QuillAudits

Audit Report
February, 2022

For

HarmonyLauncher

Contents

Scope of the Audit

The scope of this audit was to analyze and document the HarmonyLauncher smart contract codebase for quality, security, and correctness.

Checked Vulnerabilities

We have scanned the smart contract for commonly known and more specific vulnerabilities. Here are some of the commonly known vulnerabilities that we considered:

- 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
- ERC20 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, Theo.

Issue Categories

Every issue in this report has been assigned to a severity level. There are four levels of severity, and each of them has been explained below.

Risk-level	Description
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.

Number of issues per severity

Type	High	Medium	Low	Informational
Open	0	0	0	0
Acknowledged	0	0	0	2
Closed	0	0	0	0

Introduction

During the period of **Feb 08, 2022 to Feb 10 , 2022** - QuillAudits Team performed a security audit for **HarmonyLauncher** smart contracts.

The code for the audit was taken from following the official link:

V	Date	Contract Address
1	Feb 08 2022	https://bscscan.com/address/0x54f85cdb537d5a5ebc76f55b13c6dbc3b6828d57#code

Issues Found – Code Review / Manual Testing

A.Contract - HarmonyLauncher.sol

High severity issues

No issues were found.

Medium severity issues

No issues were found.

Low severity issues

No issues were found.

Informational issues

1. **Typos**

   Please consider changing the following typo:
   - L380: **Destoys** should be **Destroys**

   Status: Acknowledged

2. No error messages in require() functions

Description

There are few places in the entire codebase where the require() statement does not contain any error message. As error messages are intended to notify users about failing conditions, they should provide enough information so that appropriate corrections can be made to interact with the system. Below is a non-exhaustive list of identified instances:

- L24: require(msg.sender == owner);
- L32: require(msg.sender == newOwner);
- L46: require(!paused);
- L51: require(paused);

Remediation

Lack of error messages greatly damage the overall user experience, thus lowering the system's quality. Consider not only fixing the specific instances mentioned above, but also reviewing the entire codebase to make sure every error message is informative and user-friendly.

Status: Acknowledged

Functional Tests

Testing Token Harmony	Status
Testing getters	
should get name of the contract	Passed
should get symbol of the contract	Passed
should get owner of the contract	Passed
should get decimals of the contract	Passed
should get total supply of the contract	Passed
Testing Setters	
should get balance of the tokenHolder	Passed
should return if user has no balance	Passed
owner should transfer to account 1	Passed
Should revert on zero address transfer	Passed
balance of account 1 should be updated correctly	Passed
balance of owner should be updated correctly	Passed
should call approval to account 2	Passed
allowance between account 1 and account 2 should return correctly	Passed
testing decreaseAllowance() and increaseAllowance()	Passed
allowance value should be updated after calling decreaseAllowance()	Passed
allowance value should be updated after calling increaseAllowance()	Passed
testing TransferFrom()	Passed
should revert if exceeds the allowance	Passed
balance of account 1 should be updated correctly	Passed
balance of account 2 should be updated correctly	Passed
allowance value should be updated correctly	Passed
testing burn()	Passed
Testing burn with Zero balance	Passed

totalSupply should be updated correctly after burn	Passed
pause and paused should be called only by Owner	Passed
transfer(), transferFrom(), burn() are stopped when pause is True	Passed
testing transferOwnership	Passed

Automated Tests

Slither

```
INFO:Detectors:
BEP20.allowance(address,address).owner (HarmonyLauncher.sol#286) shadows:
        - Owned.owner (HarmonyLauncher.sol#14) (state variable)
BEP20._approve(address,address,uint256).owner (HarmonyLauncher.sol#430) shadows:
        - Owned.owner (HarmonyLauncher.sol#14) (state variable)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing
INFO:Detectors:
Owned.transferOwnership(address)._newOwner (HarmonyLauncher.sol#28) lacks a zero-check on :
                - newOwner = _newOwner (HarmonyLauncher.sol#29)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
SafeMath.div(uint256,uint256) (HarmonyLauncher.sol#223-230) is never used and should be removed
SafeMath.mul(uint256,uint256) (HarmonyLauncher.sol#198-210) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Parameter Owned.transferOwnership(address)._newOwner (HarmonyLauncher.sol#28) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
HarmonyLauncher.constructor(address) (HarmonyLauncher.sol#448-454) uses literals with too many digits:
        - _mint(tokenHolder,2000000000 * 10 ** 18) (HarmonyLauncher.sol#452)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits
INFO:Detectors:
totalSupply() should be declared external:
        - BEP20.totalSupply() (HarmonyLauncher.sol#271-273)
balanceOf(address) should be declared external:
        - BEP20.balanceOf(address) (HarmonyLauncher.sol#278-280)
allowance(address,address) should be declared external:
        - BEP20.allowance(address,address) (HarmonyLauncher.sol#286-288)
approve(address,uint256) should be declared external:
        - BEP20.approve(address,uint256) (HarmonyLauncher.sol#297-300)
increaseAllowance(address,uint256) should be declared external:
        - BEP20.increaseAllowance(address,uint256) (HarmonyLauncher.sol#315-318)
decreaseAllowance(address,uint256) should be declared external:
        - BEP20.decreaseAllowance(address,uint256) (HarmonyLauncher.sol#334-337)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
```

SOLHINT LINTER

```
======SOLHINT======


HarmonyLauncher.sol
   24:9   warning  Provide an error message for require                    reason-string
   32:9   warning  Provide an error message for require                    reason-string
   46:7   warning  Provide an error message for require                    reason-string
   51:7   warning  Provide an error message for require                    reason-string
   55:46  warning  Visibility modifier must be first in list of modifiers  visibility-modifier-order
   60:45  warning  Visibility modifier must be first in list of modifiers  visibility-modifier-order
  207:9   warning  Error message for require is too long                   reason-string
  354:9   warning  Error message for require is too long                   reason-string
  355:9   warning  Error message for require is too long                   reason-string
  391:9   warning  Error message for require is too long                   reason-string
  431:9   warning  Error message for require is too long                   reason-string
  432:9   warning  Error message for require is too long                   reason-string
  445:5   warning  Constant name must be in capitalized SNAKE_CASE         const-name-snakecase

✖ 13 problems (0 errors, 13 warnings)
```

Mythril

```
enderphan@enderphan contracts % myth a HarmonyLauncher.sol

The analysis was completed successfully. No issues were detected.
```

Manticore

```
2022-02-11 16:10:35,434: [82728] m.c.manticore:WARNING: Manticore is only supported on Linux. Proceed at your own risk!
2022-02-11 16:10:35,595: [82728] m.main:INFO: Registered plugins: <class 'manticore.core.plugin.IntrospectionAPIPlugin'>, DetectInvalid, DetectIntegerOverflow, DetectUninitialize
dStorage, DetectUninitializedMemory, DetectReentrancySimple, DetectReentrancyAdvanced, DetectUnusedRetVal, DetectSuicidal, DetectDelegatecall, DetectExternalCallAndLeak, DetectEn
vInstruction, DetectManipulableBalance
2022-02-11 16:10:35,595: [82728] m.main:INFO: Beginning analysis
2022-02-11 16:10:35,599: [82728] m.e.manticore:INFO: Starting symbolic create contract
2022-02-11 16:30:30,393: [82728] m.e.detectors:WARNING: Potentially reading uninitialized memory at instruction (address: 127024756529990889082940626980867659341945888 1132, offse
t 128)
2022-02-11 16:30:30,535: [82728] m.e.detectors:WARNING: Potentially reading uninitialized memory at instruction (address: 127024756529990889082940626980867659341945888 1132, offse
t 129)
2022-02-11 16:30:30,669: [82728] m.e.detectors:WARNING: Potentially reading uninitialized memory at instruction (address: 127024756529990889082940626980867659341945888 1132, offse
t 130)
2022-02-11 16:30:30,805: [82728] m.e.detectors:WARNING: Potentially reading uninitialized memory at instruction (address: 127024756529990889082940626980867659341945888 1132, offse
t 131)
2022-02-11 16:30:30,948: [82728] m.e.detectors:WARNING: Potentially reading uninitialized memory at instruction (address: 127024756529990889082940626980867659341945888 1132, offse
t 132)
2022-02-11 16:30:31,085: [82728] m.e.detectors:WARNING: Potentially reading uninitialized memory at instruction (address: 127024756529990889082940626980867659341945888 1132, offse
t 133)
2022-02-11 16:30:31,214: [82728] m.e.detectors:WARNING: Potentially reading uninitialized memory at instruction (address: 127024756529990889082940626980867659341945888 1132, offse
t 134)
2022-02-11 16:30:31,346: [82728] m.e.detectors:WARNING: Potentially reading uninitialized memory at instruction (address: 127024756529990889082940626980867659341945888 1132, offse
t 135)
2022-02-11 16:30:31,481: [82728] m.e.detectors:WARNING: Potentially reading uninitialized memory at instruction (address: 127024756529990889082940626980867659341945888 1132, offse
t 136)
2022-02-11 16:30:31,611: [82728] m.e.detectors:WARNING: Potentially reading uninitialized memory at instruction (address: 127024756529990889082940626980867659341945888 1132, offse
t 137)
2022-02-11 16:30:31,741: [82728] m.e.detectors:WARNING: Potentially reading uninitialized memory at instruction (address: 127024756529990889082940626980867659341945888 1132, offse
t 138)
2022-02-11 16:30:31,877: [82728] m.e.detectors:WARNING: Potentially reading uninitialized memory at instruction (address: 127024756529990889082940626980867659341945888 1132, offse
t 139)
2022-02-11 16:30:32,028: [82728] m.e.detectors:WARNING: Potentially reading uninitialized memory at instruction (address: 127024756529990889082940626980867659341945888 1132, offse
t 140)
2022-02-11 16:30:32,187: [82728] m.e.detectors:WARNING: Potentially reading uninitialized memory at instruction (address: 127024756529990889082940626980867659341945888 1132, offse
t 141)
2022-02-11 16:30:32,332: [82728] m.e.detectors:WARNING: Potentially reading uninitialized memory at instruction (address: 127024756529990889082940626980867659341945888 1132, offse
t 142)
2022-02-11 16:30:32,469: [82728] m.e.detectors:WARNING: Potentially reading uninitialized memory at instruction (address: 127024756529990889082940626980867659341945888 1132, offse
t 143)
2022-02-11 16:30:32,611: [82728] m.e.detectors:WARNING: Potentially reading uninitialized memory at instruction (address: 127024756529990889082940626980867659341945888 1132, offse
t 144)
2022-02-11 16:30:32,751: [82728] m.e.detectors:WARNING: Potentially reading uninitialized memory at instruction (address: 127024756529990889082940626980867659341945888 1132, offse
t 145)
2022-02-11 16:30:32,902: [82728] m.e.detectors:WARNING: Potentially reading uninitialized memory at instruction (address: 127024756529990889082940626980867659341945888 1132, offse
t 146)
2022-02-11 16:30:33,049: [82728] m.e.detectors:WARNING: Potentially reading uninitialized memory at instruction (address: 127024756529990889082940626980867659341945888 1132, offse
t 147)
2022-02-11 16:30:33,187: [82728] m.e.detectors:WARNING: Potentially reading uninitialized memory at instruction (address: 127024756529990889082940626980867659341945888 1132, offse
t 148)
2022-02-11 16:30:33,330: [82728] m.e.detectors:WARNING: Potentially reading uninitialized memory at instruction (address: 127024756529990889082940626980867659341945888 1132, offse
t 149)
2022-02-11 16:30:33,471: [82728] m.e.detectors:WARNING: Potentially reading uninitialized memory at instruction (address: 127024756529990889082940626980867659341945888 1132, offse
t 150)
2022-02-11 16:30:33,612: [82728] m.e.detectors:WARNING: Potentially reading uninitialized memory at instruction (address: 127024756529990889082940626980867659341945888 1132, offse
t 151)
2022-02-11 16:30:33,752: [82728] m.e.detectors:WARNING: Potentially reading uninitialized memory at instruction (address: 127024756529990889082940626980867659341945888 1132, offse
t 152)
2022-02-11 16:30:33,893: [82728] m.e.detectors:WARNING: Potentially reading uninitialized memory at instruction (address: 127024756529990889082940626980867659341945888 1132, offse
t 153)
2022-02-11 16:30:34,036: [82728] m.e.detectors:WARNING: Potentially reading uninitialized memory at instruction (address: 127024756529990889082940626980867659341945888 1132, offse
t 154)
2022-02-11 16:30:34,200: [82728] m.e.detectors:WARNING: Potentially reading uninitialized memory at instruction (address: 127024756529990889082940626980867659341945888 1132, offse
t 155)
2022-02-11 16:30:34,361: [82728] m.e.detectors:WARNING: Potentially reading uninitialized memory at instruction (address: 127024756529990889082940626980867659341945888 1132, offse
t 156)
2022-02-11 16:30:34,510: [82728] m.e.detectors:WARNING: Potentially reading uninitialized memory at instruction (address: 127024756529990889082940626980867659341945888 1132, offse
t 157)
2022-02-11 16:30:34,654: [82728] m.e.detectors:WARNING: Potentially reading uninitialized memory at instruction (address: 127024756529990889082940626980867659341945888 1132, offse
t 158)
2022-02-11 16:30:34,814: [82728] m.e.detectors:WARNING: Potentially reading uninitialized memory at instruction (address: 127024756529990889082940626980867659341945888 1132, offse
t 159)
2022-02-11 16:31:06,323: [82728] m.e.manticore:INFO: Found a concrete globalsha3 b'\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x0
0\x00\x00\x00\x00\x00\x00\x00\x05' -> 1546678032441257452667456735582814959992782782816731922691272282333561699760
2022-02-11 16:31:48,323: [82728] m.e.manticore:INFO: Found a concrete globalsha3 b'\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x0
0\x00\x00\x00\x00\x00\x00\x00\x06' -> 1114140778158634005100040646299735959615791736655892242035036621493737249866 87
2022-02-11 16:32:46,395: [82728] m.e.detectors:WARNING: Potentially reading uninitialized storage
2022-02-11 16:33:49,107: [82728] m.e.manticore:INFO: Starting symbolic transaction: 0
2022-02-11 16:50:53,464: [82728] m.e.detectors:WARNING: Potentially reading uninitialized storage
2022-02-11 16:51:22,490: [82728] m.e.manticore:INFO: Found a concrete globalsha3 b'\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\xc6\xbc\x8d\xd2\xd0\x1eE\xc4CN=p0JX\x91\x8a\xe
9\xabl\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x03' -> 805684704994635974813047623714855283689
871997271296046392691726606982141412 53
2022-02-11 16:53:02,209: [82728] m.e.detectors:WARNING: Potentially reading uninitialized storage
2022-02-11 16:53:42,880: [82728] m.e.detectors:WARNING: Potentially reading uninitialized storage
2022-02-11 16:58:33,187: [82728] m.e.detectors:WARNING: Potentially reading uninitialized storage
2022-02-11 17:28:56,336: [82728] m.e.manticore:INFO: Found a concrete globalsha3 b'\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\xc6\xbc\x8d\xd2\xd0\x1eE\xc4CN=p0JX\x91\x8a\xe
9\xabl\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x02' -> 674245331932323800511433164767216005776
397821560249488543122674306866707185 86
2022-02-11 17:28:58,516: [82728] m.e.detectors:WARNING: Potentially reading uninitialized storage
2022-02-11 17:33:46,418: [82728] m.e.detectors:WARNING: Potentially reading uninitialized storage
2022-02-11 17:35:57,328: [82728] m.e.detectors:WARNING: Potentially reading uninitialized storage
```

Solidity static analysis

Gas costs:

Gas requirement of function HarmonyLauncher.approve is infinite:
If the gas requirement of a function is higher than the block gas limit, it cannot be executed.
Please avoid loops in your functions or actions that modify large areas of storage
(this includes clearing or copying arrays in storage)
Pos: 297:4:

Gas costs:

Gas requirement of function HarmonyLauncher.increaseAllowance is infinite:
If the gas requirement of a function is higher than the block gas limit, it cannot be executed.
Please avoid loops in your functions or actions that modify large areas of storage
(this includes clearing or copying arrays in storage)
Pos: 315:4:

Gas costs:

Gas requirement of function HarmonyLauncher.decreaseAllowance is infinite:
If the gas requirement of a function is higher than the block gas limit, it cannot be executed.
Please avoid loops in your functions or actions that modify large areas of storage
(this includes clearing or copying arrays in storage)
Pos: 334:4:

Gas costs:

Gas requirement of function HarmonyLauncher.name is infinite:
If the gas requirement of a function is higher than the block gas limit, it cannot be executed.
Please avoid loops in your functions or actions that modify large areas of storage
(this includes clearing or copying arrays in storage)
Pos: 443:4:

Gas costs:

Gas requirement of function HarmonyLauncher.symbol is infinite:
If the gas requirement of a function is higher than the block gas limit, it cannot be executed.
Please avoid loops in your functions or actions that modify large areas of storage
(this includes clearing or copying arrays in storage)
Pos: 444:4:

Data truncated:

Division of integer values yields an integer value again. That means e.g. 10 / 100 = 0 instead of 0.1 since the result is an integer again. This does not hold for division of (only) literal values since those yield rational constants.
Pos: 207:16:

Data truncated:

Division of integer values yields an integer value again. That means e.g. 10 / 100 = 0 instead of 0.1 since the result is an integer again. This does not hold for division of (only) literal values since those yield rational constants.
Pos: 226:20:

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.

more

Pos: 354:8:

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.

more

Pos: 355:8:

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.

more

Pos: 372:8:

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.

more

Pos: 391:8:

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.

more

Pos: 431:8:

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.

more

Pos: 432:8:

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.

more

Pos: 46:6:

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.

more

Pos: 51:6:

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.

more

Pos: 168:8:

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.

more

Pos: 183:8:

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.

more

Pos: 207:8:

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.

more

Pos: 225:8:

Similar variable names:

BEP20._mint(address,uint256) : Variables have very similar names "account" and "amount". Note: Modifiers are currently not considered by this static analysis.
Pos: 375:39:

Similar variable names:

BEP20._mint(address,uint256) : Variables have very similar names "account" and "amount". Note: Modifiers are currently not considered by this static analysis.
Pos: 375:52:

Similar variable names:

BEP20._mint(address,uint256) : Variables have very similar names "account" and "amount". Note: Modifiers are currently not considered by this static analysis.
Pos: 376:34:

Similar variable names:

BEP20._mint(address,uint256) : Variables have very similar names "account" and "amount". Note: Modifiers are currently not considered by this static analysis.
Pos: 376:43:

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 24:8:

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 32:8:

Gas costs:

Gas requirement of function HarmonyLauncher.transfer is infinite:
If the gas requirement of a function is higher than the block gas limit, it cannot be executed.
Please avoid loops in your functions or actions that modify large areas of storage
(this includes clearing or copying arrays in storage)
Pos: 464:4:

Gas costs:

Gas requirement of function HarmonyLauncher.transferFrom is infinite:
If the gas requirement of a function is higher than the block gas limit, it cannot be executed.
Please avoid loops in your functions or actions that modify large areas of storage
(this includes clearing or copying arrays in storage)
Pos: 486:4:

Gas costs:

Gas requirement of function HarmonyLauncher.burn is infinite:
If the gas requirement of a function is higher than the block gas limit, it cannot be executed.
Please avoid loops in your functions or actions that modify large areas of storage
(this includes clearing or copying arrays in storage)
Pos: 494:4:

Miscellaneous

Similar variable names:

BEP20._mint(address,uint256) : Variables have very similar names "account" and "amount". Note: Modifiers are currently not considered by this static analysis.
Pos: 372:16:

Similar variable names:

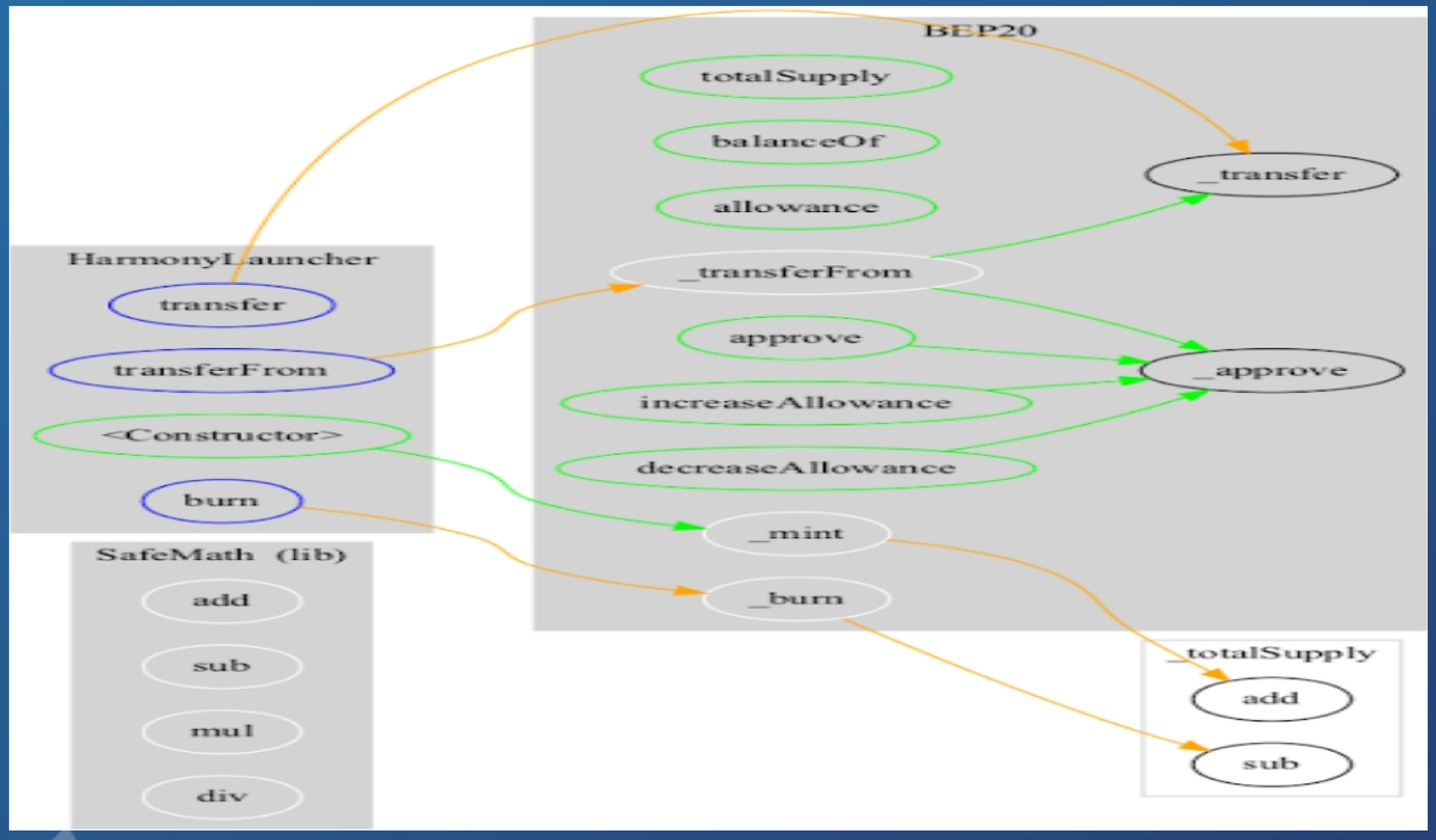
BEP20._mint(address,uint256) : Variables have very similar names "account" and "amount". Note: Modifiers are currently not considered by this static analysis.
Pos: 374:40:

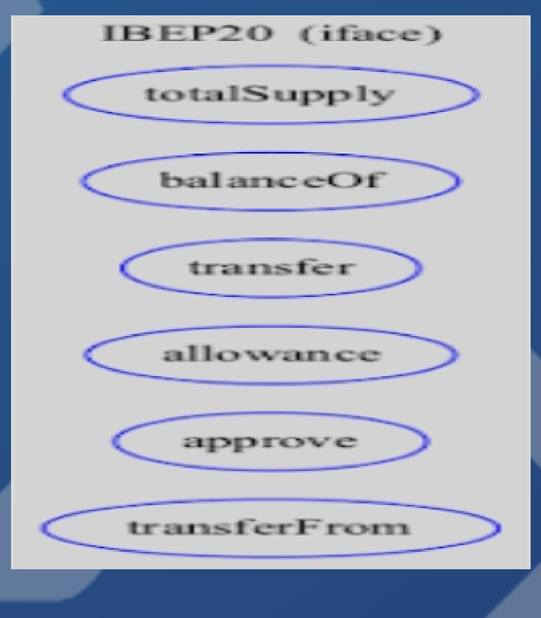
Similar variable names:

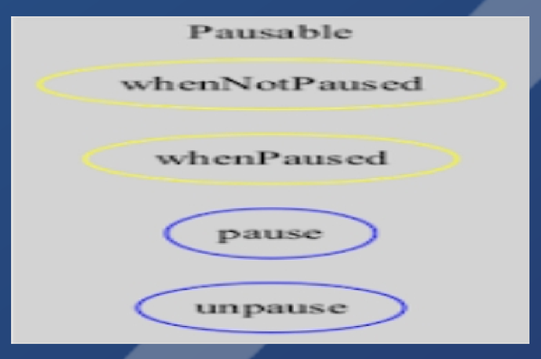
BEP20._mint(address,uint256) : Variables have very similar names "account" and "amount". Note: Modifiers are currently not considered by this static analysis.
Pos: 375:18:

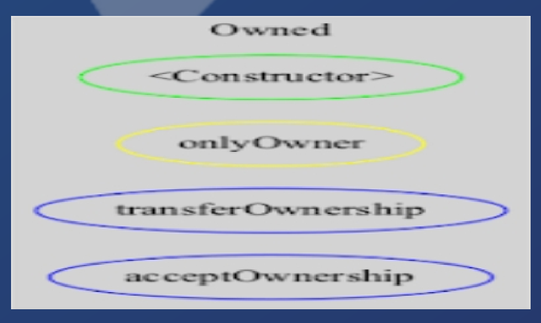
Sūrya


Legend
Internal Call
External Call
Defined Contract
Undefined Contract


BEP20
totalSupply
balanceOf
allowance
_transfer
_transferFrom
approve
_approve
increaseAllowance
decreaseAllowance
_mint
_burn
_totalSupply
add
sub
HarmonyLauncher
transfer
transferFrom
<Constructor>
burn
SafeMath (lib)
add
sub
mul
div


IBEP20 (iface)
totalSupply
balanceOf
transfer
allowance
approve
transferFrom


Pausable
whenNotPaused
whenPaused
pause
unpause


Owned
<Constructor>
onlyOwner
transferOwnership
acceptOwnership

Results

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.

Closing Summary

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

The audit showed an informational severity issue. Some changes were proposed to follow best practices and reduce potential attack surface. We highly recommend addressing them.

Disclaimer

Quillhash audit is not a security warranty, investment advice, or an endorsement of the HarmonyLauncher 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 HarmonyLauncher Team put in place a bug bounty program to encourage further analysis of the smart contract by other third parties.

Audit Report
February, 2022

For

HarmonyLauncher

QuillAudits

Canada, India, Singapore, United Kingdom

audits.quillhash.com

audits@quillhash.com