SMART CONTRACTS

TABLE OF CONTENTS

ABOUT TRUTH SECURITY

Truth Security Audits conducts a comprehensive security review of blockchain applications, using modern tools and employing only the most experienced solidity experts on the market. During the elaboration of the audit, auditors analyze possible attack vectors both from the project owners and its users and rate them by Severity (see Appendix A) from Informational to Critical as per common reason, giving an adequate explanation in the body of the audit. Our Audits are versioned, and clients get a grace period to alleviate or comment on all of our findings.

METHODOLOGY

As per standard code review practices, we use manual and static analysis. During the manual phase, auditor(s) review source code line-by-line, studying its intended and actual behavior, referencing known vulnerabilities (including SWC Registry https://swcregistry.io/), comparing the code to common contracts, and noting all things that are out of the ordinary for confirmation or rejection. Static analysis refers to a computer-aided analysis of the code, providing automatized and powerful insight into additional subtle issues possibly present in the code.

Some auditors additionally write their own test cases and try to break the contracts in their own local simulated blockchain environment. This is called dynamic analysis and is often employed in the more complex contracts, where consolidated testing scenarios help assess the completeness of contract logic or reversely give a proof-of-concept for potential security vulnerability.

Finally, all auditors do on-chain verification of live contracts. This crucial step confirms contracts were not swapped for malicious only after audit, or that parameters are set based on reasonable expectations. Issues such as non-renounced ownerships are also assessed in this step.

PROJECT DETAILS

Project Name	EMP
Description	EMP.Money is a revolutionary DeFi protocol on the BNB Chain. EMP's token value is algorithmically pegged to the price of Ethereum via seigniorage at a rate of 4000 EMP:1 ETH
Links	https://emp.money/
Code Language	Solidity
Chain	Ethereum

VERSION DETAILS

Version	Based on status at	Published at	Elaborated by	Notes
V	December 8th, 2022	December 8th, 2022	December 8th, 2022	

OBJECTS OF REVIEW

In Versions	Source	Contents
v.		All Contracts
		https://github.com/empm0ney

SUMMARY

AUDIT PROCEDURE

Auditors	RMIII
Audited as	Truth Seekers
Methodology	Static Analysis + Dynamic Analysis Static analysis examines code to identify issues within the logic and techniques. This type of analysis addresses weaknesses in source code that might lead to vulnerabilities. Static analysis supports d operations by creating an automated feedback loop. Static review can lead to false positives/negatives which Truth Seekers confirms or denies with penetration testing. Static analysis is being used on because it specifically identifies defects before you run a program (e.g, between coding and unit testing)
Tools	Dynamic analysis finds vulnerabilities in a runtime environment. Automated tools analyze the input and output of an application for potential threats like SQL injection. Tools can also search for other ap issues and analyze server configuration errors. The purpose of dynamic analysis is to analyze the program as an attacker would, looking for entry points and vulnerable sections during program execu create a well written suite of manually ran unit and regression tests,

Truth Security Audits has no control over website UI projects provide. Always double check you are signing a contract matching one of the contracts in section Objects of Review.

Truth Security Audits concerns itself exclusively with code quality and smart contract security.
We have not audited tokenomics, nor a general likelihood of making money with this project. Truth Security report is not financial advice.

FINDINGS

Finding ID	001	Severity	High
Type	No Slippage Protection	Status	Unresolved
Location	Zap.sol - L: 964 - 969		
Description	The zapper contracts do not check for slippage on swaps and can be vulnerable to price manipulation. A user can be front-run if swapping a large amount or be affected by a very high price impact without noticing. Additionally, a malicious router can cause users to lose all funds. A high price impact can also result in the loss of some output tokens		
Recommendation	Add price impact/slippage protection.		
Alleviation			

Finding ID	002	Severity	High
Type	Stale Price	Status	Unresolved
Location	Oracle.Sol: L: 54-73		
Description	A Uniswap LP oracle is expected to make use of overflowing to deal with cumulative prices. However, from solidity 0.8.0, basic arithmetic operations automatically revert on overflow. If the update fails because of this, the oracle price will no longer update as expected and the treasury will operate with a stale price.		
Recommendation	Add unchecked to the oracle logic where expected.		
Alleviation	-		

Finding ID	003	Severity	Medium
Type	Unsupported Token Recovery is Unrestricted after pools expire	Status	Unresolved
Location	EShareReward Pool: L: 262-272		
Description	After the governance pools have closed, the operator will be able to transfer any remaining tokens out after a period of days to an arbitrary address		
Recommendation	Remove the conditional such that users' pool tokens can never be recovered from the pool.		
Alleviation			

Finding ID	004	Severity	Medium
Type	Reentrancy Risk Using Token with Callbacks	Status	Unresolved
Location	EShareReward pool: L: 172 - 191		
Description	If a token has callbacks (such as anonReceivedfunction in the standard ERC721), then a malicious actor may be able to drain the contract of tokens either via the withdraw or deposit function.		
Recommendation	Follow the checks-effects-interactions pattern. Alternatively, add a reentrancy guard		
Alleviation			

Finding ID	005	Severity	Medium
Type	Initialize Can be Called by Anyone	Status	Unresolved
Location	Treasury.Sol: L: 236-243		
Description	Anyone can initialize the contract and become the operator if the deployer does not initialize in the same transaction.		
Recommendation	If the Treasury need to be changed make sure that the initializer is protected		
Alleviation			

Finding ID	006	Severity	Low
Type	Team Can Add New Team Pool	Status	Unresolved
Location	EShareReward pool: L: 124-132		
Description	A require statement is used to prevent the operator from modifying the team's allocation pool. However, the operator can deploy a new token and pool and effectively bypass this restriction.		
Recommendation	Put a limit on adding a new pool. Otherwise, remove the require statement as redundant.		
Alleviation			

Finding ID	007	Severity	Informational
Type	No limits for discounts	Status	Unresolved
Location	Treasury.Sol: L: 361-365		
Description	The noted setters have no limits and can therefore change themaxDiscountRate and maxPremiumRateto any value.		
Recommendation	Add a reasonable limit for these values.		
Alleviation			

Finding ID	008	Severity	Informational
Type	Dependent Code	Status	Unresolved
Location	lib/UniswapV2Library.sol: L: 19-32		
Description	The init code hash used in the *pairFor* function is specific to the Uniswap on Ethereum mainnet and will not correctly calculate pairs when deployed on other networks.		
Recommendation	Update the library to be compatible with the DEX and chain it will be deployed to.		
Alleviation			

Finding ID	009	Severity	Informational
Type	No Lower Limit For Lockup Epochs	Status	Unresolved
Location	Boardroom.Sol: L: 141-144.		
Description	It is possible for the operator to set the withdrawLockupEpochs and rewardLockupEpochs to 0, withdraw their rewards, then change it back		
Recommendation	Add a lower bound to prevent this. A timelock can also resolve this issue.		
Alleviation			

Finding ID	010	Severity	Informational
Type	Oracle may Drift	Status	Unresolved
Location	Oracle.Sol: L: 54-72		
Description	The checkEpoch modifier ensures that the contract is only updated within a certain epoch. If the oracle is updatedafter the epoch starts, then the OraclePERIOD may cause the update to occur later and later relative to the epoch start.		
Recommendation	Ensure that this is intended behavior as the Oracle update could occur near the end of an epoch or could miss an epoch entirely.		
Alleviation			

APPENDIX A: ISSUE SEVERITY CLASSIFICATION

The Auditor(s) assign one of the following Severities to every finding as per common practice.

Critical. Such issues may result in significant loss of funds, complete contract logic breakdown, or the ability of project owners to withdraw liquidity in an unreasonable way. Code snippets proving the project owner's malicious intent are flagged as critical as well. Critical issues require immediate attention, and investing in projects with critical issues is extremely risky. Examples include unguarded mint functions or their executions, provably illicit pool drainage logic, or potential flash-loan vulnerabilities.

Major. Although not proving malicious intent by themselves, major issues may still be exploited by project owners or users for a significant loss of funds or very irregular contract behavior. Examples include centralized ownership without Timelocks and multi-signatures, potential reentrancy vulnerabilities, and concentrated holdings of tokens.

Medium. Such issues do not pose an immediate and severe risk but may pose a risk of partial loss of funds or irregular contract behavior. Examples include susceptibility to obviously unintended investment strategies, high-impact integer overflows, or high-impact standardization faults such as library usage,

Minor. These issues pose a low risk to contract logic or investor funds but may be convenient to consider. Examples include integer overflows in non-essential places, nonversioned libraries, missing or faulty licensing, misleading function names, or low-impact standardization mistakes.

Informational. These issues do not pose any risk to contract logic and investor funds, Examples include tokenomics clarifications, gas optimizations, redundant code, misleading comments, style, and convention.

Confirmational. In specific situations, we issue these findings, which confirm some of the universally-concerning facts that many investors seek. Examples include contract renounces and confirmation of a contract being fork of another protocol. Note: These points are not actual issues. Obviously, only a small subset of tests ran in an audit suite receives its Confirmational Finding.

APPENDIX B: DISCLAIMER

This audit is for informational purposes only and does not provide any financial or investment advice. This report does not substitute, in any way, due diligence and your own research. This report represents result extensive process intending to help our customers improve quality of their code and readers to assess quality of customers' code, but should not be used in any way to make decisions around involvement in any particular project.

Audit has been done in accordance to methodology as outlined in AboutTruth Security and Audit Procedure sections Unless explicitly and specifically stated, only code quality has been reviewed, focusing on security flaws which could cause loss of funds or logical breakdowns within the contracts. Unless explicitly stated, tokenomics have not been reviewed (although in cases of forks of one project, Auditor may point out cases of significant deviations from common settings). Website UI has not been reviewed, as it is impossible for any auditing body to assure security of domains which are under absolute control of owners - always check you are signing correct contracts.

The report does not signify an approval," „endorsement," or ,,disapproval of the Project The audit does not indicate in any way your likelihood of making, or not losing, money in the project, as we have no control over issues such as general viability of financial primitives presented, their tokenomics, and actions of project owners including, but not limited to, selling their positions or abandoning the project.

The audit has been based on status dated in section Version Details, on artifacts detailed in Objects of Review. Specifically, we have no control nor knowledge of changes made after the date, or on different artifacts. In case the Objects of Review are not live contracts, but private code or GitHub repositories, we expect these artifacts to be full, unaltered, unabridged, and not misleading.

The audit has been elaborated by paid professional(s) as mentioned in section Audit Procedure. Please note that all statements made in this report are Auditor(s)' and do not reflect stance of © Truth Security Audits itself.