SPI Staking

SMART CONTRACT AUDIT

01.05.2021

Made in Germany by Chainsulting.de

Table of contents

1. Disclaimer

The audit makes no statements or warrantees about utility of the code, safety of the code, suitability of the business model, investment advice, endorsement of the platform or its products, regulatory regime for the business model, or any other statements about fitness of the contracts to purpose, or their bug free status. The audit documentation is for discussion purposes only.

The information presented in this report is confidential and privileged. If you are reading this report, you agree to keep it confidential, not to copy, disclose or disseminate without the agreement of AZ EXPRESS RETAIL LLC. If you are not the intended receptor of this document, remember that any disclosure, copying or dissemination of it is forbidden.

Major Versions / Date	Description
0.1 (28.04.2021)	Layout
0.5 (29.04.2021)	Automated Security Testing Manual Security Testing
0.7 (29.04.2021)	Test Deployment
0.8 (29.04.2021)	Verify Claims
0.9 (29.04.2021)	Summary and Recommendation
1.0 (30.04.2021)	Final Document
1.1 (01.05.2021)	Added deployed contract

2. About the Project and Company

Company address:

AZ EXPRESS RETAIL LLC
4281 EXPRESS LN
SARASOTA
FLORIDA, 34249
United States

Website: https://shopping.io

Instagram: https://www.instagram.com/shopping.io_official/

Twitter: https://twitter.com/shopping_io

Discord: https://discord.gg/36xNXa6

Telegram: https://t.me/shoppingio

Facebook: https://www.facebook.com/shopping.io/

2.1 Project Overview

Shopping.io was established as of December 2020. They are founded by dropshipping veterans with a vision to change how we make purchases with crypto. Shopping.io allows users to purchase goods directly from some of the leading ecommerce giants using over 100 different cryptocurrencies. To avail of this facility, all one has to do is sign up on Shopping.io by entering their email address and setting up the desired password. Once the account is created, they get access to a personal dashboard from which they can start searching and ordering stuff from the likes of Amazon, Walmart, eBay and more. The entire process from scratch takes no longer than a few minutes. Apart from the convenience of making payment in cryptocurrencies, customers on Shopping.io also stand to enjoy additional discounts and freedom from miscellaneous charges like shipping charges, sales tax, and more.

Shopping.io is in the constant process of developing and introducing new features for the cryptocurrency community. Some of the upcoming developments include the launch of its own token that offers an additional 5% discount and international shipping service to token holders. The platform will also be introducing new membership tiers after the conclusion of the limited period 10% discount offer on Free accounts. While users will still be able to purchase goods with cryptocurrencies using a free account, the discounts will be limited to Starter and Pro accounts at 5% and 10% respectively. All products shopped on Shopping.io, irrespective of account type will be eligible for 2-day free delivery within the United States along with a return/refund policy applicable for a maximum of 30 days from the date of delivery. Shopping.io will also be expanding its support for other leading ecommerce platforms including the upcoming AliExpress integration.

3. Vulnerability & Risk Level

Risk represents the probability that a certain source-threat will exploit vulnerability, and the impact of that event on the organization or system. Risk Level is computed based on CVSS version 3.0.

Level	Value	Vulnerability	Risk (Required Action)
Critical	9 – 10	A vulnerability that can disrupt the contract functioning in a number of scenarios, or creates a risk that the contract may be broken.	Immediate action to reduce risk level.
High	7 – 8.9	A vulnerability that affects the desired outcome when using a contract, or provides the opportunity to use a contract in an unintended way.	Implementation of corrective actions as soon as possible.
Medium	4 – 6.9	A vulnerability that could affect the desired outcome of executing the contract in a specific scenario.	Implementation of corrective actions in a certain period.
Low	2 – 3.9	A vulnerability that does not have a significant impact on possible scenarios for the use of the contract and is probably subjective.	Implementation of certain corrective actions or accepting the risk.
Informational	0 – 1.9	A vulnerability that have informational character but is not effecting any of the code.	An observation that does not determine a level of risk

4. Auditing Strategy and Techniques Applied

Throughout the review process, care was taken to evaluate the repository for security-related issues, code quality, and adherence to specification and best practices. To do so, reviewed line-by-line by our team of expert pentesters and smart contract developers, documenting any issues as there were discovered.

4.1 Methodology

The auditing process follows a routine series of steps:

1. Code review that includes the following:
   i. Review of the specifications, sources, and instructions provided to Chainsulting to make sure we understand the size, scope, and functionality of the smart contract.
   ii. Manual review of code, which is the process of reading source code line-by-line in an attempt to identify potential vulnerabilities.
   iii. Comparison to specification, which is the process of checking whether the code does what the specifications, sources, and instructions provided to Chainsulting describe.
2. Testing and automated analysis that includes the following:
   i. Test coverage analysis, which is the process of determining whether the test cases are actually covering the code and how much code is exercised when we run those test cases.
   ii. Symbolic execution, which is analysing a program to determine what inputs causes each part of a program to execute.
3. Best practices review, which is a review of the smart contracts to improve efficiency, effectiveness, clarify, maintainability, security, and control based on the established industry and academic practices, recommendations, and research.
4. Specific, itemized, actionable recommendations to help you take steps to secure your smart contracts.

4.2 Used Code from other Frameworks/Smart Contracts

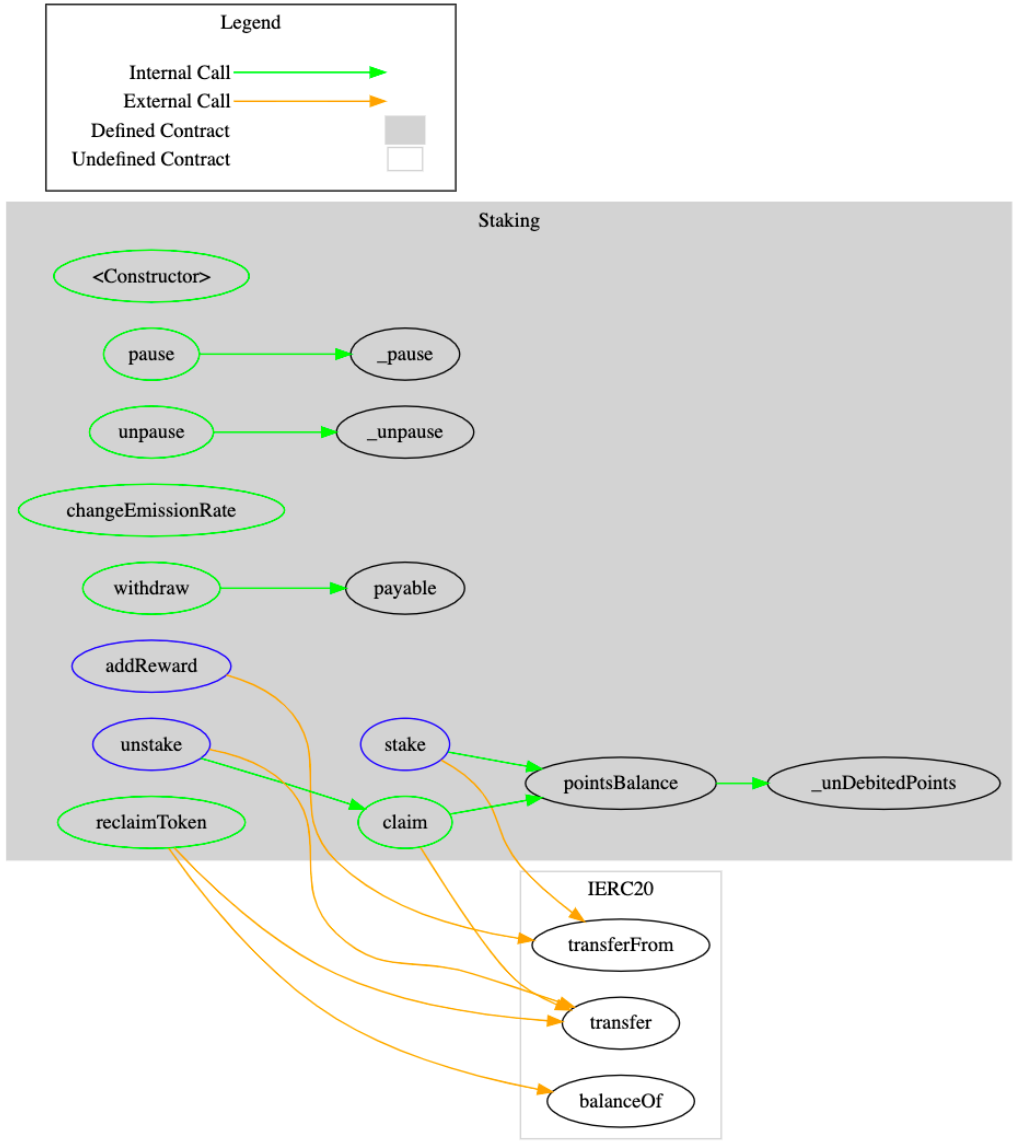
1. SafeMath.sol
https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts/math/SafeMath.sol
2. Pausable.sol
https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts/security/Pausable.sol
3. IERC20.sol
https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts/token/ERC20/IERC20.sol
4. Ownable.sol
https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts/access/Ownable.sol
5. ReentrancyGuard.sol
https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts/security/ReentrancyGuard.sol
6. IERC1155.sol
https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts/token/ERC1155/IERC1155.sol

4.3 Tested Contract Files

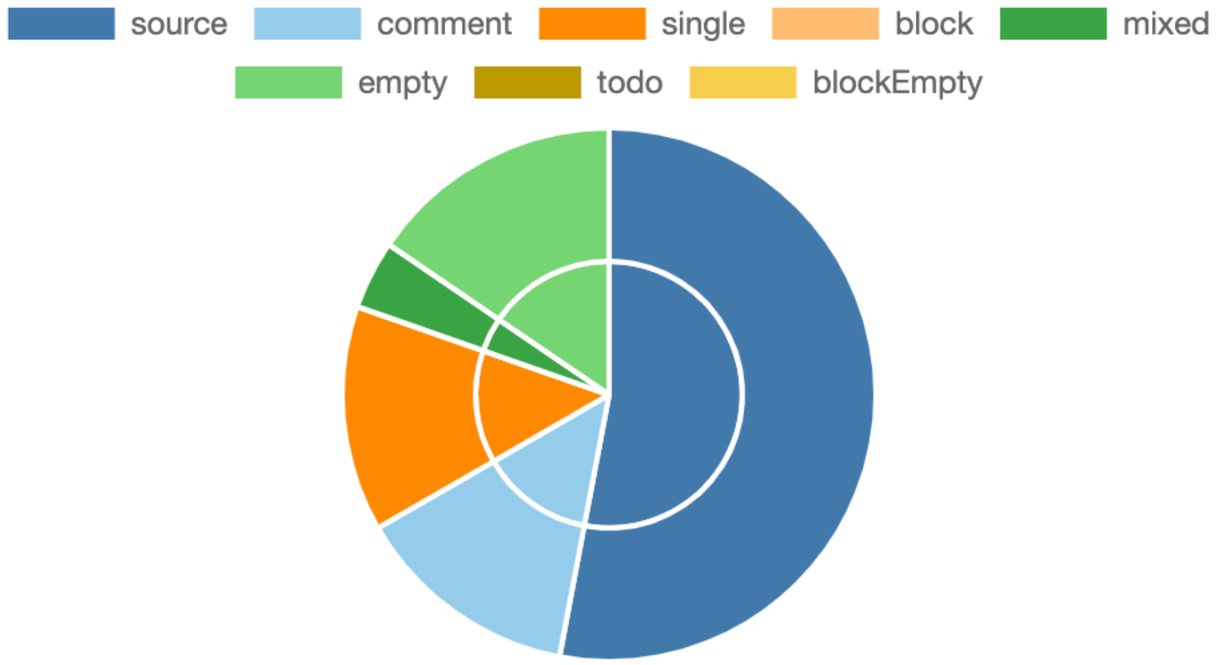
The following are the SHA-256 hashes of the reviewed files. A file with a different SHA-256 hash has been modified, intentionally or otherwise, after the security review. You are cautioned that a different SHA-256 hash could be (but is not necessarily) an indication of a changed condition or potential vulnerability that was not within the scope of the review

File	Fingerprint (SHA256)
spi_staking.sol	73840710d35ebc2b5f5a5fbc707e2b20

4.4 Metrics / CallGraph



4.5 Metrics / Source Lines



4.6 Metrics / Capabilities

Solidity Versions observed	🧪 Experimental Features	💰 Can Receive Funds	🖥 Uses Assembly	💣 Has Destroyable Contracts
`^0.8.3`			**** (0 asm blocks)	

📤 Transfers ETH	⚡ Low-Level Calls	👥 DelegateCall	🧮 Uses Hash Functions	🔖 ECRecover	🌀 New/Create/Create2
`yes`					

🌐Public	💰Payable
10	0

External	Internal	Private	Pure	View
3	11	0	0	2

StateVariables

Total	🌐Public
6	5

4.7 Metrics / Source Unites in Scope

Type	File	Logic Contracts	Interfaces	Lines	nLines	nSLOC	Comment Lines	Complex. Score	Capabilities
📝	spi_staking.sol	1		131	131	89	23	80	📤
📝	**Totals**	**1**		**131**	**131**	**89**	**23**	**80**	📤

Legend: [➖]

- **Lines**: total lines of the source unit
- **nLines**: normalized lines of the source unit (e.g. normalizes functions spanning multiple lines)
- **nSLOC**: normalized source lines of code (only source-code lines; no comments, no blank lines)
- **Comment Lines**: lines containing single or block comments
- **Complexity Score**: a custom complexity score derived from code statements that are known to introduce code complexity (branches, loops, calls, external interfaces, ...)

5. Scope of Work

The Shopping.io Team provided us with the file that needs to be tested. The scope of the audit is the SPI Staking contract.

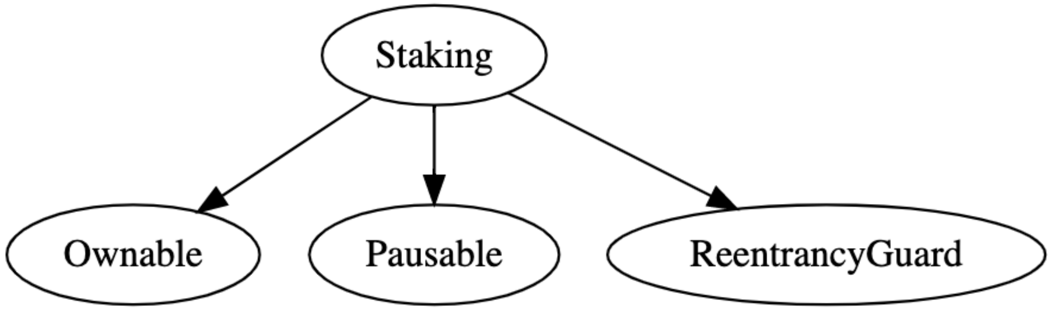
Following contracts with the direct imports has been tested:

- spi_staking.sol

The team put forward the following assumptions regarding the security, usage of the contracts:

- The user can stake SPI at any time.
- Contract deployer or owner is not able to withdraw staked SPI or rewards
- The user is not able to withdraw reward or staked SPI before the lockTime ends, but after the lockTime ends.
- The emissionRate is correctly calculated
- If the owner pause the contract, no one can stake SPI / unstake or withdraw rewards
- User can't stake less than minimumAmount or more than maximumAmount
- Mathematical operations within the contract are safe
- Check the overall smart contract security and functions

The main goal of this audit was to verify these claims. The auditors can provide additional feedback on the code upon the client's request.



5.1 Manual and Automated Vulnerability Test

CRITICAL ISSUES

During the audit, Chainsulting's experts found **no Critical issues** in the code of the smart contract.

HIGH ISSUES

During the audit, Chainsulting's experts found **no High issues** in the code of the smart contract.

MEDIUM ISSUES

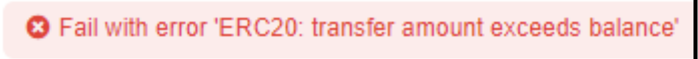
During the audit, Chainsulting's experts found **no Medium issues** in the code of the smart contract.

LOW ISSUES

5.1.1 Locking scenario

Severity: LOW
Status: ACKNOWLEDGED
Update: https://etherscan.io/address/0x5dFE942a8B781ACdfe06EE55DAEf06B9FaB86aF8#readContract
Owner of the contract got verified as the founder of Shopping.io. It's in the best interest of the project to provide rewards and guarantee the pay-outs.
File(s) affected: All

Attack / Description	Code Snippet	Result/Recommendation
If there is no reward sent to the contract with the addReward function, the contract runs out of balance and locks user's staked SPI. There is no function to check if the contract has balance to pay-out rewards. There is no	`Stake Tokens:` https://kovan.etherscan.io/tx/0x7a2b8b5294188d245927db358185e8308a520b44d6495e1967a0db099416c611 `Claim Rewards:` https://kovan.etherscan.io/tx/0x7634ff012bdfb90b501e47a760d60805ad152171c61f70d7213627d2c60a3844	Only allow staking, if the balance of the contract can guarantee the pay-out of rewards and stakes after locking time.

guarantee for getting back staked tokens.	`Cannot unstake anymore until someone sends tokens to contract:`  ❌ Fail with error 'ERC20: transfer amount exceeds balance' https://kovan.etherscan.io/tx/0x7f8ad2730248e73f0b1590143af7aec51c486e2bde0d66f449346ff8d8875577	

INFORMATIONAL ISSUES

5.1.2 A floating pragma is set

Severity: INFORMATIONAL
Status: ACKNOWLEDGED
File(s) affected: All

Attack / Description	Code Snippet	Result/Recommendation
The current pragma Solidity directive is ^0.8.3; It is recommended to specify a fixed version to ensure that the bytecode produced does not vary between builds. This is especially important if you rely on bytecode-level verification of the code.	Line 1 `pragma solidity ^0.8.3;`	It is recommended to follow the example(0.8.3), as future compiler versions may handle certain language constructions in a way the developer did need foresee. Not effecting the overall contract functionality.

5.1.3 Event not emitted
Severity: INFORMATIONAL
Status: ACKNOWLEDGED
File(s) affected: All

Attack / Description	Code Snippet	Result/Recommendation
The *StakeClaimed* event is defined but never emitted.	Line 34 event StakeClaimed(address user, uint256 amount);	We recommend emitting the event after claiming stake in the *unstake* function. Otherwise remove the unused event.

6. Test Deployment

6.0.1 Deployment of ERC20-Token as test token (SPIX)
Tx: https://kovan.etherscan.io/tx/0xfc3d13f343f6eeb6e03e3e814c971b19925d1a42b6ff0a9e7d243cc9be4abc01
Contract: https://kovan.etherscan.io/address/0xd2feb940bafd1ba55fce130aa0e95ca0ecc2f720

6.0.2 Deployment of SPI staking contract
Tx: https://kovan.etherscan.io/tx/0x9c5f0d39ac16b51cb3806bac87cf284822b6baf10a93d01dacc193f0f5cdecc1
Contract: https://kovan.etherscan.io/address/0xb1020a4b06fdba0c8dff52644f015e6ea3993af1

6.1 Verify Claims

6.1.1 The user can stake SPI at any time.

The users are able to stake at any time if the contract is not paused. The owner can pause the contract at any time and thus disable staking (see 2.5.).

```
    function stake(uint256 _amount) external whenNotPaused nonReentrant {
```

Tx: https://kovan.etherscan.io/tx/0x9a54275ecc4c1e6fbfb0f1929a7bf54afc48d08ff390acd0eb926221e44a94cb

6.1.2 Contract deployer or owner is not able to withdraw staked SPI or rewards

There is no fuctionality in the contract, which allows the owner of the contract to withdraw or claim users staked funds and rewards. The owner can only withdraw ether or other erc20 tokens (except staking token) sent to the contract.

6.1.3 The user is not able to withdraw reward or staked SPI before the lockTime ends, but after the lockTime ends.

Before the lockTime ends, the user is not able to unstake tokens. However it is possible to claim the rewards before the lockTime ends by calling *claim* function. The *claim* function has no timelock and can be called by anyone at any time.

```
        require(user.createdAt + lockTime <= block.timestamp, "tokens are locked");
```

Claim before lock ends:
Tx: https://kovan.etherscan.io/tx/0x200a97f6d01f565bd6108ad1869d3a40e606b1d0481cc9fdbfd27adcc9048631
Unstake before lock ends:
Tx: https://kovan.etherscan.io/tx/0x0ad7ab2db1336c867990e9d8b6166512306eba903bf26312308ee5648fc81731

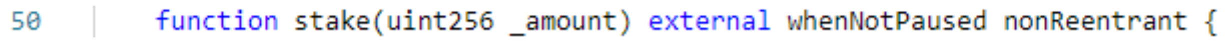
6.1.4 The emissionRate is correctly calculated

The emissionRate can be set and changed by the owner at any time. The rewards are properly calculated with the following formula:

```
    //calculates the undebitted points. (seconds since staked) X emission rate X amount / 10^18
    function _unDebitedPoints(UserInfo memory user) internal view returns (uint256) {
        return block.timestamp.sub(user.lastUpdateAt).mul(emissionRate).mul(user.amount).div(1e18);
    }
```

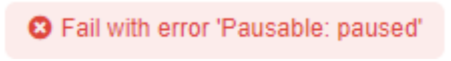
6.1.5 If the owner pause the contract, no one can stake SPI / unstake or withdraw rewards

The staking function is locked while the contract is paused. However unstaking and withdrawing rewards is still possible if the contract is paused.
Staking is locked by whenNotPaused modifier:



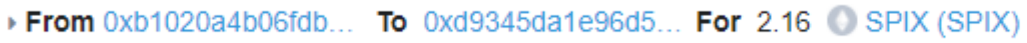
```
    function stake(uint256 _amount) external whenNotPaused nonReentrant {
```

Pause contract:
Tx: https://kovan.etherscan.io/tx/0x42e2b9af62f3b0fe25e7bb15002e5c9b0e6fdf122c129bd941c31ec51acca6a4
Staking fails as expected:



Fail with error 'Pausable: paused'

Tx: https://kovan.etherscan.io/tx/0xb841f36da0744a8e50f99bcac006bebe0b1fde641ea68224e8a57dab033ac14e
Claim rewards and unstaking is still possible:

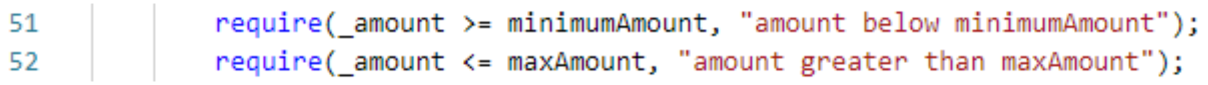


From 0xb1020a4b06fdb... **To** 0xd9345da1e96d5... **For** 2.16 SPIX (SPIX)

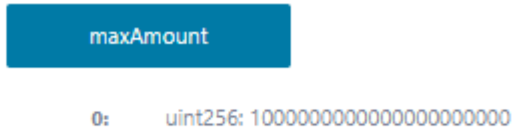
Tx: https://kovan.etherscan.io/tx/0x821b2f901c141c6781f06a4cf5dae86afb02a0df623a26b52e8ca4dbb928ec5a

6.1.6 User can't stake less than minimumAmount or more than maximumAmount

The staking function is locked by the following requirements and thus staking less than minAmount or more than maxAmount at one time is not possible. However the user can stake multiple times the maxAmount. This means staking is not limited, only the staking for one transaction.

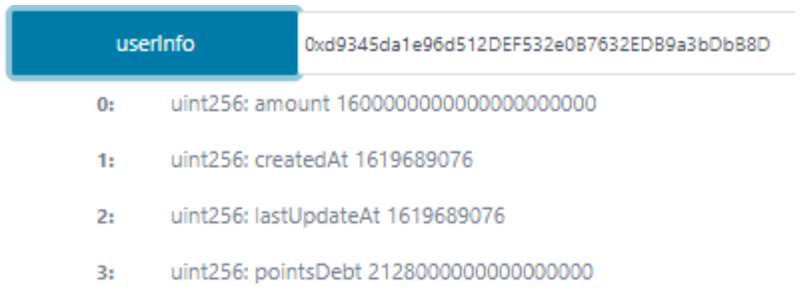


```
        require(_amount >= minimumAmount, "amount below minimumAmount");
        require(_amount <= maxAmount, "amount greater than maxAmount");
```



0: uint256: 1000000000000000000000

Stake first time: https://kovan.etherscan.io/tx/0x9a54275ecc4c1e6fbfb0f1929a7bf54afc48d08ff390acd0eb926221e44a94cb
Stake second time (maxAmount):
https://kovan.etherscan.io/tx/0x7a2b8b5294188d245927db358185e8308a520b44d6495e1967a0db099416c611
Total stake more than maxAmount:



```
userInfo    0xd9345da1e96d512DEF532e0B7632EDB9a3bDbB8D
0:    uint256: amount 1600000000000000000000
1:    uint256: createdAt 1619689076
2:    uint256: lastUpdateAt 1619689076
3:    uint256: pointsDebt 2128000000000000000
```

6.1.7 Mathematical operations within the contract are safe

The contract uses SafeMath library from OpenZeppelin, which makes mathematical calculations secure.

7. Executive Summary

Two (2) independent Chainsulting experts performed an unbiased and isolated audit of the smart contract codebase. The final debriefs took place on the April 30, 2021. The code is not overloaded with unnecessary functions; these is greatly benefiting the security of the contract. It correctly implemented widely used and reviewed contracts from OpenZeppelin and for safe mathematical operations.

The main goal of the audit was to verify the claims regarding the security of the smart contract and the claims inside the scope of work. During the audit, no critical issues were found after the manual and automated security testing. Only informational issues were found, to increase the code quality and one low issue, which can lead to a problem at runtime.

8. Deployed Smart Contract

VERIFIED

SPI Staking
https://etherscan.io/address/0x5dFE942a8B781ACdfe06EE55DAEf06B9FaB86aF8#code