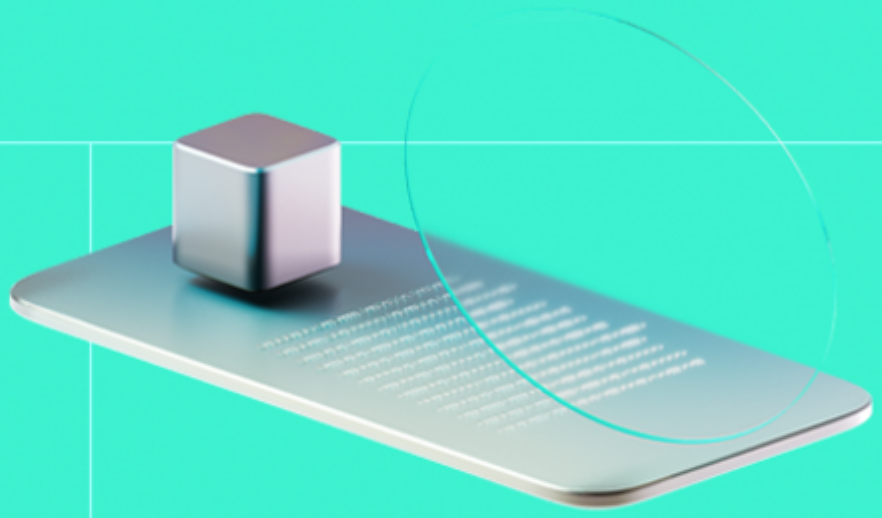
HACKEN

Smart Contract Code Review And Security Analysis Report



Customer: Pikamoon

Date: 06/03/2024

We express our gratitude to the Tigercrypto team for the collaborative engagement that enabled the execution of this Smart Contract Security Assessment.

Pikamoon is a GameFi token with a focus on community.

Platform: EVM

Language: Solidity

Tags: ERC20, Upgradable

Timeline: 04/03/2024 - 06/03/2024

Methodology: https://hackenio.cc/sc_methodology

Review Scope

Repository	https://github.com/orbit-cosmos/PikaMoon
Commit	d1875af2d9c17470ed72eb507aa3dd6af3f2bb5b
Remediation Commit	7f252cca31d4c3ec92dc8f0e26c07a5ab802d907

Audit Summary

10/10	10/10	65.91%	10/10
Security Score	Code quality score	Test coverage	Documentation quality score

Total 10/10

The system users should acknowledge all the risks summed up in the risks section of the report

4	4	0	0
Total Findings	Resolved	Accepted	Mitigated

Findings by severity

Critical	0
High	0
Medium	2
Low	2

Vulnerability	Status
F-2024-1211 - The OWNER_ROLE can burn tokens from users	Fixed
F-2024-1243 - Missing validation of tax value in tax setters functions	Fixed
F-2024-1272 - Several supposedly constant variables can be modified	Fixed
F-2024-1273 - Misuse of the feeMultiply variable	Fixed

This report may contain confidential information about IT systems and the intellectual property of the Customer, as well as information about potential vulnerabilities and methods of their exploitation.

The report can be disclosed publicly after prior consent by another Party. Any subsequent publication of this report shall be without mandatory consent.

Document

Name	Smart Contract Code Review and Security Analysis Report for Pikamoon
Audited By	Niccolò Pozzolini, Kornel Światłowski
Approved By	Przemyslaw Swiatowiec
Website	https://www.pikamoon.io/
Changelog	05/03/2024 - Preliminary Report; 06/03/2024 Second Review

Table of Contents

System Overview

PikaMoon — simple ERC20 token contract that allows minting and burning of tokens.

It has the following attributes:

- Name: Pikamoon
- Symbol: PIKA
- Decimals: 9
- Total supply: 50_000_000_000 tokens.

Privileged roles

PikaMoon contract uses a AccessControlUpgradeable library from OpenZeppelin to restrict access to important functions. There is defined `OWNER_ROLE` and addresses with this role can:

- set Automated MarketMaker pair,
- mint new tokens and assign them to a specified address,
- burn existing tokens from a specified owner's balance,
- change fee multiply,
- set ecosystem address,
- set marketing address,
- update isExcludeFromTax mapping to exclude or include from tax,
- toggle on/off tax
- set marketing tax,
- set eco system tax,
- set burn tax

Executive Summary

This report presents an in-depth analysis and scoring of the customer's smart contract project. Detailed scoring criteria can be referenced in the scoring methodology.

Documentation quality

The total Documentation Quality score is **10** out of **10**.

- Functional requirements are complete.
- Technical description is provided.

Code quality

The total Code Quality score is **10** out of **10**.

- The development environment is configured.

Test coverage

Code coverage of the project is **65.91%** (branch coverage).

- Negative cases coverage is missed.

Security score

Upon auditing, the code was found to contain **0** critical, **0** high, **2** medium, and **2** low severity issues. All issues were fixed in the remediation part of auditing process, leading to a security score of **10** out of **10**.

All identified issues are detailed in the “Findings” section of this report.

Summary

The comprehensive audit of the customer's smart contract yields an overall score of **10**. This score reflects the combined evaluation of documentation, code quality, test coverage, and security aspects of the project.

Risks

- PikaMoon contract is upgradeable: the owner can change the logic at any time, possibly affecting user balances.
- The project whitepaper precisely details the PikaMoon tokenomics, but a vesting contract is lacking from the provided repository and thus cannot be verified.
- The Solidity version 0.8.20 employs the recently introduced `PUSH0` opcode in the Shanghai EVM. This opcode might not be universally supported across all blockchain networks and Layer 2 solutions. Thus, as a result, it might be not possible to deploy solution with version 0.8.20 >= on some blockchains.

Findings

Vulnerability Details

F-2024-1211 - The OWNER_ROLE can burn tokens from users - Medium

Description: Accounts holding the `OWNER_ROLE` have the capability to burn tokens from any designated account. Privileged roles should not possess access to user tokens.

```
/**
* @dev Function to burn existing tokens from a specified owner's bal
ance.
* @param owner The address from which the tokens are burned.
* @param amount The amount of tokens to be burned.
*/

function burn(address owner, uint amount) external onlyRole(OWNER_RO
LE) {
// Call the internal _burn function from ERC20 to destroy tokens
_burn(owner, amount);
}
```

Assets:

- contracts/PikaMoon.sol [https://github.com/orbit-cosmos/PikaMoon]

Status: Fixed

Classification

Severity: Medium

Impact:
Likelihood [1-5]: 3
Impact [1-5]: 5
Exploitability [0-2]: 2
Complexity [0-2]: 0
Final Score: 2.5 (Medium)
Hacken Calculator Version: 0.6

Recommendations

Recommendation: It is recommended to utilize `ERC20BurnableUpgradeable` library from OpenZeppelin. This library includes the `burn()` and `burnFrom()` functions, ensuring that only the owner of `ERC20` tokens or an approved address can burn tokens. By implementing this library, addresses with the

`OWNER_ROLE` will no longer have the capability to burn user tokens and tokens will be protected.

Remediation: The problematic `burn` function has been removed.

Remediation Commit: 1a3c943

Evidences

POC

Reproduce:

```
import { loadFixture } from "@nomicfoundation/hardhat-toolbox/networ
k-helpers";
import { expect } from "chai";
import { ethers, upgrades } from "hardhat";
import { PikaMoon } from "../typechain-types";

const ZeroAddress = ethers.ZeroAddress;
const toWei = (value: number) => ethers.parseEther(value.toString())
;

describe("audit", function () {

async function deployFixture() {
const [owner, acc1, acc2, acc3] = await ethers.getSigners();
const pikamoon = await ethers.getContractFactory("PikaMoon");

const token = await upgrades.deployProxy(pikamoon,
[
"PIKAMoon",
"PIKA",
toWei(50_000_000_000),
owner.address,
owner.address
],
{ initializer: "initialize" }

);
return { token, owner, acc1, acc2, acc3 };
}
describe("burn()", function () {
it("owner can burn tokens from users", async function () {
const { token, owner, acc1 } = await loadFixture(deployFixture);
const amount = ethers.parseEther("100");

await token.connect(owner).mint(acc1.address, amount);
expect(await token.balanceOf(acc1.address)).to.be.equal(amount);
console.log(`Acc1 balance before: ${await token.balanceOf(acc1.addre
ss)}`);

await token.connect(owner).burn(acc1.address, amount);
expect(await token.balanceOf(acc1.address)).to.be.equal(0);
console.log(`Acc1 balance after : ${await token.balanceOf(acc1.addre
ss)}`);
});
});
});
```

Results:

```
  audit
    burn()
Acc1 balance before: 100000000000000000000
Acc1 balance after : 0
      ✓owner can burn tokens from users (9452ms)
```

F-2024-1272 - Several supposedly constant variables can be modified - Medium

Description: The `_cap`, `marketingTax`, `ecosystemTax`, and `burnTax` variables are described as known and constant in the project documentation. However, in the `initialize` function in `PikaMoon.sol`, these variables are dynamically set. Furthermore, the tax variables (`marketingTax`, `ecosystemTax`, `burnTax`) have setter functions, which contradicts their description as constant.

Assets:

- contracts/PikaMoon.sol [https://github.com/orbit-cosmos/PikaMoon]

Status: Fixed

Classification

Severity: Medium

Impact:
Likelihood [1-5]: 5
Impact [1-5]: 5
Exploitability [0-2]: 2
Complexity [0-2]: 0
Final Score: 2.9 (Medium)
Hacken Calculator Version: 0.6

Recommendations

Recommendation: To ensure consistency between the code and the documentation, and to avoid potential confusion or misuse, these variables should be declared as constant in the code. This change would align the implementation with the documented design and expectations.

Remediation: Tax and cap values have been made constant.

Remediation Commit: edc11f1

F-2024-1243 - Missing validation of tax value in tax setters functions - Low

Description:

The `transfer()` and `transferFrom()` functions deduct three types of taxes - marketing, ecosystem, and burn tax. The percentage value of each tax is stored in corresponding state variables: `marketingTax`, `ecosystemTax`, and `burnTax`. These variables are set inside the `initialize()` function. However, addresses with `OWNER_ROLE` can change each tax value using dedicated setter functions. These setter functions do not have an upper cap.

The absence of validation within `setMarketingTax()`, `setEcoSystemTax()`, and `setBurnTax()` functions may lead to a situation where the sum of taxes equals or is bigger than the transferred amount, resulting in the transfer recipient receiving 0 tokens or Denial of Service.

```
function setMarketingTax(uint16 _marketingTax) external onlyRole(OWN
ER_ROLE) {
marketingTax = _marketingTax; // 1%
}

function setEcoSystemTax(uint16 _ecosystemTax) external onlyRole(OWN
ER_ROLE) {
ecosystemTax = _ecosystemTax; // 1%
}

function setBurnTax(uint16 _burnTax) external onlyRole(OWNER_ROLE) {
burnTax = _burnTax;
}
```

Assets:

- contracts/PikaMoon.sol [https://github.com/orbit-cosmos/PikaMoon]

Status: Fixed

Classification

Severity: Low

Impact:

Likelihood [1-5]: 2
Impact [1-5]: 5
Exploitability [0-2]: 2
Complexity [0-2]: 0
Final Score: 2.3 (Low)
Hacken Calculator Version: 0.6

Recommendations

Recommendation: It is recommended to introduce an upper cap for new tax value in: `setMarketingTax()`, `setEcoSystemTax()` and `setBurnTax()`. Additionally, it is recommended to validate that the cumulative sum of all taxes remains below the `feeMultiply` value. For example in `setMarketingTax()` this can be achieved by adding the following check:

```
function setMarketingTax(uint16 _marketingTax) external onlyRole(OWN
ER_ROLE) {
require(burnTax + _marketingTax + ecosystemTax < feeMultiply, ERROR_
MSG);
marketingTax = _marketingTax;
}
```

Remediation: Tax values have been made constant.

Remediation Commit: edc11f1

F-2024-1273 - Misuse of the feeMultiply variable - Low

Description: The `feeMultiply` variable in `PikaMoon.sol` represents the precision used when applying percentages, such as transfer fees. As such, it should not be subject to change, as there is no valid reason for doing so.

The `changeFeeMultiply` function currently allows for the modification of `feeMultiply`. However, if `feeMultiply` is changed without a corresponding change in the fees percentages, the contract could enter a faulty state each time this function is called.

Assets:

- contracts/PikaMoon.sol [https://github.com/orbit-cosmos/PikaMoon]

Status: Fixed

Classification

Severity: Low

Impact:
Likelihood [1-5]: 3
Impact [1-5]: 4
Exploitability [0-2]: 2
Complexity [0-2]: 0
Final Score: 2.3 (Low)
Hacken Calculator Version: 0.6

Recommendations

Recommendation: To resolve this issue, `feeMultiply` should be declared as a constant, ensuring its value remains consistent and unchangeable throughout the contract's lifecycle. Consequently, the `changeFeeMultiply` function, which serves as a setter for `feeMultiply`, should be removed from the contract. This change will prevent potential inconsistencies and faulty states in the contract.

Remediation: The `feeMultiplier` setter has been removed.

Remediation Commit: 1a3c943

Observation Details

F-2024-1203 - Floating Pragma - Info

Description: The project uses floating pragmas `^0.8.20`.

This may result in the contracts being deployed using the wrong pragma version, which is different from the one they were tested with. For example, they might be deployed using an outdated pragma version which may include bugs that affect the system negatively.

Assets:

- contracts/PikaMoon.sol [https://github.com/orbit-cosmos/PikaMoon]
- contracts/interfaces/IPikaMoon.sol [https://github.com/orbit-cosmos/PikaMoon]
- contracts/interfaces/IUniswapV2Router02.sol [https://github.com/orbit-cosmos/PikaMoon]
- contracts/libraries/Errors.sol [https://github.com/orbit-cosmos/PikaMoon]

Status: Fixed

Recommendations

Recommendation: Consider locking the pragma version whenever possible and avoid using a floating pragma in the final deployment. Consider known bugs for the compiler version that is chosen.

Remediation: Pragma version has been locked to 0.8.20.

Remediation Commit: 1a3c943

F-2024-1204 - Redundant imports - Info

Description: Following interfaces are imported but never used:

- `IPikaMoon`
- `IERC20`
- `IUniswapV2Router02`

This redundancy in import operations has the potential to result in unnecessary gas consumption during deployment and could potentially impact the overall code quality.

Assets:

- contracts/PikaMoon.sol [https://github.com/orbit-cosmos/PikaMoon]

Status: Fixed

Recommendations

Recommendation: Remove redundant imports, and ensure that the contract is imported only in the required locations, avoiding unnecessary duplications.

Remediation: Redundant imports have been removed.

Remediation Commit: 1a3c943

F-2024-1207 - Public functions that should be external - Info

Description: Functions that are meant to be exclusively invoked from external sources should be designated as `external` rather than `public`. This is essential to enhance both the gas efficiency and the overall security of the contract.

`external` visibility can be added to:

- initialize,
- setAutomatedMarketMakerPair

Assets:

- contracts/PikaMoon.sol [https://github.com/orbit-cosmos/PikaMoon]

Status: Fixed

Recommendations

Recommendation: To optimize gas usage and improve code clarity, declare functions that are not called internally within the contract and are intended for external access as `external` rather than `public`. This ensures that these functions are only callable externally, reducing unnecessary gas consumption and potential security risks.

Remediation: The visibility of the mentioned functions has been changed to external..

Remediation Commit: 1a3c943

F-2024-1208 - Missing events emitting for critical functions - Info

Description:

Events for critical state changes should be emitted for tracking actions off-chain.

Events are crucial for tracking changes on the blockchain, especially for actions that alter significant contract states or permissions. The absence of events in these functions means that external entities, such as user interfaces or off-chain monitoring systems, cannot effectively track these important changes.

It was observed that events are missing events in the following functions:

- setMarketingTax,
- setEcoSystemTax,
- setBurnTax,
- toggleTax

Assets:

- contracts/PikaMoon.sol [https://github.com/orbit-cosmos/PikaMoon]

Status: Fixed

Recommendations

Recommendation: Consider events emitting on aforementioned functions.

Remediation: Events have been added to the mentioned functions.

Remediation Commit: 1a3c943

F-2024-1276 - Redundant nested if structure and unnecessary assignments in calculateTax function - Info

Description:

The `calculateTax` function in `PikaMoon.sol` currently uses a nested if structure to determine the tax amount. This structure is redundant and negatively impacts the code quality and readability.
Affected code:

```
/**
* @dev Function to calculate the tax
* @param from address on which tax is applied
* @param to address on which tax is applied
* @param value amount on which tax is calculated
*/
function calculateTax(
address from,
address to,
uint256 value
)
public
view
returns (
uint256 tax,
uint256 burnAmount,
uint256 marketingAmount,
uint256 ecosystemAmount
)
{
// calculate tax
if (isTaxEnabled) {
if (automatedMarketMakerPairs[from]) {
// means buying from known AMM/DEX
tax = 0;
} else {
if (isExcludeFromTax[from] || isExcludeFromTax[to]) {
// means from or to is excluded from fee
tax = 0;
} else {
burnAmount = (value * burnTax) / feeMultiply;
marketingAmount = (value * marketingTax) / feeMultiply;
ecosystemAmount = (value * ecosystemTax) / feeMultiply;
unchecked {
tax = burnAmount + marketingAmount + ecosystemAmount;
}
}
}
}
}
```

Assets:

- contracts/PikaMoon.sol [https://github.com/orbit-cosmos/PikaMoon]

Status: Fixed

Recommendations

Recommendation:

The nested if conditions can be replaced by a single if condition that checks whether tax should be calculated. The condition should be as follows:

```
if(isTaxEnabled && !automatedMarketMakerPairs[from] && !isExcludeFro
mTax[from] && !isExcludeFromTax[to])
```

This condition checks if tax is enabled and if neither the sender nor the receiver are excluded from tax, and if the sender is not a known automated market maker pair. If all these conditions are met, the tax is calculated.

Additionally, the assignments `tax = 0` are unnecessary, as 0 is the default value for uninitialized uint256 variables in Solidity. These assignments can be removed to simplify the function.

Remediation: The `calculateTax` function has been improved as suggested.

Remediation Commit: 1a3c943

Disclaimers

Hacken Disclaimer

The smart contracts given for audit have been analyzed based on best industry practices at the time of the writing of this report, with cybersecurity vulnerabilities and issues in smart contract source code, the details of which are disclosed in this report (Source Code); the Source Code compilation, deployment, and functionality (performing the intended functions).

The report contains no statements or warranties on the identification of all vulnerabilities and security of the code. The report covers the code submitted and reviewed, so it may not be relevant after any modifications. Do not consider this report as a final and sufficient assessment regarding the utility and safety of the code, bug-free status, or any other contract statements.

While we have done our best in conducting the analysis and producing this report, it is important to note that you should not rely on this report only — we recommend proceeding with several independent audits and a public bug bounty program to ensure the security of smart contracts.

English is the original language of the report. The Consultant is not responsible for the correctness of the translated versions.

Technical Disclaimer

Smart contracts are deployed and executed on a blockchain platform. The platform, its programming language, and other software related to the smart contract can have vulnerabilities that can lead to hacks. Thus, the Consultant cannot guarantee the explicit security of the audited smart contracts.

Appendix 1. Severity Definitions

When auditing smart contracts, Hacken is using a risk-based approach that considers **Likelihood**, **Impact**, **Exploitability** and **Complexity** metrics to evaluate findings and score severities.

Reference on how risk scoring is done is available through the repository in our Github organization:

hknio/severity-formula

Severity	Description
Critical	Critical vulnerabilities are usually straightforward to exploit and can lead to the loss of user funds or contract state manipulation.
High	High vulnerabilities are usually harder to exploit, requiring specific conditions, or have a more limited scope, but can still lead to the loss of user funds or contract state manipulation.
Medium	Medium vulnerabilities are usually limited to state manipulations and, in most cases, cannot lead to asset loss. Contradictions and requirements violations. Major deviations from best practices are also in this category.
Low	Major deviations from best practices or major Gas inefficiency. These issues will not have a significant impact on code execution, do not affect security score but can affect code quality score.

Appendix 2. Scope

The scope of the project includes the following smart contracts from the provided repository:

Scope Details

Repository	https://github.com/orbit-cosmos/PikaMoon
Commit	d1875af2d9c17470ed72eb507aa3dd6af3f2bb5b
Whitepaper	https://docs.pikamoon.io/
Requirements	https://github.com/orbit-cosmos/PikaMoon/blob/master/docs/index.md
Technical Requirements	https://github.com/orbit-cosmos/PikaMoon/blob/master/docs/index.md

Contracts in Scope

contracts/PikaMoon.sol

contracts/interfaces/IPikaMoon.sol

contracts/interfaces/IUniswapV2Router02.sol

contracts/libraries/Errors.sol