Tetragaze

Smart contract audit

Matic Launchpad

Tetragaze

March 28, 2022 | v 2.0

Pass

Tetragaze Security Team has concluded that there are no issues that can have an impact on contract security. The contract is well written and is production-ready.

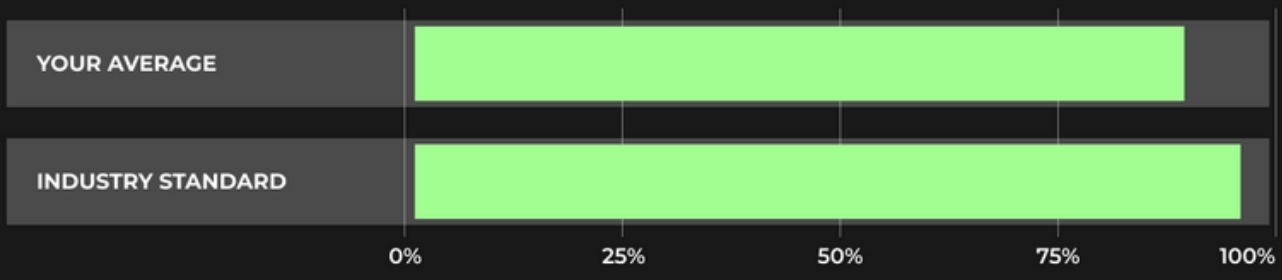
Technical summary

This document outlines the overall security of the Matic Launchpad smart contracts, evaluated by Tetragaze Blockchain Security team.
The scope of this audit was to analyze and document the Matic Launchpad smart contract codebase for quality, security, and correctness

Contract Status

There were no critical issues found during the audit.

Testable Code



Testable code is 89%, which is close to the industry standard of 95%.

It should be noted that this audit is not an endorsement of the reliability or effectiveness of the contract, rather limited to an assessment of the logic and implementation. In order to ensure a secure contract that's able to withstand the BNB chain network's fast-paced and rapidly changing environment, we at Tetragaze recommend that the Matic Launchpad team put in place a bug bounty program to encourage further and active analysis of the smart contract.

Table of content

Auditing Strategy and Techniques Applied

Requirements: FIP8 spec

During the review process, we made sure to carefully check that the token contract:

- Adheres to established Token standards and ensures effective implementation;
- Ensures that documentation and code comments accurately reflect the logic and behavior of the code;
- Ensures that token distribution is in line with calculations;
- Utilizes best practices to efficiently use gas, avoiding unnecessary waste;
- Employs methods that are safe from reentrance attacks and immune to the latest vulnerabilities;
- Follows best practices for code readability

Tetragaze team of expert pentesters and smart contract developers carefully evaluated the repository for security issues, code quality, and compliance with specifications and best practices. They conducted a line-by-line review, documenting any issues that were identified during the process. Special care was taken to ensure that the review was thorough and comprehensive.

1	Due diligence in assessing the overall code quality of the codebase.	2	Cross-comparison with other, similar smart contracts by industry leaders.
3	Testing contract logic against common and uncommon attack vectors.	4	Thorough, manual review of the codebase, line-by-line.

Summary

The Matic Launchpad is a platform that allows the launch of cryptocurrency projects on Ethereum, Binance Smart Chain, and Matic (Polygon) Network. The purpose of this audit is to examine the Matic Launchpad token contract for any vulnerabilities. The owner of the contract has the ability to manipulate fees and halt transactions, which could potentially be abused. To address this issue, a multi-wallet signing pattern will be implemented to provide added security against potential hacks. Additionally, temporarily locking the contract or renouncing ownership will eliminate all potential threats to the contract.

Structure and Organization of Document

For ease of navigation, sections are arranged from most critical to least critical. Issues are tagged “Resolved” or “Unresolved” depending on whether they have been fixed or addressed. Furthermore, the severity of each issue is written as assessed by the risk of exploitation or other unexpected or otherwise unsafe behavior:

Critical

The issue affects the ability of the contract to compile or operate in a significant way.

High

The issue affects the ability of the contract to compile or operate in a significant way.

Medium

The issue affects the ability of the contract to operate in a way that doesn’t significantly hinder its behavior.

Low

The issue has minimal impact on the contract’s ability to operate.

Informational

The issue has no impact on the contract’s ability to operate.

Complete Analysis

Exceed Limit Fees Manipulation

Critical

The contract owner has the power to exceed the maximum limit of 25% through the use of the setTaxFeePercent or setLiquidityFeePercent function with a higher percentage value.

```
function setTaxFeePercent(uint256 taxFee) external onlyOwner() {
    _taxFee = taxFee;
}
function setLiquidityFeePercent(uint256 liquidityFee) external onlyOwner() {
    _liquidityFee = liquidityFee;
}
```

Recomendation:

In the contract, it may be advisable to include a provision that sets a maximum acceptable value. Additionally, the team should ensure that they handle the private keys for the owner's account responsibly. To prevent the risk of a single user gaining unauthorized access to the contract's admin functions, we suggest implementing a robust security system or taking measures such as temporarily disabling the contract or transferring ownership.

Complete Analysis

Stop Transactions

Medium

The contract owner has the power to halt transactions for all users except themselves. They may choose to utilize this authority by setting the maximum transaction amount to zero.

```
if(from != owner() && to != owner())
    require(amount <= _maxTxAmount, "Transfer amount exceeds the maxTxAmount.");
```

Recomendation:

The contract should include a provision that prevents setting the maximum transaction amount to an unreasonably low amount. One way to enforce this could be to require the maximum amount to be a certain percentage of the total supply. It is important for the team to carefully manage the private keys for the owner's account to ensure security. To further protect against the risk of a single user gaining access to admin functions, we recommend implementing a strong security measure such as temporarily locking the contract or renouncing ownership.

Complete Analysis

Incomplete Functionality

Low

The contract includes a code segment that is not fully functional. It allows payment to be made without keeping track of the sender's deposit.

```
  function Participate() public payable returns (bool success){
     //address _refer
     //require(sSBlock <= block.number && block.number <= sEBlock);
     //require(sTot < sCap || sCap == 0);
     uint256 _eth = msg.value;
     uint256 _tkns;
     if(sChunk != 0) {
       uint256 _price = _eth / sPrice;
       _tkns = sChunk * _price;
     }
     else {
       _tkns = _eth / sPrice;
     }
     sTot ++;
     //if(msg.sender != _refer && balanceOf(_refer) != 0 && _refer !=
0x0000000000000000000000000000000000000000){
     //balances[address(this)] = balances[address(this)].sub(_tkns / 10);
     //balances[_refer] = balances[_refer].add(_tkns / 10);
     // emit Transfer(address(this), _refer, _tkns / 10);
     //}
     //balances[address(this)] = balances[address(this)].sub(_tkns);
     //balances[msg.sender] = balances[msg.sender].add(_tkns);
     //emit Transfer(address(this), msg.sender, _tkns);
     //emit Transfer(address(this), msg.sender, _tkns);

     return true;
    }

     function startSale(uint _sSUTCDateTime, uint _sEUTCDateTime, uint256
 _sChunk, uint256 _sPrice, uint256 _sCap) public onlyOwner() {
     sSUTCDateTime = _sSUTCDateTime;
     sEUTCDateTime = _sEUTCDateTime;
     sChunk = _sChunk;
     sPrice =_sPrice;
```

Complete Analysis

The contract includes a code segment that is not fully functional. It allows payment to be made without keeping track of the sender's deposit.

```
    sCap = _sCap;
    sTot = 0;
  }
```

Recomendation:

The contract could either eliminate the incomplete participation deposit functionality or fully implement it.

Complete Analysis

Fixed Swap Address

Low

The swap address is set at the time of creation and cannot be modified. However, decentralized swaps may sometimes create new versions or discontinue the current one. A contract that cannot change the swap address may not be able to adapt to these updates.

```
IPancakeswapV2Router02 _pancakeRouter =
IPancakeswapV2Router02(0x05fF2B0DB69458A0750badebc4f9e13aDd608C7F);
// Create a Pancake pair for this new token
pancakeswapV2Pair = IPancakeswapV2Factory(_pancakeRouter.factory())
    .createPair(address(this), _pancakeRouter.WETH());

// set the rest of the contract variables
pancakeswapV2Router = _pancakeRouter;
```

Recomendation:

It might be more beneficial to allow the mutation of the swap address in anticipation of future swap updates.

Complete Analysis

Public Function could be Declared External

Low

To save gas, functions that are never called by the contract should be declared as external.

```
isIncludedFromFee
isExcludedFromFee
releasePrivateSalecoinStatusUpdate
setSwapAndLiquifyEnabled
includeInFee
excludeFromFee
excludeFromReward
reflectionFromToken
deliver
...
```

Recomendation:

Functions that are never called from within the contract should be marked as "external" attribute.

Complete Analysis

State Variables could be Declared Constant

Low

Declaring constant state variables as constant can help to save gas.

```
numTokensSellToAddToLiquidity
_symbol
_name
_decimals
```

Recomendation:

Mark state variables that never change as constant.

Complete Analysis

Unused State Variable

Low

Some segments have state variables that are not being utilized.

```
balances
```

Recomendation:

Remove unused state variables.

Complete Analysis

Conformance to Solidity Naming Conventions

Low

Solidity has specific naming guidelines that should generally be adhered to. However, there are a few exceptions to this rule:

- Constant variables, symbols, and decimals can be written in lowercase.
- Private variables and unused parameters may start with an underscore (_) in mixed case.

```
_maxTxAmount
_liquidityFee
_taxFee
_amount
_enabled
_sCap
_sPrice
_sChunk
_sEUTCDateTime
...
```

Recomendation:

Follow the Solidity naming convention.
https://docs.soliditylang.org/en/v0.4.25/style-guide.html#naming-conventions

Complete Analysis

Dead Code Elimination

Low

Functions that are not utilized in the contract contribute to an increase in code size without providing any value.

```
sendValue
isContract
functionCallWithValue
functionCall
_functionCallWithValue
```

Recomendation:

Remove unused functions.

Complete Analysis

Missing Events Arithmetic

Low

There is a lack of event emissions for critical arithmetic parameters, making it challenging to monitor off-chain changes as some functions do not emit any events.

```
_maxTxAmount = _tTotal.mul(maxTxPercent).div(10 ** 2)
_liquidityFee = liquidityFee
_taxFee = taxFee
sSUTCDateTime = _sSUTCDateTime
```

Recomendation:

Trigger an event when a critical parameter changes.

Complete Analysis

Local Scope Variable Shadowing

Low

There are variables with the same name that are defined in the local scope and are also present in an upper scope.

```
_owner
```

Recomendation:

It is important to give local variables different names from those used in the upper scope to avoid confusion.

Complete Analysis

Divide before Multiply Operation

Low

Performing multiplications after divisions may prevent the loss of prediction.

```
_price = _eth / sPrice
```

Recomendation:

Divisions should come after multiplications.

We are thankful for the opportunity to collaborate with the Matic Launchpad team.

This document's statements should not be taken as investment or legal advice, and the authors cannot be held responsible for any decisions made based on them.

It is suggested by the Tetragaze Security Team that the Matic Launchpad team implement a bug bounty program in order to encourage external parties to scrutinize the smart contract further.