SALUS SECURITY

NOV 2023

CODE SECURITY ASSESSMENT

ALIENFORM

Overview

Project Summary

- Name: AlienForm
- Platform: Ethereum / BNB Smart Chain
- Language: Solidity
- Audit Range: See Appendix - 1

Project Dashboard

Application Summary

Name	AlienForm
Version	v4
Type	Solidity
Dates	Nov 30 2023
Logs	Nov 10 2023; Nov 22 2023; Nov 27 2023; Nov 30 2023;

Vulnerability Summary

Total High-Severity issues	0
Total Medium-Severity issues	2
Total Low-Severity issues	3
Total informational issues	7
Total	12

Contact

E-mail: support@salusec.io

Risk Level Description

High Risk	The issue puts a large number of users' sensitive information at risk, or is reasonably likely to lead to catastrophic impact for clients' reputations or serious financial implications for clients and users.
Medium Risk	The issue puts a subset of users' sensitive information at risk, would be detrimental to the client's reputation if exploited, or is reasonably likely to lead to a moderate financial impact.
Low Risk	The risk is relatively small and could not be exploited on a recurring basis, or is a risk that the client has indicated is low impact in view of the client's business circumstances.
Informational	The issue does not pose an immediate risk, but is relevant to security best practices or defense in depth.

Content

Introduction

1.1 About SALUS

At Salus Security, we are in the business of trust.

We are dedicated to tackling the toughest security challenges facing the industry today. By building foundational trust in technology and infrastructure through security, we help clients to lead their respective industries and unlock their full Web3 potential.

Our team of security experts employ industry-leading proof-of-concept (PoC) methodology for demonstrating smart contract vulnerabilities, coupled with advanced red teaming capabilities and a stereoscopic vulnerability detection service, to deliver comprehensive security assessments that allow clients to stay ahead of the curve.

In addition to smart contract audits and red teaming, our Rapid Detection Service for smart contracts aims to make security accessible to all. This high calibre, yet cost-efficient, security tool has been designed to support a wide range of business needs including investment due diligence, security and code quality assessments, and code optimisation.

We are reachable on Telegram (https://t.me/salusec), Twitter (https://twitter.com/salus_sec), or Email (support@salusec.io).

1.2 Audit Breakdown

The objective was to evaluate the repository for security-related issues, code quality, and adherence to specifications and best practices. Possible issues we looked for included (but are not limited to):

- Risky external calls
- Integer overflow/underflow
- Transaction-ordering dependence
- Timestamp dependence
- Access control
- Call stack limits and mishandled exceptions
- Number rounding errors
- Centralization of power
- Logical oversights and denial of service
- Business logic specification
- Code clones, functionality duplication

1.3 Disclaimer

Note that this security audit is not designed to replace functional tests required before any software release and does not give any warranties on finding all possible security issues with the given smart contract(s) or blockchain software, i.e., the evaluation result does not guarantee the nonexistence of any further findings of security issues.

Findings

2.1 Summary of Findings

ID	Title	Severity	Category	Status
1	A testnet address is used	Medium	Configuration	Resolved
2	Wrong conditional judgment	Medium	Data Validation	Resolved
3	Rounding Issue in takeFee()	Low	Calculation	Resolved
4	Centralization risk	Low	Centralization	Mitigated
5	Should use call instead of transfer	Low	Code Quality	Resolved
6	Weak slippage protection	Informational	Data Validation	Acknowledged
7	Gas optimization suggestions	Informational	Gas Optimization	Resolved
8	Floating pragma version	Informational	Configuration	Acknowledged
9	Missing zero-address check	Informational	Data Validation	Resolved
10	Missing events	Informational	Logging	Acknowledged
11	Could use Ownable2Step instead of Ownable	Informational	Access Control	Resolved
12	Dos caused by loop	Informational	Denial of Service	Acknowledged

2.2 Notable Findings

Significant flaws that impact system confidentiality, integrity, or availability are listed below.

1. A testnet address is used

Severity: Medium	Category: Configuration
Target: - AlienForm.sol	

Description

In the AlienForm contract, the comment states that the routerAddress for the BSC mainnet is 0xdc4904b5f716Ff30d8495e35dC99c109bb5eCf81. However, this is the address for the BSC testnet. If you plan to deploy the project on the BSC mainnet, using the incorrect address will result in the contract not functioning as expected.

AlienForm.sol:L1197

```
// address routerAddress = 0xdc4904b5f716Ff30d8495e35dC99c109bb5eCf81; // BSC Mainnet
```

Recommendation

Ensure that using the correct router address for the BSC Mainnet. The PancakeRouter address in BSC Mainnet is 0x10ED43C718714eb63d5aA57B78B54704E256024E.

Status

This issue has been resolved by the team.

2. Wrong conditional judgment

Severity: Medium	Category: Data Validation
Target: - Bridge.sol	

Description

Bridge.sol: L713-L714

```
function receiveTokens(address from, address to, uint amount, uint otherChainNonce)
external onlyOwner() {
        require(sentProcessedNonce[from][otherChainNonce] == false, 'transfer already
processed');
        receivedProcessedNonce[from][otherChainNonce] = true;
        ...
}
```

The receiveTokens() function is for the 'to' address to receive fund transfers from other chains.

However, the function checks the state of sentProcessedNonce[from][otherChainNonce] rather than receivedProcessedNonce[from][otherChainNonce]. It means that the function cannot revert calls that do not conform to the expected condition, and the owner can mint unlimited tokens to a specified address.

Recommendation

Consider determining whether from and nonce are in the receiving state in the receiveTokens() function.

That is, change
`require(sentProcessedNonce[from][otherChainNonce] == false, 'transfer already processed');`
to
`require(receivedProcessedNonce[from][otherChainNonce] == false, 'transfer already processed');`

Status

This issue has been resolved by the team.

3. Rounding Issue in takeFee()

Severity: Low	Category: Calculation
Target: - AlienForm.sol	

Description

When using division calculation, the result will be rounded down. If tokensForLiquidity and ethForLiquidity both use the results of division calculation, and _liquidityTokensToSwap will be assigned 0 at the end of this function, it may cause _liquidityTokensToSwap not to be fully allocated/used.

AlienForm.sol: L1440-L1455

```
function takeFee(uint256 contractBalance) private lockTheSwap {
        ...
        uint256 tokensForLiquidity = _liquidityTokensToSwap / 2;
        ...
        if (tokensForLiquidity > 0 && ethForLiquidity > 0) {
        addLiquidity(tokensForLiquidity, ethForLiquidity);
        }
        ...
}
```

Recommendation

Consider using division for one variable and subtraction for the other to ensure more accurate distribution.

Can change
`addLiquidity(tokensForLiquidity, ethForLiquidity);`
to
`addLiquidity(_liquidityTokensToSwap - tokensForLiquidity, ethForLiquidity);`

Status

This issue has been resolved by the team.

4. Centralization risk

Severity: Low	Category: Centralization
Target: - AlienForm.sol	

Description

The AlienForm contract presents a centralization risk through its fee settings. There is no upper limit set for the fees in functions like updateBuyFee(), updateSellFee(), updateTransferFee(), and updateMarketingFeeAddress(). Incorrect fee settings may cause from to be charged excessive fees during the transfer process. For example:

AlienForm.sol:L1307-1317

```
function updateBuyFee(
        uint16 _buyBurnFee,
        uint16 _buyLiquidityFee,
        uint16 _buyMarketingFee,
        uint16 _buyMiscFee
) external onlyOwner {
        buyBurnFee = _buyBurnFee;
        buyLiquidityFee = _buyLiquidityFee;
        buyMarketingFee = _buyMarketingFee;
        buyMiscFee = _buyMiscFee;
}
```

Recommendation

It is suggested to add an upper limit for the fee settings to avoid excessively high fees. Additionally, we recommend transferring privileged accounts to multi-sig accounts with timelock governors for enhanced security. This ensures that no single person has full control over the accounts and that any changes must be authorized by multiple parties.

Status

The team has mitigated this issue by adding a maxFeeLimit check in the relevant functions.

5. Should use call instead of transfer

Severity: Low	Category: Code Quality
Target: - AlienForm.sol - Bridge.sol	

Description

In both the Bridge and AlienForm contracts, the transfer function is used for handling native tokens. However, it's important to note that the transfer function has a gas limit of 2300 units, making it less suitable for handling larger transactions or interacting with contracts that might require more gas. It is suggested that using the call function instead of the transfer. For more information, please refer to this article.

AlienForm.sol:L1511

```
function withdrawETH() external onlyOwner {
  payable(owner()).transfer(address(this).balance);
}
```

Bridge.sol:L742

```
function withdrawETH() external onlyOwner {
  payable(owner()).transfer(address(this).balance);
}
```

Recommendation

Consider using the address.call{value: xxx}(“”) instead of the transfer function to handle native token transfer.

Status

This issue has been resolved by the team.

2.3 Informational Findings

6. Weak slippage protection

Severity: Informational	Category: Data Validation
Target: - AlienForm.sol	

Description

The protocol uses the swapExactTokensForETHSupportingFeeOnTransferTokens() on Uniswap's router contract and sets the minimum number of tokens to zero. Due to the lack of slippage protection, trades may encounter excessive slippage and potential price manipulation, such as front-running.

AlienForm.sol: L1477-L1488

```
function swapTokensForETH(uint256 tokenAmount) private {
        address[] memory path = new address[](2);
        path[0] = address(this);
        path[1] = uniswapRouter.WETH();
        uniswapRouter.swapExactTokensForETHSupportingFeeOnTransferTokens(
        tokenAmount,
        0,
        path,
        address(this),
        block.timestamp
        );
}
```

Recommendation

Consider calculating the amountOutMin based on the latest price to ensure that the transaction results meet expectations or avoid sandwich attacks.

Status

This issue has been acknowledged by the team.

7. Gas optimization suggestions

Severity: Informational	Category: Gas Optimization

Target:
- AlienForm.sol
- MultiSend.sol
- Bridge.sol

Description

AlienForm.sol: L1141, L1146, L1151

```
uint16 public buyBurnFee = 0;
...
uint16 public sellBurnFee = 0;
...
uint16 public transferBurnFee = 0;
```

The uint256 variables are initialized to 0 by default. You can remove the assignments to save gas.

Bridge.sol: L698, L713, L727

```
function sendTokens(address to, uint amount) external onlyBridgeEnabled() {
        require(sentProcessedNonce[msg.sender][nonce] == false, 'transfer already
processed');
        ...
}

function receiveTokens(address from, address to, uint amount, uint otherChainNonce)
external onlyOwner() {
        require(sentProcessedNonce[from][otherChainNonce] == false, 'transfer already
processed');
        ...
}

function bridgeFailed(address from, address to, uint amount, uint sameChainNonce)
external onlyOwner() {
        require(failedProcessedNonce[from][sameChainNonce] == false, 'bridge fail already
processed');
        ...
}
```

In the above functions, it is unnecessary to compare boolean variables with true or false.

MultiSend.sol:L660

```
function multiSend(
address[] calldata _addresses,
uint256[] calldata _amounts
) payable external onlyOwner {
    ...
}
```

The MultiSend contract does not handle native tokens, so there is no need to use the payable modifier.

AlienForm.sol:L1248

```
function addBotWalletBulk(address[] memory wallets) external onlyOwner {
    for (uint256 i = 0; i < wallets.length; i++) {
        ...
    }
}
```

There are some loops in the AlienForm and MultiSend contracts, changing ‘i++’ to ‘++i’ in the for loop can save gas. For example, the code snippet above.

AlienForm.sol:L1412-L1421

```
function _transfer(
    address from,
    address to,
    uint256 amount
) internal override {
    ...
    uint256 _transferAmount = amount -
    _burnFee -
    _liquidityFee -
    _marketingFee -
    _miscFee;
    super._transfer(from, to, _transferAmount);
    uint256 _feeTotal = _burnFee +
    _liquidityFee +
    _marketingFee +
    _miscFee;
    ...
}
```

It is possible to optimize gas usage by calculating _feeTotal once and then using it to compute the _transferAmount as `amount - _feeTotal`.

Recommendation

Consider making changes based on the above suggestions.

Status

This issue has been resolved by the team.

8. Floating pragma version

Severity: Informational	Category: Configuration

Target:
- AlienForm.sol
- MultiSend.sol
- Bridge.sol

Description

```
pragma solidity ^0.8.17;
```

Using a floating pragma statement is discouraged, as code may compile to different bytecodes with different compiler versions. Use a locked pragma statement to get a deterministic bytecode. Also use the latest Solidity version to get all the compiler features, bug fixes, and optimizations.

Recommendation

It is recommended to use a locked Solidity version throughout the project. It is also recommended to use the most stable and up-to-date version.

Status

This issue has been acknowledged by the team.

9. Missing zero-address check

Severity: Informational	Category: Data Validation
Target: - Bridge.sol	

Description

The token shouldn't be a zero address, but there is no zero address check in the constructor, and there is no function in the contract to change the token.

It is considered a security best practice to verify addresses against the zero address in the constructor or setting. However, this precautionary step is absent for the variables highlighted below.

Bridge.sol:L688

```
constructor(address _token) {
    token = IToken(_token);
    ...
}
```

Recommendation

Consider adding a zero-address check.

Status

This issue has been resolved by the team.

10. Missing events

Severity: Informational	Category: Logging
Target: - AlienForm.sol	

Description

Important parameter or configuration changes should trigger an event to enable off-chain tracking. However, when certain key parameters are updated in the AlienForm contract, no corresponding events are triggered to facilitate off-chain tracking. These parameters include updating fee values, updating the bridge address, updating the fee recipient address, enabling or disabling the contract, etc.

Recommendation

Design proper events and add them to the functions in this contract.

Status

This issue has been acknowledged by the team.

11. Could use Ownable2Step instead of Ownable

Severity: Informational	Category: Access Control
Target: - AlienForm.sol - MultiSend.sol - Bridge.sol	

Description

The contracts inherit from the Ownable contract. However, it's recommended to use the Ownable2Step contract which provides a 2-stage ownership transfer mechanism that can prevent ownership from being transferred to an unintended or incorrect address.

Recommendation

Consider making changes based on the above suggestion.

Status

This issue has been resolved by the team.

12. Dos caused by loop

Severity: Informational	Category: Denial of Service
Target: - AlienForm.sol	

Description

In the AlienForm contract, calling the removeBotWallet function to remove an element involves traversing the array until the element is found. When the array is large enough, this operation might exceed the block gas limit, preventing the removal of botWallet.

```
function removeBotWallet(address wallet) external onlyOwner {
    require(botWallet[wallet], "Wallet not added");
    botWallet[wallet] = false;
    for (uint256 i = 0; i < botWallets.length; i++) {
        if (botWallets[i] == wallet) {
            botWallets[i] = botWallets[botWallets.length - 1];
            botWallets.pop();
            break;
        }
    }
}
```

Recommendation

Consider adding a limit to array length.

Status

This issue has been acknowledged by the team.

Appendix

Appendix 1 - Files in Scope

This audit covered the following files provided by the client:

File	SHA-1 hash
AlienForm.sol	c61b7a780774d9f07d5743efd347e7ef30d84ee9
Bridge.sol	368bac2b8a304e53fa42df153a5a01b1951131ec
MultiSend.sol	72f10e022ecaf529cea3523def5888c113993bdf