and we recommend these issues be fixed before moving to a live environment.

Medium

The issues marked as medium severity usually arise because of errors and deficiencies in the smart contract code. Issues on this level could potentially bring problems, and they should still be fixed.

Low

Low-level severity issues can cause minor impact and or are just warnings that can remain unfixed for now. It would be better to fix these issues at some point in the future.

Informational

These are severity issues that indicate an improvement request, a general question, a cosmetic or documentation error, or a request for information. There is low-to-no impact.

Types of Issues

Open

Security vulnerabilities identified that must be resolved and are currently unresolved.

Resolved

These are the issues identified in the initial audit and have been successfully fixed.

Acknowledged

Vulnerabilities which have been acknowledged but are yet to be resolved.

Partially Resolved

Considerable efforts have been invested to reduce the risk/impact of the security issue, but are not completely resolved.

Checked Vulnerabilities

- Re-entrancy
- Timestamp Dependence
- Gas Limit and Loops
- DoS with Block Gas Limit
- Transaction-Ordering Dependence
- Use of tx.origin
- Exception disorder
- Gasless send
- Balance equality
- Byte array
- Transfer forwards all gas
- BEP20 API violation
- Malicious libraries
- Compiler version not fixed
- Redundant fallback function
- Send instead of transfer
- Style guide violation
- Unchecked external call
- Unchecked math
- Unsafe type inference
- Implicit visibility level

Techniques and Methods

Throughout the audit of smart contract, care was taken to ensure:

- The overall quality of code.
- Use of best practices.
- Code documentation and comments match logic and expected behaviour.
- Token distribution and calculations are as per the intended behaviour mentioned in the whitepaper.
- Implementation of ERC-20 token standards.
- Efficient use of gas.
- Code is safe from re-entrancy and other vulnerabilities.

The following techniques, methods and tools were used to review all the smart contracts.

Structural Analysis

In this step, we have analysed the design patterns and structure of smart contracts. A thorough check was done to ensure the smart contract is structured in a way that will not result in future problems.

Static Analysis

Static analysis of smart contracts was done to identify contract vulnerabilities. In this step, a series of automated tools are used to test the security of smart contracts.

Code Review / Manual Analysis

Manual analysis or review of code was done to identify new vulnerabilities or verify the vulnerabilities found during the static analysis. Contracts were completely manually analysed, their logic was checked and compared with the one described in the whitepaper. Besides, the results of the automated analysis were manually verified.

Gas Consumption

In this step, we have checked the behaviour of smart contracts in production. Checks were done to know how much gas gets consumed and the possibilities of optimization of code to reduce gas consumption.

Tools and Platforms used for Audit

Remix IDE, Truffle, Truffle Team, Solhint, Mythril, Slither, Solidity statistic analysis.

Manual Testing

A. Contract - Auction.sol

High Severity Issues

No issues were found

Medium Severity Issues

A.1 transfer() is being utilized

Description

Low-level transfer() function has been found to be used in the contract at line:

L148: to.transfer(amount);
L177: payable(_msgSender()).transfer(address(this).balance);
L172: _dst.transfer(address(this).balance);

Due to the fact that .transfer() and .send() forward exactly 2,300 gas to the recipient. This hardcoded gas stipend aimed to prevent reentrancy vulnerabilities, but this only makes sense under the assumption that gas costs are constant. Recently EIP 1884 was included in the Istanbul hard fork. One of the changes included in EIP 1884 is an increase to the gas cost of the SLOAD operation, causing a contract's fallback function to cost more than 2300 gas.

```
// bad
contract Vulnerable {
    function withdraw(uint256 amount) external {
        // This forwards 2300 gas, which may not be enough
if the recipient
        // is a contract and gas costs change.
        msg.sender.transfer(amount);
    }
}
```

```
// good
contract Fixed {
    function withdraw(uint256 amount) external {
        // This forwards all available gas. Be sure to check
the return value!
        (bool success, ) = msg.sender.call.value(amount)("");
        require(success, "Transfer failed.");
    }
}
```

Remediation

The auditee needs to ensure that the to, _dst and _msgSender() are not a contract .On the other hand, it's recommended to stop using .transfer() and .send() and instead use .call().

Status

Fixed

Low Severity Issues

A.2 Missing zero address validation

Description

We've detected missing zero address validation for the _dst variable in the rescueTokens() function.

Remediation

Consider implementing require statements where appropriate to validate all user-controlled input, including constructor, to avoid the potential for erroneous values to result in unexpected behaviors or wasted gas.

Status

Fixed

Informational Issues

A.3 Missing error message for require functions

Description

The following require functions are missing the message error:

165: require(address(_token) != address(currency) && address(_token) != address(underlying));

171: require(address(currency) != address(0));

176: require(address(currency) != address(0));

Remediation

We recommend adding the message error for each require function listed above

Status

Fixed

A.4 Commented code

Description

It was discovered that the following code are commented in this contract, for instance:

```
//amt = currency == address(0) ? address(this).balance : IERC20(currency).balanceOf(address(this));
      //amt =
amt.add(totalSettledUnderlying.mul(price).div(settleRate).mul(uint(1e18).sub(settleRate)).div(1e18)).
sub(totalPurchasedCurrency.mul(uint(1e18).sub(settleRate)).div(1e18));
```

Unused code is allowed in Solidity, and they do not pose a direct security issue. It is best practice, though, to avoid them as they can:

- cause an increase in computations (and unnecessary gas consumption)
- indicate bugs or malformed data structures, and they are generally sign of poor code quality
- cause code noise and decrease the readability of the code

Remediation

We recommend removing all unused variables/code from the codebase.

Status

Fixed

A.5 Use double quotes for string literals

Description

Single quote found in the above string variables. Whilst, the double quotes are being utilized for other string literals.

Remediation

We recommend using double quotes for string literals in the entire codebase

Status

Fixed

Functional Testing

Some of the tests performed are mentioned below

- purchase() is reverted when expired
- purchase() is reverted when amount error
- purchase() is reverted when Maximum number exceeded
- purchaseBNB() is reverted when currency is not set
- purchaseBNB() is reverted when expired
- purchaseBNB() is reverted when amount error
- totalSettleable() should return the correct value of settleable
- settle is reverted when block.timestap < time
- settle is reverted when already settled.
- withdraw() should be called only by the owner
- allWithdraw() should be called only by the owner
- rescueTokens() should be called only by the owner
- withdrawBNB() should be called only by the owner

Automated Tests

No major issues were found. Some false positive errors were reported by the tools. All the other issues have been categorized above according to their level of severity.

```
INFO:Detectors:
Starter.withdraw(address,uint256,uint256) (StarStarter.sol#142-153) sends eth to arbitrary user
        Dangerous calls:
        - to.transfer(amount) (StarStarter.sol#148)
Starter.withdrawBNB(address) (StarStarter.sol#170-173) sends eth to arbitrary user
        Dangerous calls:
        - _dst.transfer(address(this).balance) (StarStarter.sol#172)
Starter.withdrawBNB() (StarStarter.sol#175-178) sends eth to arbitrary user
        Dangerous calls:
        - address(_msgSender()).transfer(address(this).balance) (StarStarter.sol#177)
Offering.withdrawBNB(address) (StarStarter.sol#333-336) sends eth to arbitrary user
        Dangerous calls:
        - _dst.transfer(address(this).balance) (StarStarter.sol#335)
Offering.withdrawBNB() (StarStarter.sol#338-341) sends eth to arbitrary user
        Dangerous calls:
        - address(_msgSender()).transfer(address(this).balance) (StarStarter.sol#340)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#functions-that-send-ether-to-arbitrary-destinations
INFO:Detectors:
Reentrancy in Starter.settle() (StarStarter.sol#110-131):
        External calls:
        - currency.safeTransfer(_msgSender(),amount) (StarStarter.sol#123)
        External calls sending eth:
        - address(_msgSender()).transfer(amount) (StarStarter.sol#121)
        State variables written after the call(s):
        - settledUnderlyingOf[_msgSender()] = settledUnderlyingOf[_msgSender()].add(volume) (StarStarter.sol#126)
        - totalSettledUnderlying = totalSettledUnderlying.add(volume) (StarStarter.sol#127)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities
INFO:Detectors:
OwnableUpgradeable.__gap (@openzeppelin/contracts-upgradeable/access/OwnableUpgradeable.sol#87) shadows:
        - ContextUpgradeable.__gap (@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#36)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variable-shadowing
```

```
INFO:Detectors:
renounceOwnership() should be declared external:
        - OwnableUpgradeable.renounceOwnership() (@openzeppelin/contracts-upgradeable/access/OwnableUpgradeable.sol#59-61)
transferOwnership(address) should be declared external:
        - OwnableUpgradeable.transferOwnership(address) (@openzeppelin/contracts-upgradeable/access/OwnableUpgradeable.sol#67-70)
initialize(address,address,uint256,uint256,uint256,uint256,uint256) should be declared external:
        - Starter.initialize(address,address,uint256,uint256,uint256,uint256,uint256) (StarStarter.sol#43-45)
savecompleted() should be declared external:
        - Starter.savecompleted() (StarStarter.sol#64-66)
totalSettleable() should be declared external:
        - Starter.totalSettleable() (StarStarter.sol#88-90)
rescueTokens(address,address) should be declared external:
        - Starter.rescueTokens(address,address) (StarStarter.sol#164-168)
initialize(address,address,uint256,address,uint256,uint256,uint256,uint256) should be declared external:
        - Offering.initialize(address,address,uint256,address,uint256,uint256,uint256,uint256) (StarStarter.sol#221-223)
savecompleted() should be declared external:
        - Offering.savecompleted() (StarStarter.sol#241-243)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
```

```
INFO:Detectors:
Reentrancy in Offering.purchaseBNB() (StarStarter.sol#280-297):
        External calls:
        - recipient.transfer(amount) (StarStarter.sol#289)
        State variables written after the call(s):
        - purchasedUnderlyingOf[_msgSender()] = volume (StarStarter.sol#291)
        - totalPurchasedUnderlying = totalPurchasedUnderlying.add(volume) (StarStarter.sol#292)
Reentrancy in Offering.purchaseBNB() (StarStarter.sol#280-297):
        External calls:
        - recipient.transfer(amount) (StarStarter.sol#289)
        - address(_msgSender()).transfer(msg.value.sub(amount)) (StarStarter.sol#295)
        Event emitted after the call(s):
        - Purchase(_msgSender(),amount,volume,totalPurchasedUnderlying) (StarStarter.sol#296)
Reentrancy in Starter.settle() (StarStarter.sol#110-131):
        External calls:
        - address(_msgSender()).transfer(amount) (StarStarter.sol#121)
        State variables written after the call(s):
        - settledUnderlyingOf[_msgSender()] = settledUnderlyingOf[_msgSender()].add(volume) (StarStarter.sol#126)
        - totalSettledUnderlying = totalSettledUnderlying.add(volume) (StarStarter.sol#127)
        Event emitted after the call(s):
        - Settle(_msgSender(),amount,volume,rate) (StarStarter.sol#130)
Reentrancy in Starter.withdraw(address,uint256,uint256) (StarStarter.sol#142-153):
        External calls:
        - to.transfer(amount) (StarStarter.sol#148)
        Event emitted after the call(s):
        - Withdrawn(to,amount,volume) (StarStarter.sol#152)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-4
INFO:Detectors:
OwnableUpgradeable.__gap (@openzeppelin/contracts-upgradeable/access/OwnableUpgradeable.sol#87) is never used in Starter (StarStarter.sol#14-190)
OwnableUpgradeable.__gap (@openzeppelin/contracts-upgradeable/access/OwnableUpgradeable.sol#87) is never used in Offering (StarStarter.sol#192-354)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variables
```

```
INFO:Detectors:
Function OwnableUpgradeable.__Ownable_init() (@openzeppelin/contracts-upgradeable/access/OwnableUpgradeable.sol#29-31) is not in mixedCase
Function OwnableUpgradeable.__Ownable_init_unchained() (@openzeppelin/contracts-upgradeable/access/OwnableUpgradeable.sol#33-35) is not in mixedCase
Variable OwnableUpgradeable.__gap (@openzeppelin/contracts-upgradeable/access/OwnableUpgradeable.sol#87) is not in mixedCase
Function ContextUpgradeable.__Context_init() (@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#18-19) is not in mixedCase
Function ContextUpgradeable.__Context_init_unchained() (@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#21-22) is not in mixedCase
Variable ContextUpgradeable.__gap (@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#36) is not in mixedCase
Function Starter.__Starter_init(address,address,uint256,uint256,uint256,uint256,uint256) (StarStarter.sol#47-51) is not in mixedCase
Function Starter.__Starter_init_unchained(address,address,uint256,uint256,uint256,uint256,uint256) (StarStarter.sol#53-62) is not in mixedCase
Parameter Starter.rescueTokens(address,address)._token (StarStarter.sol#164) is not in mixedCase
Parameter Starter.rescueTokens(address,address)._dst (StarStarter.sol#164) is not in mixedCase
Parameter Starter.withdrawBNB(address)._dst (StarStarter.sol#170) is not in mixedCase
Function Offering.__Offering_init(address,address,uint256,address,uint256,uint256,uint256,uint256) (StarStarter.sol#225-228) is not in mixedCase
Function Offering.__Offering_init_unchained(address,address,uint256,address,uint256,uint256,uint256,uint256) (StarStarter.sol#230-239) is not in mixedCase
Parameter Offering.rescueTokens(address,address)._token (StarStarter.sol#314) is not in mixedCase
Parameter Offering.rescueTokens(address,address)._dst (StarStarter.sol#314) is not in mixedCase
Parameter Offering.allWithdraw(address,address)._token (StarStarter.sol#320) is not in mixedCase
Parameter Offering.allWithdraw(address,address)._dst (StarStarter.sol#320) is not in mixedCase
Parameter Offering.withdrawToken(address)._dst (StarStarter.sol#325) is not in mixedCase
Parameter Offering.withdrawBNB(address)._dst (StarStarter.sol#333) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
```

```
INFO:Detectors:
Pragma version^0.8.0 (@openzeppelin/contracts-upgradeable/access/OwnableUpgradeable.sol#4) necessitates a version too recent to be trusted. Consider deploying with 0.6
.12/0.7.6
Pragma version^0.8.2 (@openzeppelin/contracts-upgradeable/proxy/utils/Initializable.sol#4) necessitates a version too recent to be trusted. Consider deploying with 0.6
.12/0.7.6
Pragma version^0.8.0 (@openzeppelin/contracts-upgradeable/token/ERC20/IERC20Upgradeable.sol#4) necessitates a version too recent to be trusted. Consider deploying with
 0.6.12/0.7.6
Pragma version^0.8.0 (@openzeppelin/contracts-upgradeable/token/ERC20/utils/SafeERC20Upgradeable.sol#4) necessitates a version too recent to be trusted. Consider deplo
ying with 0.6.12/0.7.6
Pragma version^0.8.1 (@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#4) necessitates a version too recent to be trusted. Consider deploying with 0.6.
12/0.7.6
Pragma version^0.8.0 (@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#4) necessitates a version too recent to be trusted. Consider deploying with 0.6.
12/0.7.6
Pragma version^0.8.0 (@openzeppelin/contracts-upgradeable/utils/math/MathUpgradeable.sol#4) necessitates a version too recent to be trusted. Consider deploying with 0.
6.12/0.7.6
Pragma version^0.8.0 (@openzeppelin/contracts-upgradeable/utils/math/SafeMathUpgradeable.sol#4) necessitates a version too recent to be trusted. Consider deploying wit
h 0.6.12/0.7.6
Pragma version^0.8.0 (StarStarter.sol#3) necessitates a version too recent to be trusted. Consider deploying with 0.6.12/0.7.6
solc-0.8.9 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Low level call in AddressUpgradeable.sendValue(address,uint256) (@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#60-65):
        - (success) = recipient.call{value: amount}() (@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#63)
Low level call in AddressUpgradeable.functionCallWithValue(address,bytes,uint256,string) (@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#128-139):
        - (success,returndata) = target.call{value: value}(data) (@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#137)
Low level call in AddressUpgradeable.functionStaticCall(address,bytes,string) (@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#157-166):
        - (success,returndata) = target.staticcall(data) (@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#164)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
```

```
INFO:Detectors:
Different versions of Solidity is used:
        - Version used: ['^0.8.0', '^0.8.1', '^0.8.2']
        - ^0.8.0 (@openzeppelin/contracts-upgradeable/access/OwnableUpgradeable.sol#4)
        - ^0.8.2 (@openzeppelin/contracts-upgradeable/proxy/utils/Initializable.sol#4)
        - ^0.8.0 (@openzeppelin/contracts-upgradeable/token/ERC20/IERC20Upgradeable.sol#4)
        - ^0.8.0 (@openzeppelin/contracts-upgradeable/token/ERC20/utils/SafeERC20Upgradeable.sol#4)
        - ^0.8.1 (@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#4)
        - ^0.8.0 (@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#4)
        - ^0.8.0 (@openzeppelin/contracts-upgradeable/utils/math/MathUpgradeable.sol#4)
        - ^0.8.0 (@openzeppelin/contracts-upgradeable/utils/math/SafeMathUpgradeable.sol#4)
        - ^0.8.0 (StarStarter.sol#3)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used
INFO:Detectors:
AddressUpgradeable.functionCall(address,bytes) (@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#85-87) is never used and should be removed
AddressUpgradeable.functionCallWithValue(address,bytes,uint256) (@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#114-120) is never used and should be
removed
AddressUpgradeable.functionStaticCall(address,bytes) (@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#147-149) is never used and should be removed
AddressUpgradeable.functionStaticCall(address,bytes,string) (@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#157-166) is never used and should be remo
ved
AddressUpgradeable.sendValue(address,uint256) (@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#60-65) is never used and should be removed
ContextUpgradeable.__Context_init() (@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#18-19) is never used and should be removed
ContextUpgradeable._msgData() (@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol#27-29) is never used and should be removed
Initializable._disableInitializers() (@openzeppelin/contracts-upgradeable/proxy/utils/Initializable.sol#129-131) is never used and should be removed
MathUpgradeable.average(uint256,uint256) (@openzeppelin/contracts-upgradeable/utils/math/MathUpgradeable.sol#28-31) is never used and should be removed
MathUpgradeable.ceilDiv(uint256,uint256) (@openzeppelin/contracts-upgradeable/utils/math/MathUpgradeable.sol#39-42) is never used and should be removed
MathUpgradeable.max(uint256,uint256) (@openzeppelin/contracts-upgradeable/utils/math/MathUpgradeable.sol#13-15) is never used and should be removed
OwnableUpgradeable.__Ownable_init() (@openzeppelin/contracts-upgradeable/access/OwnableUpgradeable.sol#29-31) is never used and should be removed
SafeERC20Upgradeable.safeApprove(IERC20Upgradeable,address,uint256) (@openzeppelin/contracts-upgradeable/token/ERC20/utils/SafeERC20Upgradeable.sol#45-58) is never use
d and should be removed
SafeERC20Upgradeable.safeDecreaseAllowance(IERC20Upgradeable,address,uint256) (@openzeppelin/contracts-upgradeable/token/ERC20/utils/SafeERC20Upgradeable.sol#69-80) is
 never used and should be removed
SafeERC20Upgradeable.safeIncreaseAllowance(IERC20Upgradeable,address,uint256) (@openzeppelin/contracts-upgradeable/token/ERC20/utils/SafeERC20Upgradeable.sol#60-67) is
 never used and should be removed
SafeMathUpgradeable.div(uint256,uint256,string) (@openzeppelin/contracts-upgradeable/utils/math/SafeMathUpgradeable.sol#191-200) is never used and should be removed
SafeMathUpgradeable.mod(uint256,uint256) (@openzeppelin/contracts-upgradeable/utils/math/SafeMathUpgradeable.sol#151-153) is never used and should be removed
SafeMathUpgradeable.mod(uint256,uint256,string) (@openzeppelin/contracts-upgradeable/utils/math/SafeMathUpgradeable.sol#217-226) is never used and should be removed
SafeMathUpgradeable.sub(uint256,uint256,string) (@openzeppelin/contracts-upgradeable/utils/math/SafeMathUpgradeable.sol#168-177) is never used and should be removed
SafeMathUpgradeable.tryAdd(uint256,uint256) (@openzeppelin/contracts-upgradeable/utils/math/SafeMathUpgradeable.sol#22-28) is never used and should be removed
SafeMathUpgradeable.tryDiv(uint256,uint256) (@openzeppelin/contracts-upgradeable/utils/math/SafeMathUpgradeable.sol#64-69) is never used and should be removed
SafeMathUpgradeable.tryMod(uint256,uint256) (@openzeppelin/contracts-upgradeable/utils/math/SafeMathUpgradeable.sol#76-81) is never used and should be removed
SafeMathUpgradeable.tryMul(uint256,uint256) (@openzeppelin/contracts-upgradeable/utils/math/SafeMathUpgradeable.sol#47-67) is never used and should be removed
SafeMathUpgradeable.trySub(uint256,uint256) (@openzeppelin/contracts-upgradeable/utils/math/SafeMathUpgradeable.sol#35-40) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
```

```
INFO:Detectors:
Starter.purchase(uint256) (StarStarter.sol#68-76) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(block.timestamp < time,expired) (StarStarter.sol#69)
Starter.purchaseBNB() (StarStarter.sol#78-86) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(block.timestamp < time,expired) (StarStarter.sol#80)
Starter.settle() (StarStarter.sol#110-131) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(block.timestamp >= time,It is not time yet) (StarStarter.sol#111)
        - require(bool,string)(amount > 0 || block.timestamp >= timeSettle,It is not time to settle underlying) (StarStarter.sol#124)
        - block.timestamp >= timeSettle (StarStarter.sol#125)
Offering.purchase(uint256) (StarStarter.sol#261-278) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(block.timestamp >= time,it's not time yet) (StarStarter.sol#263)
        - require(bool,string)(block.timestamp < timeSettle,expired) (StarStarter.sol#264)
Offering.purchaseBNB() (StarStarter.sol#280-297) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(block.timestamp >= time,it's not time yet) (StarStarter.sol#282)
        - require(bool,string)(block.timestamp < timeSettle,expired) (StarStarter.sol#283)
Offering.settle() (StarStarter.sol#299-309) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(block.timestamp >= timeSettle,It is not time yet) (StarStarter.sol#300)
Offering.rescueTokens(address,address) (StarStarter.sol#314-318) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool)(block.timestamp > timeSettle) (StarStarter.sol#315)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
INFO:Detectors:
AddressUpgradeable.verifyCallResult(bool,bytes,string) (@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#174-194) uses assembly
        - INLINE ASM (@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol#186-189)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
```

```
INFO:Detectors:
Reentrancy in Starter.allWithdraw(address,uint256,uint256) (StarStarter.sol#155-159):
        External calls:
        - currency.safeTransfer(to,amount) (StarStarter.sol#156)
        - underlying.safeTransfer(to,volume) (StarStarter.sol#157)
        Event emitted after the call(s):
        - Withdrawn(to,amount,volume) (StarStarter.sol#158)
Reentrancy in Starter.purchase(uint256) (StarStarter.sol#68-76):
        External calls:
        - currency.safeTransferFrom(_msgSender(),address(this),amount) (StarStarter.sol#72)
        Event emitted after the call(s):
        - Purchase(_msgSender(),amount,totalPurchasedCurrency) (StarStarter.sol#75)
Reentrancy in Offering.purchase(uint256) (StarStarter.sol#261-278):
        External calls:
        - currency.safeTransferFrom(_msgSender(),recipient,amount) (StarStarter.sol#272)
        Event emitted after the call(s):
        - Purchase(_msgSender(),amount,volume,totalPurchasedUnderlying) (StarStarter.sol#277)
Reentrancy in Starter.settle() (StarStarter.sol#110-131):
        External calls:
        - currency.safeTransfer(_msgSender(),amount) (StarStarter.sol#123)
        - underlying.safeTransfer(_msgSender(),volume) (StarStarter.sol#128)
        External calls sending eth:
        - address(_msgSender()).transfer(amount) (StarStarter.sol#121)
        Event emitted after the call(s):
        - Settle(_msgSender(),amount,volume,rate) (StarStarter.sol#130)
Reentrancy in Offering.settle() (StarStarter.sol#299-309):
        External calls:
        - underlying.safeTransfer(recipient,underlying.balanceOf(address(this)).add(totalSettledUnderlying).sub(totalPurchasedUnderlying)) (StarStarter.sol#303)
        - underlying.safeTransfer(_msgSender(),volume) (StarStarter.sol#307)
        Event emitted after the call(s):
        - Settle(_msgSender(),volume,totalSettledUnderlying) (StarStarter.sol#308)
Reentrancy in Starter.withdraw(address,uint256,uint256) (StarStarter.sol#142-153):
        External calls:
        - currency.safeTransfer(to,amount) (StarStarter.sol#150)
        - underlying.safeTransfer(to,volume) (StarStarter.sol#151)
        External calls sending eth:
        - to.transfer(amount) (StarStarter.sol#148)
        Event emitted after the call(s):
        - Withdrawn(to,amount,volume) (StarStarter.sol#152)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3
```

```
INFO:Detectors:
Starter.withdraw(address,uint256,uint256).to (StarStarter.sol#142) lacks a zero-check on :
                - to.transfer(amount) (StarStarter.sol#148)
Starter.withdrawBNB(address)._dst (StarStarter.sol#170) lacks a zero-check on :
                - _dst.transfer(address(this).balance) (StarStarter.sol#172)
Offering.withdrawBNB(address)._dst (StarStarter.sol#333) lacks a zero-check on :
                - _dst.transfer(address(this).balance) (StarStarter.sol#335)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
Reentrancy in Starter.purchase(uint256) (StarStarter.sol#68-76):
        External calls:
        - currency.safeTransferFrom(_msgSender(),address(this),amount) (StarStarter.sol#72)
        State variables written after the call(s):
        - totalPurchasedCurrency = totalPurchasedCurrency.add(amount) (StarStarter.sol#74)
Reentrancy in Offering.purchase(uint256) (StarStarter.sol#261-278):
        External calls:
        - currency.safeTransferFrom(_msgSender(),recipient,amount) (StarStarter.sol#272)
        State variables written after the call(s):
        - totalPurchasedUnderlying = totalPurchasedUnderlying.add(volume) (StarStarter.sol#275)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2
```

```
INFO:Detectors:
Starter.settleable(address) (StarStarter.sol#92-108) performs a multiplication on the result of a division:
        -rate = totalUnderlying.mul(price).div(totalCurrency) (StarStarter.sol#100)
        -settleAmount = purchasedCurrency.mul(rate).div(1e18) (StarStarter.sol#105)
Starter.settleable(address) (StarStarter.sol#92-108) performs a multiplication on the result of a division:
        -settleAmount = purchasedCurrency.mul(rate).div(1e18) (StarStarter.sol#105)
        -volume = settleAmount.mul(1e18).div(price).sub(totalSettledUnderlying) (StarStarter.sol#107)
Starter.settleable(address) (StarStarter.sol#92-108) performs a multiplication on the result of a division:
        -settleAmount = purchasedCurrency.mul(rate).div(1e18) (StarStarter.sol#105)
        -volume = settleAmount.mul(1e18).div(price).sub(settledUnderlyingOf[acct]) (StarStarter.sol#107)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply
INFO:Detectors:
Reentrancy in Starter.purchase(uint256) (StarStarter.sol#68-76):
        External calls:
        - currency.safeTransferFrom(_msgSender(),address(this),amount) (StarStarter.sol#72)
        State variables written after the call(s):
        - purchasedCurrencyOf[_msgSender()] = purchasedCurrencyOf[_msgSender()].add(amount) (StarStarter.sol#73)
Reentrancy in Offering.purchase(uint256) (StarStarter.sol#261-278):
        External calls:
        - currency.safeTransferFrom(_msgSender(),recipient,amount) (StarStarter.sol#272)
        State variables written after the call(s):
        - purchasedUnderlyingOf[_msgSender()] = volume (StarStarter.sol#274)
Reentrancy in Offering.settle() (StarStarter.sol#299-309):
        External calls:
        - underlying.safeTransfer(recipient,underlying.balanceOf(address(this)).add(totalSettledUnderlying).sub(totalPurchasedUnderlying)) (StarStarter.sol#303)
        State variables written after the call(s):
        - settledUnderlyingOf[_msgSender()] = volume (StarStarter.sol#305)
        - totalSettledUnderlying = totalSettledUnderlying.add(volume) (StarStarter.sol#306)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1
```

Solhint

```
StarStarter.sol
    3:1   error    Compiler version ^0.8.0 does not satisfy the ^0.5.8 semver requirement   compiler-version
   31:5   warning  Explicitly mark visibility of state                                      state-visibility
   47:5   warning  Function name must be in mixedCase                                       func-name-mixedcase
   53:5   warning  Function name must be in mixedCase                                       func-name-mixedcase
   61:39  error    Use double quotes for string literals                                    quotes
   69:17  warning  Avoid to make time-based decisions in your business logic                not-rely-on-time
   69:41  error    Use double quotes for string literals                                    quotes
   79:9   warning  Error message for require is too long                                    reason-string
   79:50  error    Use double quotes for string literals                                    quotes
   80:17  warning  Avoid to make time-based decisions in your business logic                not-rely-on-time
   80:41  error    Use double quotes for string literals                                    quotes
  111:17  warning  Avoid to make time-based decisions in your business logic                not-rely-on-time
  112:98  error    Use double quotes for string literals                                    quotes
  124:13  warning  Error message for require is too long                                    reason-string
  124:35  warning  Avoid to make time-based decisions in your business logic                not-rely-on-time
  124:66  error    Use double quotes for string literals                                    quotes
  125:12  warning  Avoid to make time-based decisions in your business logic                not-rely-on-time
  126:13  warning  Possible reentrancy vulnerabilities. Avoid state changes after transfer  reentrancy
  127:13  warning  Possible reentrancy vulnerabilities. Avoid state changes after transfer  reentrancy
  165:9   warning  Provide an error message for require                                    reason-string
  171:9   warning  Provide an error message for require                                    reason-string
  176:9   warning  Provide an error message for require                                    reason-string
  187:5   warning  When fallback is not payable you will not be able to receive ether      payable-fallback
  225:5   warning  Function name must be in mixedCase                                      func-name-mixedcase
  230:5   warning  Function name must be in mixedCase                                      func-name-mixedcase
  242:22  warning  Avoid to make time-based decisions in your business logic                not-rely-on-time
  262:9   warning  Error message for require is too long                                    reason-string
  262:50  error    Use double quotes for string literals                                    quotes
  263:17  warning  Avoid to make time-based decisions in your business logic                not-rely-on-time
  264:17  warning  Avoid to make time-based decisions in your business logic                not-rely-on-time
  267:29  error    Use double quotes for string literals                                    quotes
  268:76  error    Use double quotes for string literals                                    quotes
  269:61  error    Use double quotes for string literals                                    quotes
  270:59  error    Use double quotes for string literals                                    quotes
  276:82  error    Use double quotes for string literals                                    quotes
  281:9   warning  Error message for require is too long                                    reason-string
  281:50  error    Use double quotes for string literals                                    quotes
  282:17  warning  Avoid to make time-based decisions in your business logic                not-rely-on-time
  283:17  warning  Avoid to make time-based decisions in your business logic                not-rely-on-time
  286:29  error    Use double quotes for string literals                                    quotes
  287:59  error    Use double quotes for string literals                                    quotes
  291:9   warning  Possible reentrancy vulnerabilities. Avoid state changes after transfer  reentrancy
  292:9   warning  Possible reentrancy vulnerabilities. Avoid state changes after transfer  reentrancy
  293:82  error    Use double quotes for string literals                                    quotes
  300:17  warning  Avoid to make time-based decisions in your business logic                not-rely-on-time
  315:9   warning  Provide an error message for require                                    reason-string
```

```
  300:17  warning  Avoid to make time-based decisions in your business logic                not-rely-on-time
  315:9   warning  Provide an error message for require                                    reason-string
  315:17  warning  Avoid to make time-based decisions in your business logic                not-rely-on-time
  334:9   warning  Provide an error message for require                                    reason-string
  339:9   warning  Provide an error message for require                                    reason-string
  350:5   warning  When fallback is not payable you will not be able to receive ether      payable-fallback

✖ 50 problems (17 errors, 33 warnings)
```

Closing Summary

In this report, we have considered the security of the SpaceFi smart contracts. We performed our audit according to the procedure described above.

Some issues of Medium, Low and informational severity were found, some suggestions and best practices are also provided in order to improve the code quality and security posture.

In the End, the SpaceFi Team Resolved all Issues.

Disclaimer

QuillAudits smart contract audit is not a security warranty, investment advice, or an endorsement of the SpaceFi Platform. This audit does not provide a security or correctness guarantee of the audited smart contracts.

The statements made in this document should not be interpreted as investment or legal advice, nor should its authors be held accountable for decisions made based on them. Securing smart contracts is a multistep process. One audit cannot be considered enough. We recommend that the SpaceFi Team put in place a bug bounty program to encourage further analysis of the smart contract by other third parties.

About QuillAudits

QuillAudits is a secure smart contracts audit platform designed by QuillHash Technologies. We are a team of dedicated blockchain security experts and smart contract auditors determined to ensure that Smart Contract-based Web3 projects can avail the latest and best security solutions to operate in a trustworthy and risk-free ecosystem.

500+
Audits Completed

$15B
Secured

500K
Lines of Code Audited

Follow Our Journey

Audit Report
June, 2022

For

SPACE

QuillAudits

Canada, India, Singapore, United Kingdom

audits.quillhash.com

audits@quillhash.com