TechRate

Blockchain solutions and consulting

Smart Contract Security Audit

Audit details:

Audited project:	PolyShark
Deployer address	0xA658F3689388556B6CFA5Fb99969a568e452a352
Blockchain:	Matic
Project website:	Not provided

May, 2021

TechRate

Disclaimer

This is a limited report on our findings based on our analysis, in accordance with good industry practice as at the date of this report, in relation to cybersecurity vulnerabilities and issues in the framework and algorithms based on smart contracts, the details of which are set out in this report. In order to get a full view of our analysis, it is crucial for you to read the full report. While we have done our best in conducting our analysis and producing this report, it is important to note that you should not rely on this report and cannot claim against us on the basis of what it says or doesn't say, or how we produced it, and it is important for you to conduct your own independent investigations before making any decisions. We go into more detail on this in the below disclaimer below – please make sure to read it in full.

DISCLAIMER: By reading this report or any part of it, you agree to the terms of this disclaimer. If you do not agree to the terms, then please immediately cease reading this report, and delete and destroy any and all copies of this report downloaded and/or printed by you. This report is provided for information purposes only and on a non-reliance basis, and does not constitute investment advice. No one shall have any right to rely on the report or its contents, and TechRate and its affiliates (including holding companies, shareholders, subsidiaries, employees, directors, officers and other representatives) (TechRate) owe no duty of care towards you or any other person, nor does TechRate make any warranty or representation to any person on the accuracy or completeness of the report. The report is provided "as is", without any conditions, warranties or other terms of any kind except as set out in this disclaimer, and TechRate hereby excludes all representations, warranties, conditions and other terms (including, without limitation, the warranties implied by law of satisfactory quality, fitness for purpose and the use of reasonable care and skill) which, but for this clause, might have effect in relation to the report. Except and only to the extent that it is prohibited by law, TechRate hereby excludes all liability and responsibility, and neither you nor any other person shall have any claim against TechRate, for any amount or kind of loss or damage that may result to you or any other person (including without limitation, any direct, indirect, special, punitive, consequential or pure economic loss or damages, or any loss of income, profits, goodwill, data, contracts, use of money, or business interruption, and whether in delict, tort (including without limitation negligence), contract, breach of statutory duty, misrepresentation (whether innocent or negligent) or otherwise under any claim of any nature whatsoever in any jurisdiction) in any way arising from or connected with this report and the use, inability to use or the results of use of this report, and any reliance on this report.

The analysis of the security is purely based on the smart contracts alone. No applications or operations were reviewed for security. No product code has been reviewed.

Background

TechRate was commissioned by PolyShark to perform an audit of smart contracts:

- *https://explorer-mainnet.maticvigil.com/address/0xD201B8511aaB3E9b094b35ABcD5d7863c78D6d0e/contracts*
- *https://explorer-mainnet.maticvigil.com/address/0x1414529422AdD5a715C3dE929C32B2d88aE4A761/contracts*

The purpose of the audit was to achieve the following:

- Ensure that the smart contract functions as intended.
- Identify potential security issues with the smart contract.

The information in this report should be used to understand the risk exposure of the smart contract, and as a guide to improve the security posture of the smart contract by remediating the issues that were identified.

Contracts details

Token contract details for 10.05.2021.

Contract name:	PolyShark
Compiler version:	v0.6.12+commit.27d51765
Contract address:	0xD201B8511aaB3E9b094b35ABcD5d7863c78D6d0e
Total supply:	1000000000000000000000
Token ticker:	Shark
Decimals:	18
Token holders:	78
Transactions count:	263
Contract deployer address:	0xA658F3689388556B6CFA5Fb99969a568e452a352
Contract’s current owner address:	0x1414529422AdD5a715C3dE929C32B2d88aE4A761

PolyShark top 10 token holders

0xA658F3689388556B6CFA5Fb99969a568e452a352
400.825081726914702375 Shark 40.0825%

0x09F8D66377756cB3e86A73b3E6176C9939F3fF7C
400 Shark 40.0000%

0xb746Ed49B63f2239A8Ef33eE6D0F5Bb52eDa26Bc
71.401740409227229724 Shark 7.1402%

0x945b79FA31C424517494525474DC0Fc31F06243E
20.059897992267058488 Shark 2.0060%

0x1414529422AdD5a715C3dE929C32B2d88aE4A761
16.739338592017379973 Shark 1.6739%

0x3d40DF4adA235CadBB6B385486DC7c2Af77a952e
15.43 Shark 1.5430%

0x000000000000000000000000000000000000dEaD
13.277048389606122499 Shark 1.3277%

0x6c2ffD3923db08Cb93a0dcf117F4D108f7B8Ea6d
9.9 Shark 0.9900%

0xCaF76ca5cF79510F7928EEE79EfEB92372957c10
7.533235138228966146 Shark 0.7533%

0x568B5b0F64DDD153b7eFA4E7e6e02643C0253c9e
7.000058 Shark 0.7000%

MasterChef contract details for 10.05.2021.

Contract name:	MasterChef
Compiler version:	v0.6.12+commit.27d51765
Contract address:	0x1414529422AdD5a715C3dE929C32B2d88aE4A761
Deployer address:	0xA658F3689388556B6CFA5Fb99969a568e452a352
Dev address:	0xA658F3689388556B6CFA5Fb99969a568e452a352
Fee address:	0xA658F3689388556B6CFA5Fb99969a568e452a352
Shark contract address:	0xD201B8511aaB3E9b094b35ABcD5d7863c78D6d0e
Shark per block:	5000000000000000000
Contract owner address:	0xA658F3689388556B6CFA5Fb99969a568e452a352
Pool length:	9
Start block:	14341800
Total alloc point:	13500
Bonus multiplier:	1

MasterChef functions outline

+ [Lib] SafeMath
  - [Int] tryAdd
  - [Int] trySub
  - [Int] tryMul
  - [Int] tryDiv
  - [Int] tryMod
  - [Int] add
  - [Int] sub
  - [Int] mul
  - [Int] div
  - [Int] mod
  - [Int] sub
  - [Int] div
  - [Int] mod

+ Context
  - [Int] _msgSender
  - [Int] _msgData

+ Ownable (Context)
  - [Int] <Constructor> #
  - [Pub] owner
  - [Pub] renounceOwnership #
    - modifiers: onlyOwner
  - [Pub] transferOwnership #
    - modifiers: onlyOwner

+ ReentrancyGuard
  - [Int] <Constructor> #

+ [Lib] Address
  - [Int] isContract
  - [Int] sendValue #
  - [Int] functionCall #
  - [Int] functionCall #
  - [Int] functionCallWithValue #
  - [Int] functionCallWithValue #
  - [Int] functionStaticCall
  - [Int] functionStaticCall
  - [Int] functionDelegateCall #
  - [Int] functionDelegateCall #
  - [Prv] _verifyCallResult

+ [Int] IBEP20

- [Ext] totalSupply
- [Ext] decimals
- [Ext] symbol
- [Ext] name
- [Ext] getOwner
- [Ext] balanceOf
- [Ext] transfer #
- [Ext] allowance
- [Ext] approve #
- [Ext] transferFrom #

+ BEP20 (Context, IBEP20, Ownable)
  - [Pub] <Constructor> #
  - [Ext] getOwner
  - [Pub] name
  - [Pub] decimals
  - [Pub] symbol
  - [Pub] totalSupply
  - [Pub] balanceOf
  - [Pub] transfer #
  - [Pub] allowance
  - [Pub] approve #
  - [Pub] transferFrom #
  - [Pub] increaseAllowance #
  - [Pub] decreaseAllowance #
  - [Pub] mint #
    - modifiers: onlyOwner
  - [Int] _transfer #
  - [Int] _mint #
  - [Int] _burn #
  - [Int] _approve #
  - [Int] _burnFrom #

+ SharkToken (BEP20)
  - [Pub] mint #
    - modifiers: onlyOwner
  - [Int] _transfer #
  - [Ext] delegates
  - [Ext] delegate #
  - [Ext] delegateBySig #
  - [Ext] getCurrentVotes
  - [Ext] getPriorVotes
  - [Int] _delegate #
  - [Int] _moveDelegates #
  - [Int] _writeCheckpoint #
  - [Int] safe32
  - [Int] getChainId

```
+ [Lib] SafeBEP20
  - [Int] safeTransfer #
  - [Int] safeTransferFrom #
  - [Int] safeApprove #
  - [Int] safeIncreaseAllowance #
  - [Int] safeDecreaseAllowance #
  - [Prv] _callOptionalReturn #

+ [Int] ISharkReferral
  - [Ext] recordReferral #
  - [Ext] getReferrer

+  MasterChef (Ownable, ReentrancyGuard)
  - [Pub] <Constructor> #
  - [Ext] poolLength
  - [Pub] add #
    - modifiers: onlyOwner
  - [Pub] set #
    - modifiers: onlyOwner
  - [Pub] getMultiplier
  - [Ext] pendingShark
  - [Pub] massUpdatePools #
  - [Pub] updatePool #
  - [Pub] deposit #
    - modifiers: nonReentrant
  - [Pub] withdraw #
    - modifiers: nonReentrant
  - [Pub] emergencyWithdraw #
    - modifiers: nonReentrant
  - [Int] safeSharkTransfer #
  - [Pub] setDevAddress #
  - [Pub] setFeeAddress #
  - [Pub] updateEmissionRate #
  - [Pub] setSharkReferral #
    - modifiers: onlyOwner
  - [Pub] setReferralCommissionRate #
    - modifiers: onlyOwner
  - [Int] payReferralCommission #


($) = payable function
# = non-constant function
```

Issues Checking Status

№	Issue description.	Checking status
1	Compiler errors.	Passed
2	Race conditions and Reentrancy. Cross-function race conditions.	Passed
3	Possible delays in data delivery.	Passed
4	Oracle calls.	Passed
5	Front running.	Passed
6	Timestamp dependence.	Passed
7	Integer Overflow and Underflow.	Passed
8	DoS with Revert.	Passed
9	DoS with block gas limit.	Low issues
10	Methods execution permissions.	Passed
11	Economy model of the contract.	Passed
12	The impact of the exchange rate on the logic.	Passed
13	Private user data leaks.	Passed
14	Malicious Event log.	Passed
15	Scoping and Declarations.	Passed
16	Uninitialized storage pointers.	Passed
17	Arithmetic accuracy.	Passed
18	Design Logic.	Medium issues
19	Cross-function race conditions.	Passed
20	Safe Open Zeppelin contracts implementation and usage.	Passed
21	Fallback function security.	Passed

Security Issues

High Severity Issues

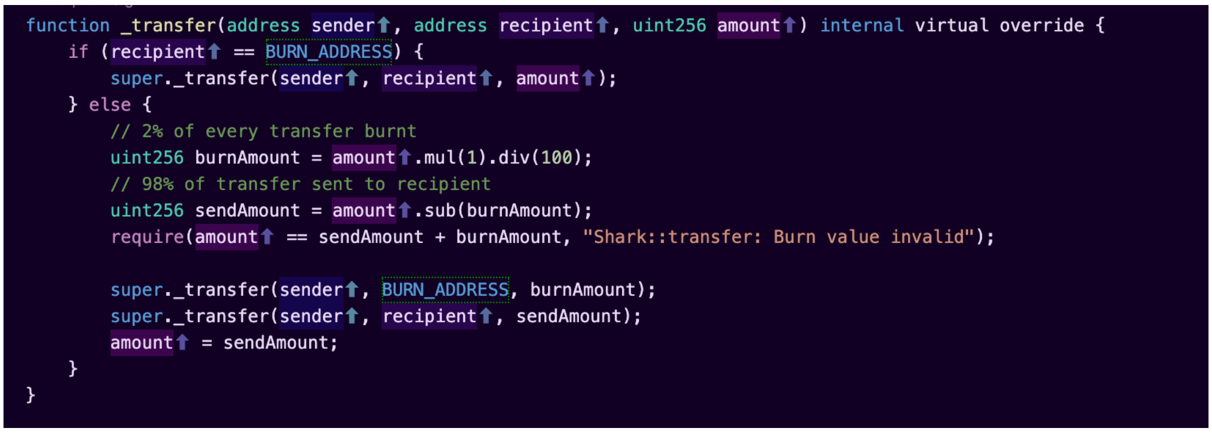
No high severity issues found.

Medium Severity Issues

1. Wrong burning

Issue:

There is sending tokens to the dead address in overridden _transfer functions, instead of burning them in the token contract.



```
function _transfer(address sender, address recipient, uint256 amount) internal virtual override {
    if (recipient == BURN_ADDRESS) {
        super._transfer(sender, recipient, amount);
    } else {
        // 2% of every transfer burnt
        uint256 burnAmount = amount.mul(1).div(100);
        // 98% of transfer sent to recipient
        uint256 sendAmount = amount.sub(burnAmount);
        require(amount == sendAmount + burnAmount, "Shark::transfer: Burn value invalid");

        super._transfer(sender, BURN_ADDRESS, burnAmount);
        super._transfer(sender, recipient, sendAmount);
        amount = sendAmount;
    }
}
```

Recommendation:

There should be a burn instead of sending to the dead address.

Low Severity Issues

1. Block gas limit

Issue:

The updateEmissionRate function can fail due to block gas limit if the pool size is too big.

2. add function issue

Issue:

If some LP token is added to the contract twice using function add, then the total amount of reward SharkReward in function updatePool will be incorrect.

Recommendation:

Add the mapping from address to bool and check that same address will not be added twice.

Owner privileges

- ❏ Owner can change the referral commission rate.
- ❏ Owner can change the shark referral.

Conclusion

Smart contracts contain medium and low severity issues!

Techrate note:
Please check the disclaimer above and note, the audit makes no statements or warranties on business model, investment attractiveness or code sustainability. The report is provided for the only contract mentioned in the report and does not include any other potential contracts deployed by Owner.