Influx Raise

Smart Contract Security Assessment

June 24, 2024

VERACITY

Disclaimer

Veracity Security ("Veracity") has conducted an independent audit to verify the integrity of and highlight any vulnerabilities or errors, intentional or unintentional, that may be present in the codes that were provided for the scope of this audit. This audit report does not constitute agreement, acceptance or advocacy for the Project that was audited, and users relying on this audit report should not consider this as having any merit for financial advice in any shape, form or nature.

The contracts audited do not account for any economic developments that may be pursued by the Project in question, and that the veracity of the findings thus presented in this report relate solely to the proficiency, competence, aptitude and discretion of our independent auditors, who make no guarantees nor assurance that the contracts are completely free of exploits, bugs, vulnerabilities or deprecation of technologies. Further, this audit report shall not be disclosed nor transmitted to any persons or parties on any objective, goal or justification without due written assent, acquiescence or approval by the Veracity team.

All information provided in this report does not constitute financial or investment advice, nor should it be used to signal that any person reading this report should invest their funds without sufficient individual due diligence regardless of the findings presented in this report. Information is provided 'as is', and Veracity is under no covenant to the completeness, accuracy or solidity of the contracts audited. In no event will Veracity or its partners, employees, agents or parties related to the provision of this audit report be liable to any parties for, or lack thereof, decisions and/or actions with regards to the information provided in this audit report.

Cryptocurrencies and any technologies by extension directly or indirectly related to cryptocurrencies are highly volatile and speculative by nature. All reasonable due diligence and safeguards may yet be insufficient, and users should exercise considerable caution when participating in any shape or form in this nascent industry.

The audit report has made all reasonable attempts to provide clear and articulate recommendations to the Project team with respect to the rectification, amendment and/or revision of any highlighted issues, vulnerabilities or exploits within the contracts provided. It is the sole responsibility of the Project team to sufficiently test and perform checks, ensuring that the contracts are functioning as intended, specifically that the functions contained within said contracts have the desired intended effects, functionalities and outcomes of the Project team.

Disclaimer **2**
1 Overview **5**
1.1 Summary 5
1.3 Testing 5
1.2 Final Contracts Assessed 5
1.3 Findings Summary 7
1.3.1 Status Classifications 7
1.3.2 Collected Issues and Statuses 8
2. Findings **9**
2.1 USDCForwarder 9
2.1.1 Privileged Roles 9
2.1.2 Initial Token Allocation 9
2.1.3 Taxes, Rules, Initial Variables. 10
2.1.4 USDCForwarder Issues & Recommendations 11
Issue Number: 1 11
Issue Number: 2 12

1 Overview

This report has been prepared for Influx Raise (https://influxstructure.com/) . Veracity provides an examination of the smart contracts to look for vulnerabilities, logic errors or other issues from both an internal and external perspective.

1.1 Summary

Name	Influx Raise
URL	https://influxstructure.com/
Platform	Ethereum, BSC, BASE, Polygon
Language	Solidity

Influx Raise consists of 1 contracts:

USDForwarder	Forwards USDC tokens to a project controlled address. 0x5f45f3C06eb7a4F6AA5242F4629f533b5200b9bF

1.3 Testing

Following an initial pass on all contracts, we performed a series of tests. However it is not possible to catch all scenarios with these tests. Veracity has implemented a suite of audit tests that also exercise the primary functions of each contract to ensure that no transaction or fund locking occurs.

Tests have been implemented with the Foundry fuzz testing framework and some of the issues discovered are listed in the tables below. No further critical issues were discovered during this secondary process.

1.2 Final Contracts Assessed

Following deployment of the contracts assessed, Veracity compares the contracts that have been deployed, and wired with the contracts that have been audited to guarantee no tampering has been possible between audit report issue and project start.

This gives project owners and community members confidence that what has been deployed matches the findings and resolution status described in this document.

Github hash: b2b4ff1ff1d6684439c838e873f4ce20efee7678

Deployment network: Ethereum, BSC, BASC, Polygon

Project wallet address: 0x5f45f3C06eb7a4F6AA5242F4629f533b5200b9bF

Links to verified contracts:

Name	Address	Network	Matched
USDCForwarder	ETH: USDT TIER 2: 0xc00df9fe2608c66cca1a7a4627757cf864f6addc USDT TIER 3: 0xc761A399A413a88E6b5E166F0C62bF57b2445439 USDC TIER 2: 0xE1bB03b64B2bf7699912667a959bcDB3F7F710BF USDC TIER 3: 0x5fCf1De3c0fddA41C6e41b4d2656D2EAC3B8fBc2 BSC: USDT TIER 2: 0xc00dF9Fe2608C66ccA1a7a4627757cF864f6aDDC USDT TIER 3: 0xc761A399A413a88E6b5E166F0C62bF57b2445439 USDC TIER 2: 0xE1bB03b64B2bf7699912667a959bcDB3F7F710BF USDC TIER 3: 0x5fCf1De3c0fddA41C6e41b4d2656D2EAC3B8fBc2 POLYGON: USDT TIER 2: 0xc00dF9Fe2608C66ccA1a7a4627757cF864f6aDDC USDT TIER 3: 0xc761A399A413a88E6b5E166F0C62bF57b2445439 USDC TIER 2: 0xE1bB03b64B2bf7699912667a959bcDB3F7F710BF USDC TIER 3: 0x5fCf1De3c0fddA41C6e41b4d2656D2EAC3B8fBc2 BASE: USDC TIER 2: 0xc00dF9Fe2608C66ccA1a7a4627757cF864f6aDDC USDC TIER 3: 0xc761A399A413a88E6b5E166F0C62bF57b2445439	Ethereum, BSC, BASC, Polygon	**YES**

1.3 Findings Summary

Individual issues found have been categorised based on criticality as high, medium, low or informational. The client is required to respond to each issue individually, although it may be by design and therefore simply acknowledged. Additional recommendations may apply to all contracts, but are replicated for each for resolution.

For example an issue relating to centralisation of financial risk may apply to all administration functions, but will be included only once per contract. The table below shows the collected number of issues found and the resolution statuses across all contracts in the project.

Severity	Found	Resolved	Partially Resolved	Acknowledged (no change)
• High	0	0	0	0
• Medium	2	1	0	1
• Low	0	0	0	0
• Informational	0	0	0	0
Total	0	0	0	0

1.3.1 Status Classifications

Severity	Description
• High	Exploits, vulnerabilities or errors that will certainly or probabilistically lead towards loss of funds, control, or impairment of the contract and its functions. Issues under this classification are recommended to be fixed with utmost urgency.
• Medium	Bugs or issues with that may be subject to exploit, though their impact is somewhat limited. Issues under this classification are recommended to be fixed as soon as possible.
• Low	Effects are minimal in isolation and do not pose a significant danger to the project or its users. Issues under this classification are recommended to be fixed nonetheless.
• Informational	Consistency, syntax or style best practices. Generally pose a negligible level of risk, if any.
• Optimization	Suboptimal implementations that may result in additional gas consumption, unnecessary computation or avoidable inefficiencies.

1.3.2 Collected Issues and Statuses

ID	Contract	Severity	Summary	Status
01	USDCForwarder	MEDIUM	The contract does not supply any tokens to the customer and simply forwards the funds to an address provided by the deployer.	ACKNOWLEDGED
02	USDCForwarder	MEDIUM	Function `transferFrom` does not handle non-standard ERC20 tokens properly.	RESOLVED

2. Findings

The contract assessed have been largely authored from scratch rather than using industry tested implementations for ERC20. Standard interfaces have been included inline which adds risk of errors, however code comparison shows no errors have been introduced during this process. This can result in the introduction of vulnerabilities or bugs that have not been seen or addressed in previous projects. However our team has made recommendations and several code sweeps to mitigate the effect of not using industry standard libraries. The following sections outline issues found with individual contracts.

2.1 USDCForwarder

This report has been prepared for Influx Raise. Veracity provides an examination of the smart contracts to look for vulnerabilities, logic errors or other issues from both an internal and external perspective.

2.1.1 Privileged Roles

The following functions can be called by the admin of the USDCForwarder contract:

- toggleActive
- changeMaxPurchasesPerTransaction
- recoverTokens

2.1.2 Initial Token Allocation

No tokens are allocated in these contracts. The contracts only forward the incoming USDC to the project address. 0x5f45f3C06eb7a4F6AA5242F4629f533b5200b9bF

2.1.3 Taxes, Rules, Initial Variables.

This is not applicable to this contract.

2.1.4 USDCForwarder Issues & Recommendations

Issue Number: 1

Severity: Medium
File: https://github.com/Kommissar29/contracts/blob/main/USDCForwarder.sol#L49

Summary:
The contract does not supply any tokens to the customer and simply forwards the funds to an address provided by the deployer.

Description:
The contract does not supply any tokens to the customer and simply forwards the funds to an address provided by the deployer. The contract does not record how much each user has supplied. However events do record each transaction amount.

Proposed Fix:
This may be intentional. However, in the event that there are disputes, the contract should record an amount supplied by each customer so that later tokens can be allocated in proportion to the amount of USD transferred.

Resolution:

Acknowledged that this is intentional.

Issue Number: 2

Severity: Medium
File: https://github.com/Kommissar29/contracts/blob/main/USDCForwarder.sol#L49

Summary:
Function `transferFrom` does not handle non-standard ERC20 tokens properly.

Description:
The `forwardUSD` function uses `transferFrom` from the ERC20 interface which is expected to return a boolean. Some tokens (like USDT) don't correctly implement the EIP20 standard and their transfer/ transferFrom function returns nothing instead of a success boolean, this would cause forwardUSD() to fail for USDT.

Proposed Fix:
Incorporate the `SafeERC20` library from OpenZeppelin to handle interactions with ERC20 tokens, which will safely manage tokens like USDT that may not follow the standard return pattern. Update the function to use `safeTransferFrom` provided by the `SafeERC20` library.

```
import "@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol";
contract USDForwarder {
   using SafeERC20 for IERC20;
   // Existing contract code...
   function forwardUSD(uint256 numPurchases) external whenActive {
   // Existing contract code...
      IERC20(usdToken).safeTransferFrom(msg.sender, recipient, totalAmount);
   // Existing contract code...
   }
}
```

Additional Recommendations:

Add a function to remove any tokens sent by accident. This is safe since funds are never stored in this contract.

```
/** * @dev Allows the owner to recover tokens mistakenly sent to this contract. * @param
tokenAddr The address of the ERC20 token contract. * @param amount The amount of
tokens to recover. */
function recoverTokens(address tokenAddr, uint256 amount) external onlyAdmin {
        IERC20 token = IERC20(tokenAddr);
token.safeTransfer(msg.sender, amount);
}
```

Resolution:

Proposed fix is implemented.