AUDIT REPORT

PolySafe

Table Of Content

Disclaimer

This report is based on the scope of materials and documentation provided for a limited review at the time provided. Results may not be complete nor inclusive of all vulnerabilities. The review and this report are provided on an as-is, where-is, and as-available basis. You agree that your access and/or use, including but not limited to any associated services, products, protocols, platforms, content, and materials, will be at your sole risk. Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty. The Solidity language itself and other smart contract languages remain under development and are subject to unknown risks and flaws. The review does not extend to the compiler layer, or any other areas beyond Solidity or the smart contract programming language, or other programming aspects that could present security risks. You may risk loss of tokens, Ether, and/or other loss. A report is not an endorsement (or other opinion) of any particular project or team, and the report does not guarantee the security of any particular project. A report does not consider, and should not be interpreted as considering or having any bearing on, the potential economics of a token, token sale or any other product, service or other asset. No third party should rely on the reports in any way, including for the purpose of making any decisions to buy or sell any token, product, service or other asset. To the fullest extent permitted by law, we disclaim all warranties, express or implied, in connection with this report, its content, and the related services and products and your use thereof, including, without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement. We do not warrant, endorse, guarantee, or assume responsibility for any product or service advertised or offered by a third party through the product, any open source or third party software, code, libraries, materials, or information linked to, called by, referenced by or accessible through the report, its content, and the related services and products, any hyperlinked website, or any website or mobile application featured in any banner or other advertising, and we will not be a party to or in any way be responsible for monitoring any transaction between you and any third-party providers of products or services. As with the purchase or use of a product or service through any medium or in any environment, you should use your best judgment and exercise caution where appropriate. You may risk loss of QSP tokens or other loss.

Technical Note: For avoidance of doubt, the report, its content, access, and/or usage thereof, including any associated services or materials, shall not be considered or relied upon as any form of financial, investment, tax, legal, regulatory, or other advice.

Overview

Project Summary

Project Name	PolySafe Protocol
Description	PolySafe (PS) allows users to stake MATIC and its native token, PS to earn daily rewards in PS and users can sell it for MATIC.
Platform	Polygon; Solidity
Codebase	Verified Smart Contract

Audit Summary

Delivery Date	Sept. 25th, 2021
Method of Audit	Static Analysis, Manual Review
Consultants Engaged	1
Timeline	Sept. 21, 2021 - Sept. 22, 2021 & Sept. 24, 2021 - Sept. 25, 2021.

Vulnerability Summary

Total Issues	2
Total Critical	0
Total Major	0
Total Minor	1
Total Informational	1

Introduction

PolyStake (powered by PolySafe)

- PolyStake is a platform that allows users to deposit MATIC to get PolySafe Token using which multiple actions can be performed.
- PolySafe tokens can be staked without any limit on this platform for users to earn more reward.
- This also includes a referral model which allows users to earn up to 3 levels with 2%, 1% and 1% for level 1,2,3 respectively.
- It also consists of AIRDROPS that can be claimed by completing the challenges listed which also includes 3 side games which can be played like Mini Lottery, Simple Duel and Simple Bet.

Tokenomics:

Token Supply: 1,000,000 (Fixed)

Fee Structure: 4% Marketing (from MATIC deposits)
4% Administration Social Network (from Staking)
4% Dev

Airdrop: 100 $PS weekly (Max. airdrop value 100,000 $PS)

Lock Period After Staking: 7 days

Findings

ID	Title	Severity
ISSUE: 01	Addresses not verified in functions	Low
ISSUE: 02	Function should be declared external	Informational

Issue-01: Addresses Not Verified In Functions

Severity: Low

Description:

These functions in the PolySafe contract do not verify if the supplied address input is non-0.

Locations:

PolySafe: 268
PolySafe: 346
PolySafe: 355
PolySafe: 371
PolySafe: 380
PolySafe: 469
PolySafe: 474

Recommendation:

Consider adding a requirement to the function in order to verify that the supplied address is non-0.

Example:

```
require(
        input_address != address(0),
        "invalid address"
);
```


Issue-02: Function Should Be Declared External

Severity: Informational

Description:

The following functions in the PolySafe contract should be declared external, as it is not used from within the contract itself and would reduce gas consumption by using the arguments from callData instead of allocating memory space.

Location:

PolySafe: 268 stakeMatic
PolySafe: 319 stakeToken
PolySafe: 336 unStakeToken
PolySafe: 396 claimToken_M
PolySafe: 414 claimToken_T
PolySafe: 418 sellToken
PolySafe: 437 claimAirdrop
PolySafe: 450 claimAirdropM
PolySafe: 460 withdrawRef
PolySafe: 469 getUserUnclaimedTokens_M
PolySafe: 474 getUserUnclaimedTokens_T
PolySafe: 483 getUserTimeToNextAirdrop
PolySafe: 487 getUserBonAirdrop
PolySafe: 496 getUserCountAirdrop
PolySafe: 504 getContractTokenBalance
PolySafe: 508 getAPY_T
PolySafe: 516 getUserMaticBalance
PolySafe: 520 getUserTokenBalance
PolySafe: 524 getUserMaticStaked
PolySafe: 528 getUserTokenStaked
PolySafe: 532 getUserTimeToUnstake
PolySafe: 536 getTokenPrice
PolySafe: 542 maticToToken
PolySafe: 550 getUserDownlineCount
PolySafe: 554 getUserReferralBonus
PolySafe: 562 getUserReferralWithdrawn
PolySafe: 566 getContractLaunchTime
PolySafe: 574 getTokenSoldToday
PolySafe: 578 getTokenAvailableToSell
PolySafe: 582 getTimeToNextDay

Current:

function function_name **public** {}

Recommendation:

Refactor the visibility of these functions from public to external function function_name **external** {}

Conclusion

Smart contracts within the scope were manually reviewed and analyzed with a static analysis mechanism. For the contract, a high-level description of functionality was presented in the introduction section of the report. Audit report contains all found security vulnerabilities and other issues in the reviewed code. Overall quality of reviewed contracts with respect to security is good. We found 2 low/informational level vulnerabilities, which do not have any significant security impact. The contract is safe and does not have any backdoors.

Smart Contract Flow Diagram:

https://ipfs.io/ipfs/QmV6RKKi5YRYmL2QjmBqKfY3TRnGp6DZMQBDCXDXXTjZ2z

Appendix

Finding Categories:

Gas Optimization

Gas Optimization refers to exhibits that do not affect the functionality but generate different, optimal EVM opcodes resulting in a reduction on the total gas cost of a transaction.

Arithmetic

Arithmetic exhibits entail findings that relate to mishandling of math formulas, such as overflows, incorrect operations etc.

Logical Issue

Logical Issue findings are exhibits that detail a fault in the logic of the linked code, such as an incorrect notion on how block timestamp works.

Control Flow

Control Flow findings concern the access control imposed on functions, such as owner-only functions being invoke-able by anyone under certain circumstances.

Volatile Code

Volatile Code findings refer to segments of code that behave unexpectedly on certain edge cases that may result in a vulnerability.

Data Flow

Data Flow checks ensure covering faults where data is handled at rest and in memory, such as the result of a struct assignment operation affecting an in-memory struct rather than an in-storage one.

Coding Style

Coding Style findings usually do not affect the generated byte-code and comment on how to make the codebase more legible and as a result easily maintainable.

Inconsistency

Inconsistency findings refer to functions that should seemingly behave similarly yet contain different code, such as a constructor assignment imposing different require statements on the input variables than a setter function.

Magic Numbers

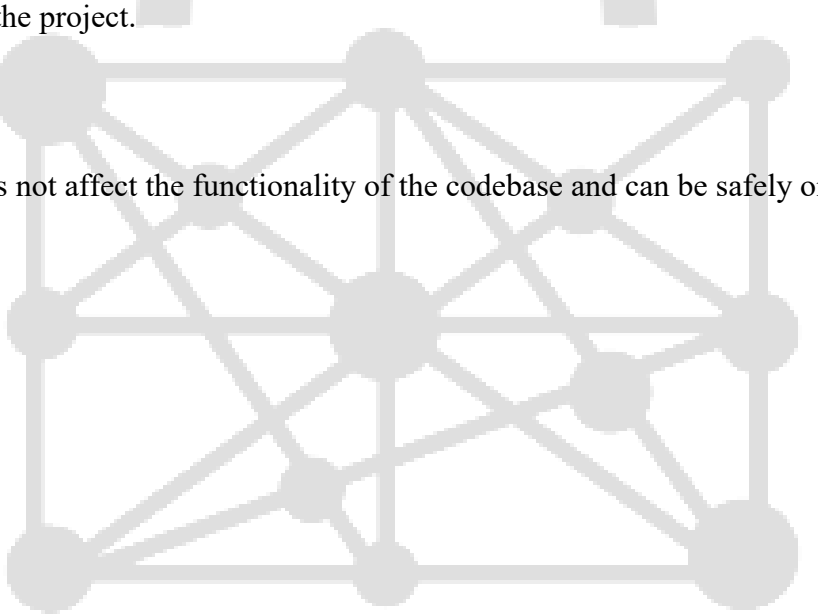
Magic Number findings refer to numeric literals that are expressed in the codebase in their raw format and should otherwise be specified as constant contract variables aiding in their legibility and maintainability.

Compiler Error

Compiler Error findings refer to an error in the structure of the code that renders it impossible to compile using the specified version of the project.

Dead Code

Code that otherwise does not affect the functionality of the codebase and can be safely omitted.



About Us

Only those who are aware of the threats can build an indestructible shield, thus the DeFI Soldity team has professional cyber security team with wide range of experience in blockchain development and security.

We provide a wide range of cybersecurity services for businesses operating in the digital world. Our specialists design best solutions, focusing on the needs of the client.

We will help you protect your corporate security and allow you to receive actionable recommendations on how to eliminate the vulnerabilities discovered in your system. Our specialists will recommend potentially required patches, code, and access changes, as well as other adjustments.

Telegram Channel: @defi_solidity | Twitter: @defisolidity