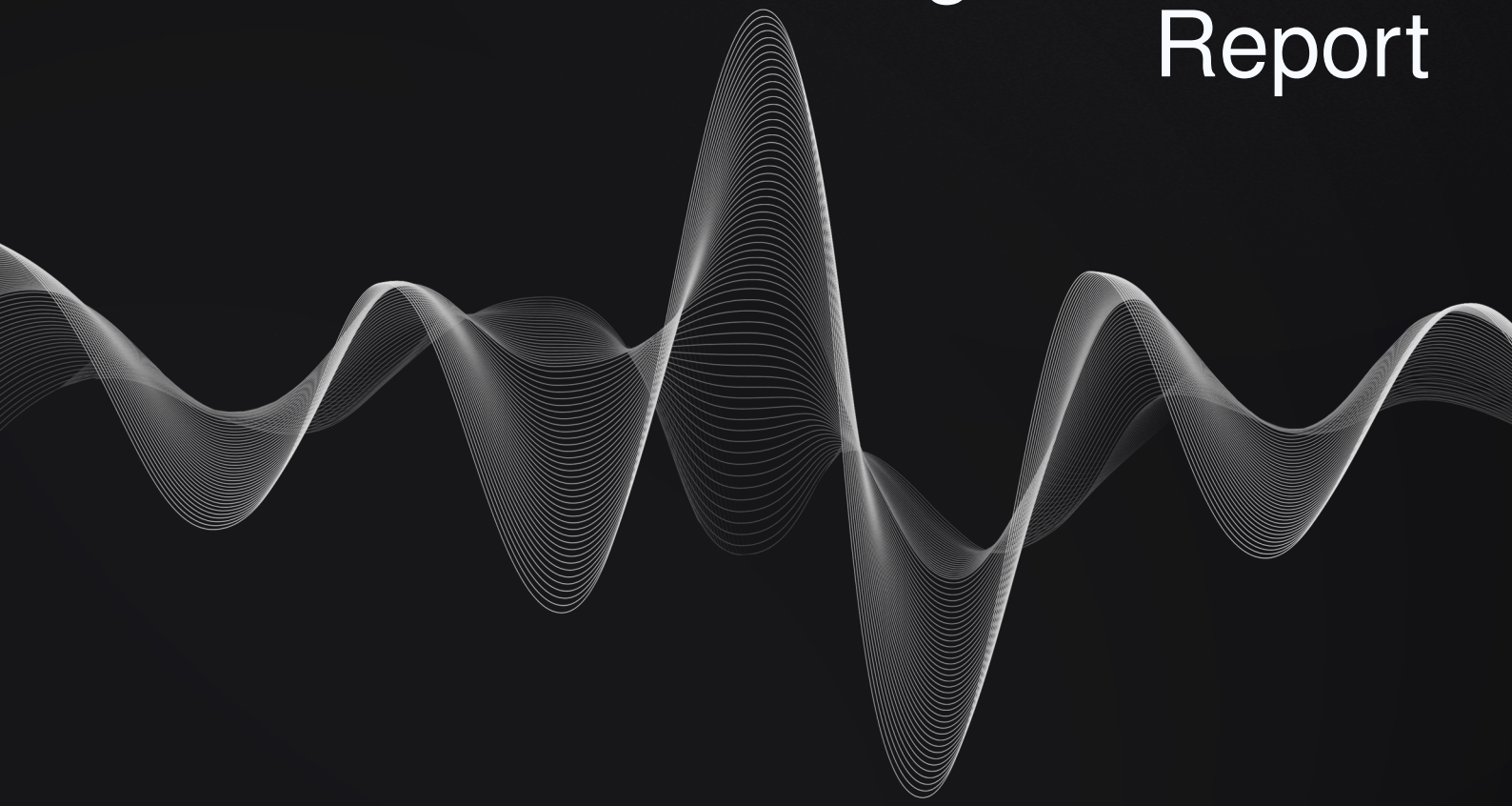
RESONANCE

bloq

Bloq

Metronome Synth AMO Integration Audit Report



Blockchain, Emerging Technology, and Web2
CYBERSECURITY PRODUCT & SERVICE ADVISORY

Document Control

CONFIDENTIAL **FINAL**(v2.0)

Audit_Report_BLOQ-AMO_FINAL_20

Sep 18, 2024	v0.1	Michał Bazyli: Initial draft
Sep 20, 2024	v0.2	João Simões: Added findings
Sep 30, 2024	v1.0	Charles Dray: Approved
Oct 16, 2024	v1.1	João Simões: Reviewed findings
Oct 23, 2024	v2.0	Charles Dray: Finalized

Points of Contact	Manoj Patidar	Bloq	manoj@bloq.com
	Charles Dray	Resonance	charles@resonance.security
Testing Team	Michał Bazyli	Resonance	michal@resonance.security
	João Simões	Resonance	joao@resonance.security
	Ilan Abitbol	Resonance	ilan@resonance.security

Copyright and Disclaimer

Contents

Executive Summary

Bloq contracted the services of Resonance to conduct a comprehensive security reaudit of their smart contracts between September 16, 2024 and September 30, 2024.The primary objective of the assessment was to identify any potential security vulnerabilities and ensure the correct functioning of smart contract operations.

During the engagement, Resonance allocated 2 engineers to perform the security review. The engineers, including an accomplished professional with extensive proficiency in blockchain and smart-contract security, encompassing specialized skills in advanced penetration testing, and in-depth knowledge of multiple blockchain protocols, devoted 11 days to the project. The project's test targets, overview, and coverage details are available throughout the next sections of the report.

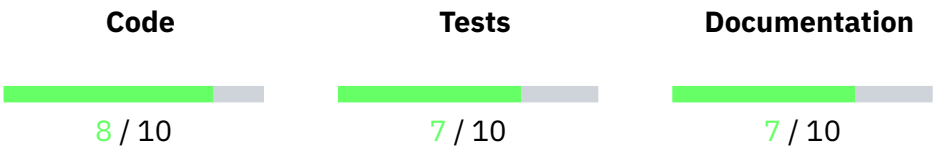
The ultimate goal of the audit was to provide Bloq with a detailed summary of the findings, including any identified vulnerabilities, and recommendations to mitigate any discovered risks. The results of the audit are presented in detail further below.

System Overview

Metronome Synth is a decentralized finance (DeFi) multi-collateral and multi-synthetic protocol. Through the Metronome dApp, users are able to deposit crypto assets as collateral, and use that collateral to mint popular crypto synthetics. These synthetics allow users to perform slippage free trades (swaps) and engage in yield farming. This protocol operates on Ethereum and Optimism.

Metronome has included bridging/cross-chain capabilities through the usage of the LayerZero protocol where messages containing funds are sent cross-chain. The relevant swaps occur on LayerZero's mainnet and all liquidity is maintained in native tokens. Additionally, an airdrop reward system was implemented based on merkle tree proofs.

Repository Coverage and Quality



Resonance's testing team has assessed the Code, Tests, and Documentation coverage and quality of the system and achieved the following results:

- The code follows development best practices and makes use of known patterns, standard libraries, and language guides. It is easily readable but does not use the latest stable version of relevant components. Overall, **code quality is good**.
- Unit and integration tests are included. The tests cover both technical and functional requirements. Code coverage is undetermined. Overall, **tests coverage and quality is good**.

- The documentation only includes the specification of the system and relevant explanations of workflows and interactions. Overall, **documentation coverage and quality is good**.

Target

The objective of this project is to conduct a comprehensive review and security analysis of the smart contracts that are contained within the specified repository.

The following items are included as targets of the security assessment:

- Repository: autonomoussoftware/metronome-synth/contracts
- Hash: 57fcfbf25996dd882c6842c7c526ade95c96c949

The following items are excluded:

- External and standard libraries
- Files pertaining to the deployment process
- Financial-related attack vectors

Methodology

In the context of security audits, Resonance's primary objective is to portray the workflow of a real-world cyber attack against an entity or organization, and document in a report the findings, vulnerabilities, and techniques used by malicious actors. While several approaches can be taken into consideration during the assessment, Resonance's core value comes from the ability to correlate automated and manual analysis of system components and reach a comprehensive understanding and awareness with the customer on security-related issues.

Resonance implements several and extensive verifications based off industry's standards, such as, identification and exploitation of security vulnerabilities both public and proprietary, static and dynamic testing of relevant workflows, adherence and knowledge of security best practices, assurance of system specifications and requirements, and more. Resonance's approach is therefore consistent, credible and essential, for customers to maintain a low degree of risk exposure.

Ultimately, product owners are able to analyze the audit from the perspective of a malicious actor and distinguish where, how, and why security gaps exist in their assets, and mitigate them in a timely fashion.

Source Code Review - Solidity EVM

During source code reviews for Web3 assets, Resonance includes a specific methodology that better attempts to effectively test the system in check:

1. Review specifications, documentation, and functionalities
2. Assert functionalities work as intended and specified
3. Deploy system in test environment and execute deployment processes and tests
4. Perform automated code review with public and proprietary tools
5. Perform manual code review with several experienced engineers
6. Attempt to discover and exploit security-related findings
7. Examine code quality and adherence to development and security best practices
8. Specify concise recommendations and action items
9. Revise mitigating efforts and validate the security of the system

Additionally and specifically for Solidity EVM audits, the following attack scenarios and tests are recreated by Resonance to guarantee the most thorough coverage of the codebase:

- Reentrancy attacks
- Frontrunning attacks
- Unsafe external calls
- Unsafe third party integrations
- Denial of service
- Access control issues

- Inaccurate business logic implementations
- Incorrect gas usage
- Arithmetic issues
- Unsafe callbacks
- Timestamp dependence
- Mishandled panics, errors and exceptions

Severity Rating

Security findings identified by Resonance are rated based on a Severity Rating which is, in turn, calculated off the **impact** and **likelihood** of a related security incident taking place. This rating provides a way to capture the principal characteristics of a finding in these two categories and produce a score reflecting its severity. The score can then be translated into a qualitative representation to help customers properly assess and prioritize their vulnerability management processes.

The **impact** of a finding can be categorized in the following levels:

1. Weak - Inconsequential or minimal damage or loss
2. Medium - Temporary or partial damage or loss
3. Strong - Significant or unrecoverable damage or loss

The **likelihood** of a finding can be categorized in the following levels:

1. Unlikely - Requires substantial knowledge or effort or uncontrollable conditions
2. Likely - Requires technical knowledge or no special conditions
3. Very Likely - Requires trivial knowledge or effort or no conditions

		Likelihood		
		Very Likely	Likely	Unlikely
Impact	Strong	Critical	High	Medium
	Medium	High	Medium	Low
	Weak	Medium	Low	Info

Repository Coverage and Quality Rating

The assessment of Code, Tests, and Documentation coverage and quality is one of many goals of Resonance to maintain a high-level of accountability and excellence in building the Web3 industry. In Resonance it is believed to be paramount that builders start off with a good supporting base, not only development-wise, but also with the different security aspects in mind. A product, well thought out and built right from the start, is inherently a more secure product, and has the potential to be a game-changer for Web3's new generation of blockchains, smart contracts, and dApps.

Accordingly, Resonance implements the evaluation of the code, the tests, and the documentation on a score **from 1 to 10** (1 being the lowest and 10 being the highest) to assess their quality and coverage. In more detail:

- Code should follow development best practices, including usage of known patterns, standard libraries, and language guides. It should be easily readable throughout its structure, completed with relevant comments, and make use of the latest stable version components, which most of the times are naturally more secure.
- Tests should always be included to assess both technical and functional requirements of the system. Unit testing alone does not provide sufficient knowledge about the correct functioning of the code. Integration tests are often where most security issues are found, and should always be included. Furthermore, the tests should cover the entirety of the codebase, making sure no line of code is left unchecked.
- Documentation should provide sufficient knowledge for the users of the system. It is useful for developers and power-users to understand the technical and specification details behind each section of the code, as well as, regular users who need to discern the different functional workflows to interact with the system.

Findings

During the security audit, several findings were identified to possess a certain degree of security-related weaknesses. These findings, represented by unique IDs, are detailed in this section with relevant information including Severity, Category, Status, Code Section, Description, and Recommendation. Further extensive information may be included in corresponding appendices should it be required.

An overview of all the identified findings is outlined in the table below, where they are sorted by Severity and include a **Remediation Priority** metric asserted by Resonance's Testing Team. This metric characterizes findings as follows:

- **"Quick Win"** Requires little work for a high impact on risk reduction.
- **"Standard Fix"** Requires an average amount of work to fully reduce the risk.
- **"Heavy Project"** Requires extensive work for a low impact on risk reduction.

RES-01	Incorrect Accounting Of amoSupply When AMO Is EOA		Acknowledged
RES-02	Integer Underflow On amoSupply		Acknowledged

Incorrect Accounting Of amoSupply When AMO Is EOA

Low | **RES-BLOQ-AMO01** | Data Validation | **Acknowledged**

Code Section

- contracts/SyntheticToken.sol
- contracts/SyntheticToken.sol

Description

The function `_mint()` allows automated market operators to mint synthetic tokens to any account provided as an input parameter, however, the function `_burn()` only allows the burning of tokens for the AMO itself. This creates a condition where it is not possible to burn other accounts' tokens, leaving the variable `amoSupply` always positive. This scenario is only possible if the `msg.sender` `amo` is an EOA account., otherwise, proper hardcoded input parameters are being used on the smart contract `AMO`, i.e. the usage of `address(this)` during minting and burning.

Recommendation

It is recommended to implement a validation to ensure that an automated market operator `msg.sender` is the same as the input parameter `account` during minting, as it is being done during the burning process.

Status

The issue was acknowledged by Bloq's team. The development team stated "Even if AMO address mint for some account, we do not want to give authority to AMO to burn for any address. AMO is suppose to get synth in account to burn from AMOSupply. AMO functionality is minimalist feature and its trusted role.".

Integer Underflow On amoSupply

Info | **RES-BLOQ-AMO02** | Arithmetic Issues | **Acknowledged**

Code Section

- contracts/SyntheticToken.sol
- contracts/AMO.sol

Description

The function `_burn()` does not verify that the variable `amount_` can be subtracted from the variable `amoSupply`, enabling the possibility of causing an integer underflow condition. This is especially impactful if the `msg.sender amo` is an EOA account.

Recommendation

It is recommended to implement a verification against an integer underflow condition. In this specific case, such condition is being checked on the smart contract `AMO`, whereas it should be moved to the contract `SyntheticToken` to account for EOA automated market operators.

Status

The issue was acknowledged by Bloq's team. The development team stated "AMO role is trusted and we assuming AMO will not mess-up with AMO supply accounting.".

Proof of Concepts

No Proof-of-Concept was deemed relevant to describe findings in this engagement.