SALUS SECURITY

FEB 2023

CODE SECURITY ASSESSMENT

UBILOAN

Overview

Project Summary

- Name: Ubiloan
- Version: Commit a6075fc
- Platform: Ethereum / Binance Smart Chain
- Language: Solidity
- Repository: https://github.com/alrite-web3/alrite-dao-eth
- Audit Range: See Appendix - 1

Project Dashboard

Application Summary

Name	UbiloanToken
Version	v2
Type	Solidity
Dates	Feb 28 2023
Logs	Feb 24 2023, Feb 28 2023

Vulnerability Summary

Total High-Severity issues	0
Total Medium-Severity issues	0
Total Low-Severity issues	1
Total informational issues	2
Total	3

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 **4**
1.1 About SALUS 4
1.2 Audit Breakdown 4
1.3 Disclaimer 4

Findings **5**
2.1 Summary of Findings 5
2.2 Notable Findings 6
1. Centralization risk 6
2.3 Informational Findings 7
2. Floating compiler version 7
3. Redundant code 8

Appendix **9**
Appendix 1 - Files in Scope 9

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	Centralization risk	Low	Centralization	Acknowledged
2	Floating compiler version	Informational	Configuration	Acknowledged
3	Redundant code	Informational	Redundancy	Resolved

2.2 Notable Findings

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

1. Centralization risk

Severity: Low	Category: Centralization
Target: - contracts/UbiloanToken.sol	

Description

UbiloanToken.sol:L13-L15

```
constructor() ERC20Permit("Ubiloan Token") ERC20("Ubiloan Token", "UNT") {
    _mint(msg.sender, INITIAL_SUPPLY * 10 ** 18);
}
```

All tokens are minted to the deployer's address during contract deployment, which is a centralization risk. If the deployer's private key is compromised, the tokens are at risk.

Recommendation

Consider using a multi-sig wallet to deploy this contract.

Status

This issue has been acknowledged by the team.

2.3 Informational Findings

2. Floating compiler version

Severity: Informational	Category: Configuration
Target: - contracts/UbiloanToken.sol	

Description

UbiloanToken.sol:L2

```
pragma solidity ^0.8.17;
```

The UbiloanToken contracts use a floating compiler version ^0.8.17.

Contracts should be deployed with the same compiler version and flags that they have been thoroughly tested with. Locking the pragma helps to ensure that contracts do not accidentally get deployed using, for example, an outdated compiler version that might introduce bugs that affect the contract system negatively.

Recommendation

It is recommended to use a locked Solidity version throughout the project.

Status

This issue has been acknowledged by the team.

3. Redundant code

Severity: Informational	Category: Redundancy
Target: - contracts/UbiloanToken.sol	

Description

UbiloanToken.sol:L4

```
import "@openzeppelin/contracts/access/Ownable.sol";
```

UbiloanToken.sol:L9

```
contract UbiloanToken is ERC20Votes, Ownable {
```

The **Ownable.sol** contract is imported but not used in the UbiloanToken contract; thus, it can be removed.

Recommendation

Consider removing the **Ownable** contract.

Status

This issue has been resolved by the team by removing the Ownable contract in commit 36fbd25.

Appendix

Appendix 1 - Files in Scope

This audit covered the following file in commit a6075fc:

File	SHA-1 hash
contracts/token/UbiloanToken.sol	3238a5074ff6b94459a880a094e5d7801c7a2ac6