



Improbable

M2 Token

SMART CONTRACT AUDIT

14.09.2022

Made in Germany by Chainsulting.de



Table of contents

1. Disclaimer.....	4
2. About the Project and Company	5
2.1 Project Overview.....	6
3. Vulnerability & Risk Level	7
4. Auditing Strategy and Techniques Applied.....	8
4.1 Methodology	8
5. Metrics	9
5.1 Tested Contract Files	9
5.2 Used Code from other Frameworks/Smart Contracts	10
5.3 CallGraph	13
5.4 Inheritance Graph	15
5.5 Source Lines & Risk	16
5.6 Capabilities	17
5.7 Source Unites in Scope	18
6. Scope of Work.....	20
6.1 Findings Overview	21
6.2 Manual and Automated Vulnerability Test.....	22
6.2.1 Floating Pragma Version Identified	22
6.2.2 Empty Code Blocks.....	23
6.2.3 Redundant Code	23
6.3 SWC Attacks	24
6.4 Verify Claims	28

6.5 Test Cases.....	28
6.6 Test Coverage.....	33
7. Executive Summary.....	34
8. Mainnet Contract.....	34
9. About the Auditor	35



1. Disclaimer

The audit makes no statements or warranties about utility of the code, safety of the code, suitability of the business model, investment advice, endorsement of the platform or its products, regulatory regime for the business model, or any other statements about fitness of the contracts to purpose, or their bug free status. The audit documentation is for discussion purposes only.

The information presented in this report is confidential and privileged. If you are reading this report, you agree to keep it confidential, not to copy, disclose or disseminate without the agreement of Improbable MV Limited. If you are not the intended receptor of this document, remember that any disclosure, copying or dissemination of it is forbidden.

Major Versions / Date	Description
0.1 (03.09.2022)	Layout
0.4 (06.09.2022)	Automated Security Testing Manual Security Testing
0.5 (09.09.2022)	Verify Claims and Test Deployment
0.6 (10.09.2022)	Testing SWC Checks
0.9 (13.09.2022)	Summary and Recommendation
1.0 (14.09.2022)	Final document
1.1 (TBA)	Deployment to Mainnet

2. About the Project and Company

Company address:

Improbable MV Limited
10 Bishops Square
London, E1 6EG
United Kingdom



Website: <https://www.improbable.io>

Twitter: <https://twitter.com/Improbableio>

Facebook: <https://www.facebook.com/improbable.io>

LinkedIn: <https://www.linkedin.com/company/improbable>

GitHub: <https://github.com/improbable>

YouTube: <https://www.youtube.com/channel/UC7BE8B2yUeQxPvZytk47NYw>

2.1 Project Overview

The M² network will combine Improbable Morpheus technology with new services designed to support interoperability, commerce in digital assets and governance in Web3. It will bring together companies, existing communities and fans in sports, music, fashion and entertainment and enable them to interact in dense virtual spaces with unprecedented fidelity. The network is being designed to support integration with existing worlds as well as new projects. Improbable has established M² as a distinct entity to better enable governance in partnership with other businesses and eventually with its community.

Improbable has for nearly a decade developed technology enabling ever greater complexity in virtual worlds: the company is a leading provider of multiplayer services to over 60 global publishers and through its large scale simulation platform supports the UK government defence mission. Its Morpheus technology, an evolution of the company's earlier SpatialOS product, supports over 10,000 players, able to interact with each other in dense virtual spaces. The platform now processes over 350 million communication operations per second (or ops) and was first demonstrated in live events with thousands of players in 2021. In January 2022, Improbable announced its transformation to accelerate in the metaverse.

3. Vulnerability & Risk Level

Risk represents the probability that a certain source-threat will exploit vulnerability, and the impact of that event on the organization or system. Risk Level is computed based on CVSS version 3.0.

Level	Value	Vulnerability	Risk (Required Action)
Critical	9 – 10	A vulnerability that can disrupt the contract functioning in a number of scenarios, or creates a risk that the contract may be broken.	Immediate action to reduce risk level.
High	7 – 8.9	A vulnerability that affects the desired outcome when using a contract, or provides the opportunity to use a contract in an unintended way.	Implementation of corrective actions as soon as possible.
Medium	4 – 6.9	A vulnerability that could affect the desired outcome of executing the contract in a specific scenario.	Implementation of corrective actions in a certain period.
Low	2 – 3.9	A vulnerability that does not have a significant impact on possible scenarios for the use of the contract and is probably subjective.	Implementation of certain corrective actions or accepting the risk.
Informational	0 – 1.9	A vulnerability that have informational character but is not effecting any of the code.	An observation that does not determine a level of risk

4. Auditing Strategy and Techniques Applied

Throughout the review process, care was taken to evaluate the repository for security-related issues, code quality, and adherence to specification and best practices. To do so, reviewed line-by-line by our team of expert pentesters and smart contract developers, documenting any issues as there were discovered.

4.1 Methodology

The auditing process follows a routine series of steps:

1. Code review that includes the following:
 - i. Review of the specifications, sources, and instructions provided to Chainsulting to make sure we understand the size, scope, and functionality of the smart contract.
 - ii. Manual review of code, which is the process of reading source code line-by-line in an attempt to identify potential vulnerabilities.
 - iii. Comparison to specification, which is the process of checking whether the code does what the specifications, sources, and instructions provided to Chainsulting describe.
2. Testing and automated analysis that includes the following:
 - i. Test coverage analysis, which is the process of determining whether the test cases are actually covering the code and how much code is exercised when we run those test cases.
 - ii. Symbolic execution, which is analysing a program to determine what inputs causes each part of a program to execute.
3. Best practices review, which is a review of the smart contracts to improve efficiency, effectiveness, clarify, maintainability, security, and control based on the established industry and academic practices, recommendations, and research.
4. Specific, itemized, actionable recommendations to help you take steps to secure your smart contracts.

5. Metrics

The metrics section should give the reader an overview on the size, quality, flows and capabilities of the codebase, without the knowledge to understand the actual code.

5.1 Tested Contract Files

The following are the MD5 hashes of the reviewed files. A file with a different MD5 hash has been modified, intentionally or otherwise, after the security review. You are cautioned that a different MD5 hash could be (but is not necessarily) an indication of a changed condition or potential vulnerability that was not within the scope of the review

File	Fingerprint (MD5)
./EggnogToken.sol	37004f7ca70f55eeaf31c775197ef3e6
./EggnogTimelockController.sol	d80b55d66c005003fea571f4a97266cd
./EggnogGovernor.sol	924786e2dc1f8c383122d65c4c335f13
./EggnogVesting.sol	e87081af18f8f90b3bf53eaab81b016c
./EggnogDAOTreasury.sol	4a324ee6225285ae17b8a6e6d3d6c110
./extensions/ERC777Votes.sol	3a272389e1f4057a56393c99f5778c67
./extensions/ERC777Permit.sol	add642a09502458fca8b1e9c16411534
./interfaces/IEggnogToken.sol	434e803c8cae418f1525f96fa8b374db
./interfaces/IEggnogTimelockController.sol	d4495dd6d1b6ad93e6822629574f97ca
./interfaces/IEggnogGovernor.sol	e95c13585de2db0092b775090e2dd7e0
./interfaces/IEggnogDAOTreasury.sol	36b9aa705745a87960c72a4b6d8aaa66
./interfaces/IEggnogVesting.sol	403b14e88c92b7cca1034dfe1ecdfa15

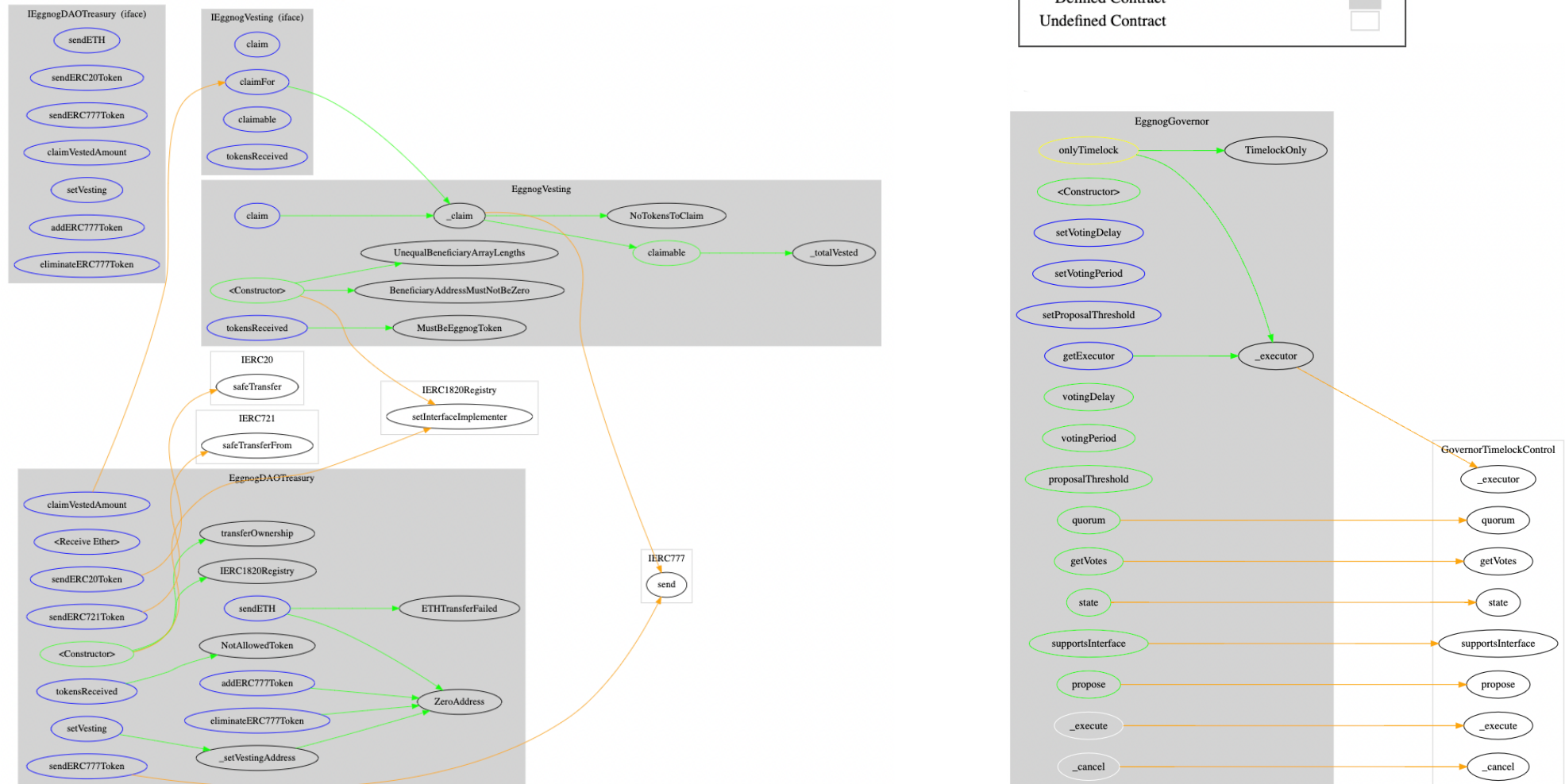
5.2 Used Code from other Frameworks/Smart Contracts (direct imports)

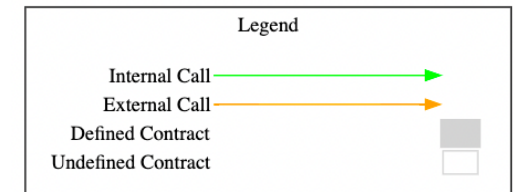
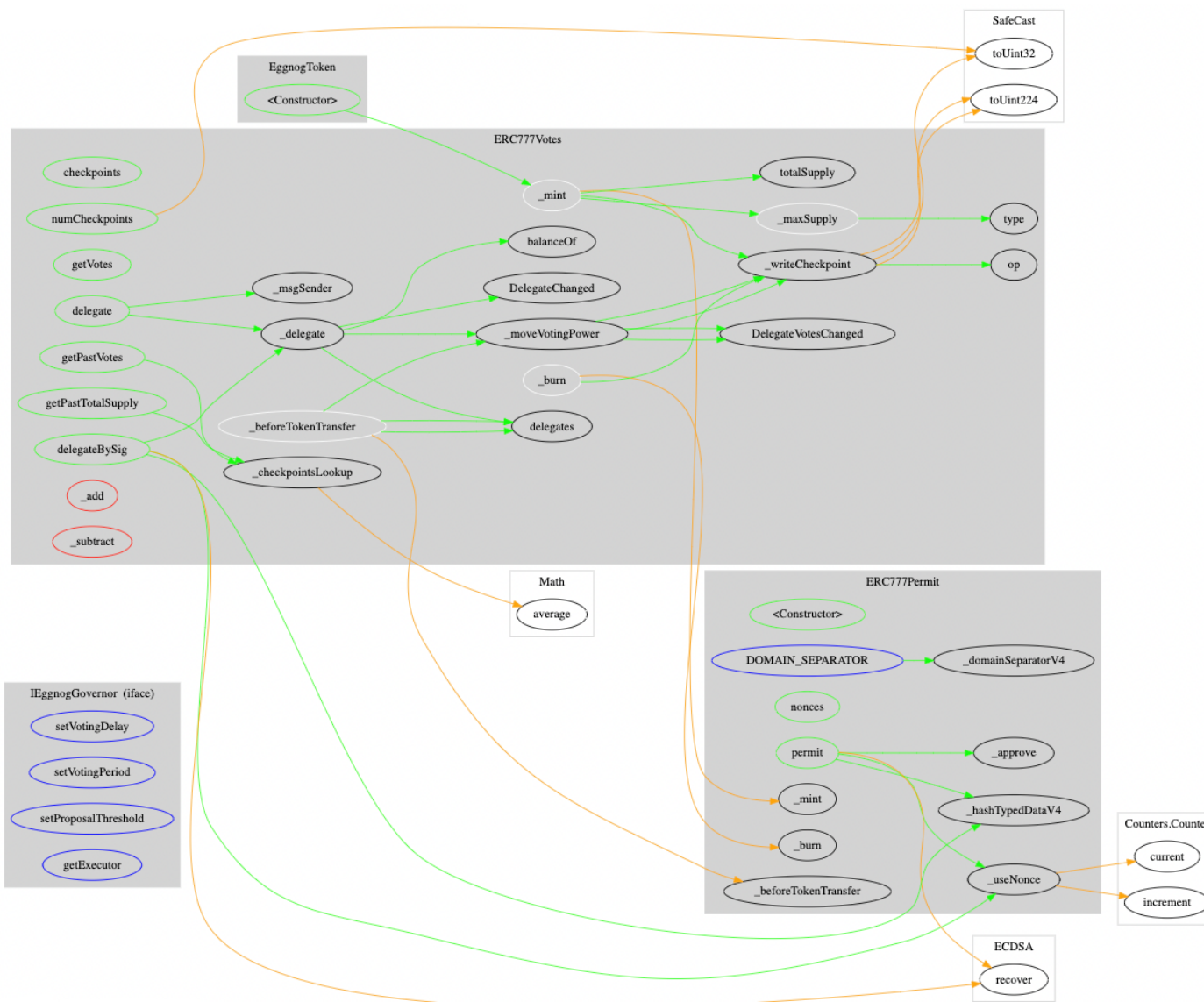
Dependency / Import Path	Source
@openzeppelin/contracts/access/Ownable.sol	https://github.com/OpenZeppelin/openzeppelin-contracts/tree/v4.7.3/contracts/access/Ownable.sol
@openzeppelin/contracts/governance/Governor.sol	https://github.com/OpenZeppelin/openzeppelin-contracts/tree/v4.7.3/contracts/governance/Governor.sol
@openzeppelin/contracts/governance/TimelockController.sol	https://github.com/OpenZeppelin/openzeppelin-contracts/tree/v4.7.3/contracts/governance/TimelockController.sol
@openzeppelin/contracts/governance/compatibility/GovernorCompatibilityBravo.sol	https://github.com/OpenZeppelin/openzeppelin-contracts/tree/v4.7.3/contracts/governance/compatibility/GovernorCompatibilityBravo.sol
@openzeppelin/contracts/governance/extensions/GovernorTimelockControl.sol	https://github.com/OpenZeppelin/openzeppelin-contracts/tree/v4.7.3/contracts/governance/extensions/GovernorTimelockControl.sol
@openzeppelin/contracts/governance/extensions/GovernorVotes.sol	https://github.com/OpenZeppelin/openzeppelin-contracts/tree/v4.7.3/contracts/governance/extensions/GovernorVotes.sol
@openzeppelin/contracts/governance/extensions/GovernorVotesQuorumFraction.sol	https://github.com/OpenZeppelin/openzeppelin-contracts/tree/v4.7.3/contracts/governance/extensions/GovernorVotesQuorumFraction.sol
@openzeppelin/contracts/governance/utils/IVotes.sol	https://github.com/OpenZeppelin/openzeppelin-contracts/tree/v4.7.3/contracts/governance/utils/IVotes.sol

Dependency / Import Path	Source
@openzeppelin/contracts/security/ReentrancyGuard.sol	https://github.com/OpenZeppelin/openzeppelin-contracts/tree/v4.7.3/contracts/security/ReentrancyGuard.sol
@openzeppelin/contracts/token/ERC20/IERC20.sol	https://github.com/OpenZeppelin/openzeppelin-contracts/tree/v4.7.3/contracts/token/ERC20/IERC20.sol
@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol	https://github.com/OpenZeppelin/openzeppelin-contracts/tree/v4.7.3/contracts/token/ERC20/utils/SafeERC20.sol
@openzeppelin/contracts/token/ERC721/IERC721.sol	https://github.com/OpenZeppelin/openzeppelin-contracts/tree/v4.7.3/contracts/token/ERC721/IERC721.sol
@openzeppelin/contracts/token/ERC777/ERC777.sol	https://github.com/OpenZeppelin/openzeppelin-contracts/tree/v4.7.3/contracts/token/ERC777/ERC777.sol
@openzeppelin/contracts/token/ERC777/IERC777.sol	https://github.com/OpenZeppelin/openzeppelin-contracts/tree/v4.7.3/contracts/token/ERC777/IERC777.sol
@openzeppelin/contracts/token/ERC777/IERC777Recipient.sol	https://github.com/OpenZeppelin/openzeppelin-contracts/tree/v4.7.3/contracts/token/ERC777/IERC777Recipient.sol
@openzeppelin/contracts/token/ERC777/IERC777Sender.sol	https://github.com/OpenZeppelin/openzeppelin-contracts/tree/v4.7.3/contracts/token/ERC777/IERC777Sender.sol
@openzeppelin/contracts/utils/Counters.sol	https://github.com/OpenZeppelin/openzeppelin-contracts/tree/v4.7.3/contracts/utils/Counters.sol

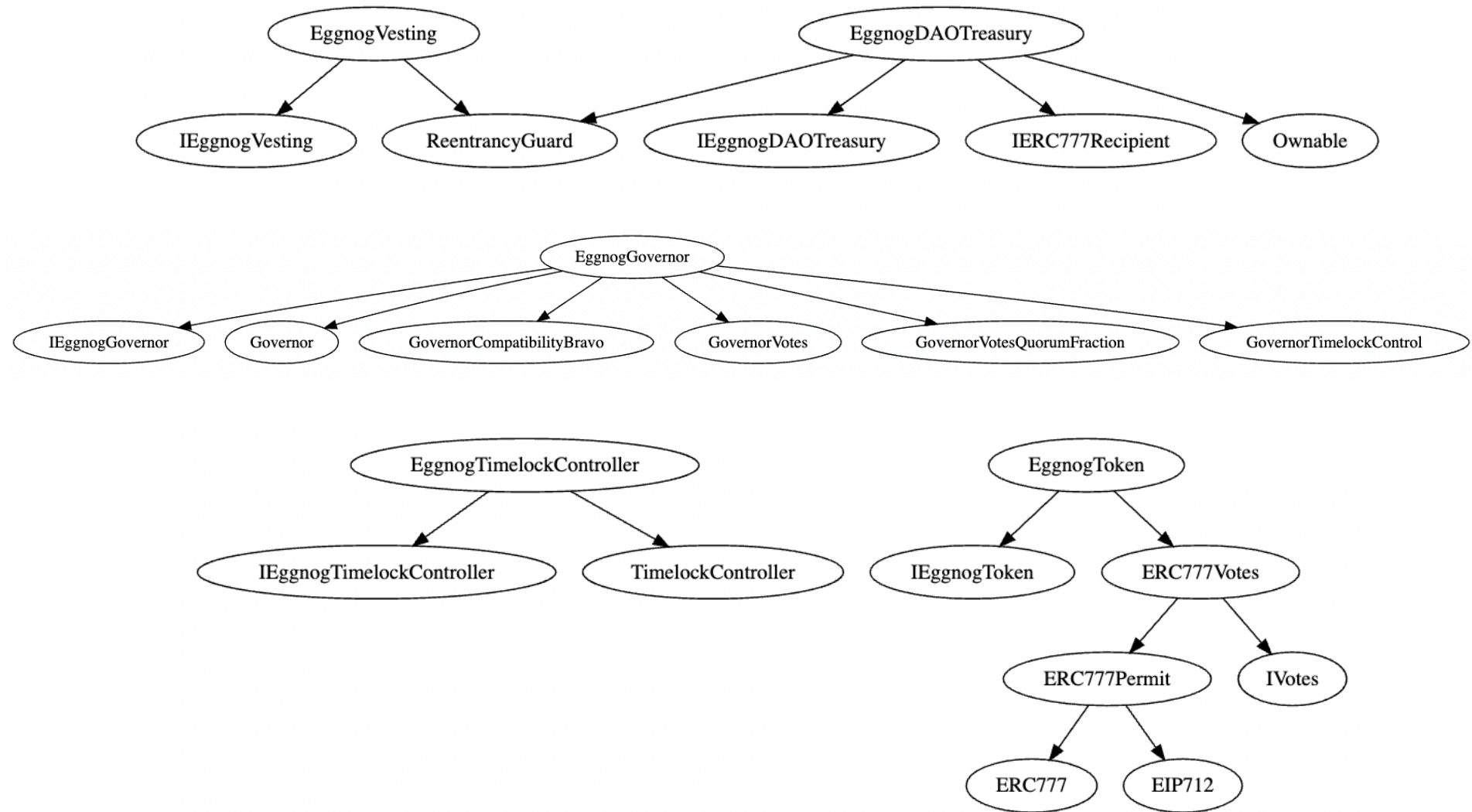
Dependency / Import Path	Source
@openzeppelin/contracts/utils/cryptography/ECDSA.sol	https://github.com/OpenZeppelin/openzeppelin-contracts/tree/v4.7.3/contracts/utils/cryptography/ECDSA.sol
@openzeppelin/contracts/utils/cryptography/draft-EIP712.sol	https://github.com/OpenZeppelin/openzeppelin-contracts/tree/v4.7.3/contracts/utils/cryptography/draft-EIP712.sol
@openzeppelin/contracts/utils/introspection/IERC1820Registry.sol	https://github.com/OpenZeppelin/openzeppelin-contracts/tree/v4.7.3/contracts/utils/introspection/IERC1820Registry.sol
@openzeppelin/contracts/utils/math/Math.sol	https://github.com/OpenZeppelin/openzeppelin-contracts/tree/v4.7.3/contracts/utils/math/Math.sol
@openzeppelin/contracts/utils/math/SafeCast.sol	https://github.com/OpenZeppelin/openzeppelin-contracts/tree/v4.7.3/contracts/utils/math/SafeCast.sol

5.3 CallGraph



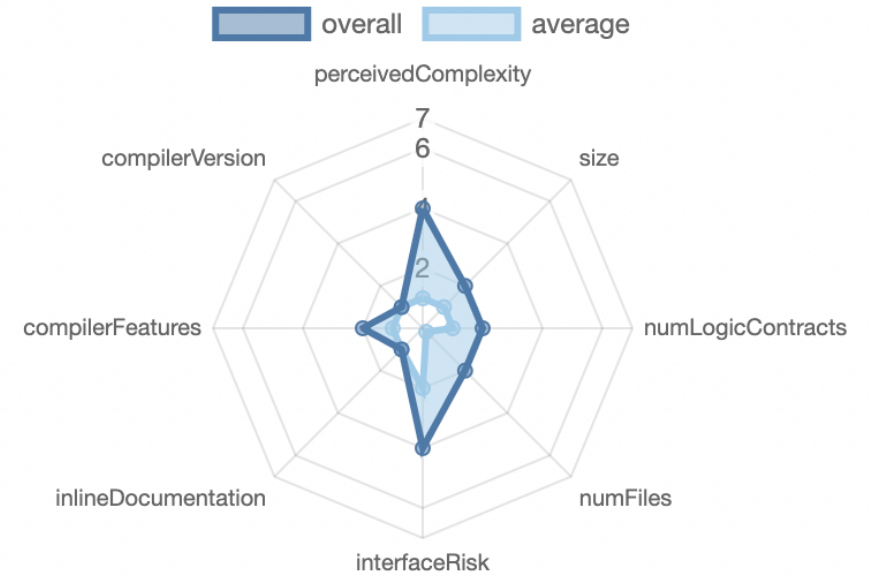


5.4 Inheritance Graph




5.5 Source Lines & Risk

source comment single block mixed
empty todo blockEmpty





5.6 Capabilities


Solidity Versions observed		 Experimental Features	 Can Receive Funds	 Uses Assembly	 Has Destroyable Contracts
<input type="text" value="^0.8.9"/>			<input type="text" value="yes"/>	<input type="text"/>	<input type="text"/>
 Transfers ETH	 Low-Level Calls	 DelegateCall	 Uses Hash Functions	 ECRrecover	 New/Create/Create2
<input type="text" value="yes"/>	<input type="text"/>	<input type="text"/>	<input type="text" value="yes"/>	<input type="text"/>	

Exposed Functions

This section lists functions that are explicitly declared public or payable. Please note that getter methods for public stateVars are not included.

 Public	 Payable				
<input type="text" value="52"/>	<input type="text" value="1"/>				
External	Internal	Private	Pure	View	
<input type="text" value="33"/>	<input type="text" value="58"/>	<input type="text" value="5"/>	<input type="text" value="2"/>	<input type="text" value="26"/>	












StateVariables

Total	 Public
<input type="text" value="17"/>	<input type="text" value="1"/>

5.7 Source Unites in Scope

Source: <https://github.com/improbable/m2-token>

Last commit: 5a3103ce381942eb4b663eaaafc278616de67148

Type	File	Logic Contracts	Interfaces	Lines	nLines	nSL OC	Comment Lines	Complex. Score	Capabilities
	contracts/interfaces/IEggnogVesting.sol	_____	1	58	35	12	26	9	_____
	contracts/interfaces/IEggnogDAOTreasury.sol	_____	1	62	24	12	19	15	_____
	contracts/interfaces/IEggnogGovernor.sol	_____	1	47	34	10	23	9	_____
	contracts/interfaces/IEggnogTimelockController.sol	_____	1	8	8	3	2	1	_____
	contracts/interfaces/IEggnogToken.sol	_____	1	11	11	5	2	1	_____
	contracts/EggnogDAOTreasury.sol	1	_____	163	144	83	32	82	
	contracts/EggnogVesting.sol	1	_____	124	117	67	27	59	
	contracts/extensions/ERC777Permit.sol	1	_____	98	90	31	45	28	

Type	File	Logic Contracts	Interfaces	Lines	nLines	nSLOC	Comment Lines	Complex. Score	Capabilities
	contracts/extensions/ERC777Votes.sol	1	_____	262	232	116	85	80	
	contracts/EggnogGovernor.sol	1	_____	126	110	71	19	60	_____
	contracts/EggnogTimelockController.sol	1	_____	17	17	10	4	6	_____
	contracts/EggnogToken.sol	1	_____	22	22	14	5	8	_____
	Totals	7	5	998	844	434	289	358	

Legend:

- **Lines:** total lines of the source unit
- **nLines:** normalized lines of the source unit (e.g. normalizes functions spanning multiple lines)
- **nSLOC:** normalized source lines of code (only source-code lines; no comments, no blank lines)
- **Comment Lines:** lines containing single or block comments
- **Complexity Score:** a custom complexity score derived from code statements that are known to introduce code complexity (branches, loops, calls, external interfaces, ...)

6. Scope of Work

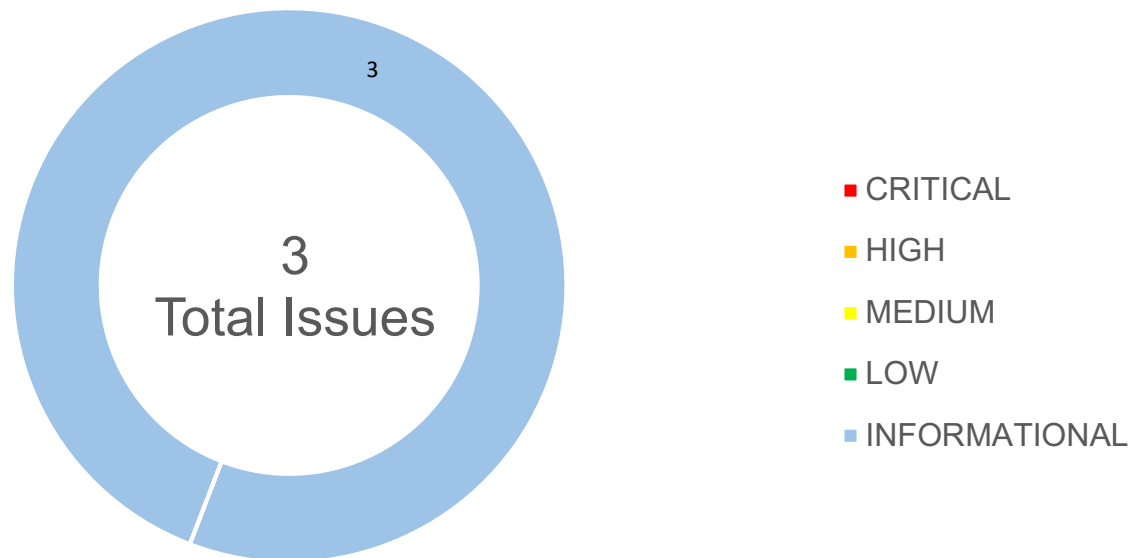
The Improbable Team provided us with the files that needs to be tested. The scope of the audit is the M2 token implementation as well as the DAO treasury and unlocking schedule contracts.

The team put forward the following assumptions regarding the security, usage of the contracts:

- The ERC-777 Token standard is correctly implemented
- Vesting is working as expected
- DAO Treasury is working as expected
- Governance is working as expected
- The smart contract is coded according to the newest standards and in a secure way.

The main goal of this audit was to verify these claims. The auditors can provide additional feedback on the code upon the client's request.

6.1 Findings Overview



No	Title	Severity	Status
6.2.1	Floating Pragma Version Identified	INFORMATIONAL	OPEN
6.2.2	Empty Code Blocks	INFORMATIONAL	OPEN
6.2.3	Redundant Code	INFORMATIONAL	OPEN

6.2 Manual and Automated Vulnerability Test

CRITICAL ISSUES

During the audit, Chainsulting's experts found **0 Critical issues** in the code of the smart contract.

HIGH ISSUES

During the audit, Chainsulting's experts found **0 High issues** in the code of the smart contract.

MEDIUM ISSUES

During the audit, Chainsulting's experts found **0 Medium issues** in the code of the smart contract.

LOW ISSUES

During the audit, Chainsulting's experts found **0 Low issues** in the code of the smart contract.

INFORMATIONAL ISSUES

During the audit, Chainsulting's experts found **3 Informational issues** in the code of the smart contract.

6.2.1 Floating Pragma Version Identified

Severity: INFORMATIONAL

Status: OPEN

Code: SWC-103

File(s) affected: ALL

Attack / Description
It is recommended to specify a fixed compiler version to ensure that the bytecode produced does not vary between builds. This is especially important if you rely

	on bytecode-level verification of the code.
Code	Line 2 <code>pragma solidity ^0.8.9;</code>
Result/Recommendation	It is recommended to follow the latter example, as future compiler versions may handle certain language constructions in a way the developer did not foresee. It is advised that floating pragma should not be used in production. Both truffle-config.js and hardhat.config.js support locking the pragma version. i.e. Pragma solidity 0.8.9

6.2.2 Empty Code Blocks

Severity: INFORMATIONAL

Status: OPEN

Code: NA

File(s) affected: interfaces/IEggnogTimelockController.sol, interfaces/IEggnogToken.sol

Attack / Description	The mentioned interfaces do not have any definitions and thus hold no information.
Code	NA
Result/Recommendation	It is recommended to remove unused interfaces to reduce repositories size

6.2.3 Redundant Code

Severity: INFORMATIONAL

Status: OPEN

Code: NA

File(s) affected: interfaces/IEggnogDAOTreasury.sol, EggnogDAOTreasury.sol

Attack / Description	The same errors are defined redundantly in the interface and implementation file.
Code	Line 11 – 13 (IEggnogDAOTreasury.sol)

	Line 17 – 19 (EggnogDAOTreasury.sol) <pre>// Errors error NotAllowedToken(); error ZeroAddress(); error ETHTransferFailed();</pre>
Result/Recommendation	It is recommended to remove redundant code from the interface or implementation of the DAO.

6.3 SWC Attacks

ID	Title	Relationships	Test Result
SWC-131	Presence of unused variables	CWE-1164: Irrelevant Code	✓
SWC-130	Right-To-Left-Override control character (U+202E)	CWE-451: User Interface (UI) Misrepresentation of Critical Information	✓
SWC-129	Typographical Error	CWE-480: Use of Incorrect Operator	✓
SWC-128	DoS With Block Gas Limit	CWE-400: Uncontrolled Resource Consumption	✓
SWC-127	Arbitrary Jump with Function Type Variable	CWE-695: Use of Low-Level Functionality	✓

ID	Title	Relationships	Test Result
SWC-125	Incorrect Inheritance Order	CWE-696: Incorrect Behavior Order	✓
SWC-124	Write to Arbitrary Storage Location	CWE-123: Write-what-where Condition	✓
SWC-123	Requirement Violation	CWE-573: Improper Following of Specification by Caller	✓
SWC-122	Lack of Proper Signature Verification	CWE-345: Insufficient Verification of Data Authenticity	✓
SWC-121	Missing Protection against Signature Replay Attacks	CWE-347: Improper Verification of Cryptographic Signature	✓
SWC-120	Weak Sources of Randomness from Chain Attributes	CWE-330: Use of Insufficiently Random Values	✓
SWC-119	Shadowing State Variables	CWE-710: Improper Adherence to Coding Standards	✓
SWC-118	Incorrect Constructor Name	CWE-665: Improper Initialization	✓
SWC-117	Signature Malleability	CWE-347: Improper Verification of Cryptographic Signature	✓

ID	Title	Relationships	Test Result
SWC-116	Timestamp Dependence	CWE-829: Inclusion of Functionality from Untrusted Control Sphere	✓
SWC-115	Authorization through tx.origin	CWE-477: Use of Obsolete Function	✓
SWC-114	Transaction Order Dependence	CWE-362: Concurrent Execution using Shared Resource with Improper Synchronization ('Race Condition')	✓
SWC-113	DoS with Failed Call	CWE-703: Improper Check or Handling of Exceptional Conditions	✓
SWC-112	Delegatecall to Untrusted Callee	CWE-829: Inclusion of Functionality from Untrusted Control Sphere	✓
SWC-111	Use of Deprecated Solidity Functions	CWE-477: Use of Obsolete Function	✓
SWC-110	Assert Violation	CWE-670: Always-Incorrect Control Flow Implementation	✓
SWC-109	Uninitialized Storage Pointer	CWE-824: Access of Uninitialized Pointer	✓
SWC-108	State Variable Default Visibility	CWE-710: Improper Adherence to Coding Standards	✓
SWC-107	Reentrancy	CWE-841: Improper Enforcement of Behavioral Workflow	✓

ID	Title	Relationships	Test Result
SWC-106	Unprotected SELFDESTRUCT Instruction	CWE-284: Improper Access Control	✓
SWC-105	Unprotected Ether Withdrawal	CWE-284: Improper Access Control	✓
SWC-104	Unchecked Call Return Value	CWE-252: Unchecked Return Value	✓
SWC-103	Floating Pragma	CWE-664: Improper Control of a Resource Through its Lifetime	✗
SWC-102	Outdated Compiler Version	CWE-937: Using Components with Known Vulnerabilities	✓
SWC-101	Integer Overflow and Underflow	CWE-682: Incorrect Calculation	✓
SWC-100	Function Default Visibility	CWE-710: Improper Adherence to Coding Standards	✓

6.4 Verify Claims

6.4.1 The ERC-777 Token standard is correctly implemented

Status: tested and verified ✓

6.4.2 Vesting is working as expected

Status: tested and verified ✓

6.4.3 DAO Treasury is working as expected

Status: tested and verified ✓

6.4.4 Governance is working as expected

Status: tested and verified ✓

6.4.5 The smart contract is coded according to the newest standards and in a secure way.

Status: tested and verified ✓

6.5 Test Cases

Deployer Balance remaining (5% of supply, to be used in public sale):

50000000.0

- ✓ check quorum requirements in terms of number of tokens
- ✓ fetches the voting weight
- ✓ proposal creation and reading defeated proposal states
- ✓ try creating the same proposal but from an account with insufficient tokens
- ✓ try queuing a defeated proposal
- ✓ proposal creation and successful proposal states

- ✓ should return the timelock address when querying the executor
- ✓ should return true for GovernorWithParams interface ID according to EIP165
- ✓ gets voting delay
- ✓ sets voting delay through voting and Timelock
- ✓ cannot set voting delay without going through timelock
- ✓ gets voting period
- ✓ sets voting period through voting and Timelock
- ✓ cannot set voting period without going through timelock
- ✓ gets proposal threshold
- ✓ sets proposal threshold through voting and Timelock
- ✓ cannot set proposal threshold without going through timelock

cancellation tests

- ✓ proposer can cancel proposal during delay period
- ✓ anyone can cancel proposal during delay period if proposer's token balance goes below proposal threshold
- ✓ proposer can cancel during voting when no votes are made
- ✓ anyone can cancel during voting when no votes are made and if proposer's token balance goes below proposal threshold
- ✓ proposer can cancel during voting when enough votes are made for proposal to be successful
- ✓ anyone can cancel during voting when enough votes are made for proposal to be successful and proposer's token balance goes below proposal threshold
- ✓ proposer can cancel proposal once voting period has ended with Successful outcome
- ✓ anyone can cancel proposal once voting period has ended with Successful outcome and proposer's token balance falls below proposal threshold

Testing permit-related token functionality...

- ✓ check initial nonce is 0
 - ✓ check domain separator works as intended
- #### permit
- ✓ accepts an account's signature
 - ✓ rejects reused signature

- ✓ rejects others' signature
- ✓ rejects an expired permit

Testing votes-related token functionality...

- ✓ check initial nonce is 0
- ✓ check domain separator works as intended

setting delegation

self delegation with on-chain transaction

- ✓ delegate voting power when account holds tokens
- ✓ delegate voting power when account holds no tokens

delegation via signatures

- ✓ accept signed delegation to self
- ✓ should reject a used signature
- ✓ rejects bad delegatee address in message
- ✓ rejects bad nonce
- ✓ rejects expired permit

changing delegation

- ✓ check if changing delegation works as intended

delegation changes involving token transfers

- ✓ no delegation changes during transfer for non-delegated tokens
- ✓ delegation changes when delegator sends tokens to a non-delegated address
- ✓ delegation changes when a delegated recipient receives tokens from a non-delegated address
- ✓ delegation changes when both sender and recipient have delegated

checkpointing related tests

- ✓ returns the number of checkpoints for a delegate
- ✓ does not add more than 1 checkpoint in a block

getting past votes

- ✓ should revert if block number provided is ahead of current block
- ✓ should return 0 if there are no checkpoints

- ✓ returns latest amount of voting power
- ✓ returns zero for voting power if block number provided is before checkpoint
- ✓ combining previous two tests with transfers to third party for checking various checkpoints getting past total supply of tokens
- ✓ should revert if block number provided is ahead of current block
- ✓ should return 0 if there is no supply
- ✓ returns correct total supply before and after a change (burn in this case)

Eggnog Token ecosystem

Treasury Contract Functionality, without Timelock

- ✓ can receive ETH without the transaction being reverted
- ✓ can send ETH to a EOA
- ✓ reverts if trying to send ETH to Zero Address
- ✓ reverts if sending ETH to a contract without receive() function
- ✓ Reverts if trying to send ETH without being owner
- ✓ Can receive ERC777 Tokens through send() function without reverting
- ✓ Can send ERC777 Tokens to another contract that implements ERC777Receiver Hook through ERC777 Interface
- ✓ Can send ERC777 tokens to an EOA through ERC777 Interface
- ✓ can send ERC777 tokens through ERC20 interface to a contract that implements ERC777Received Hook
- ✓ Should send ERC777 tokens to an EOA through ERC20 interface
- ✓ Should fail to send ERC777 tokens through ERC777 interface to contract that does not implement ERC777 Receiver Hook
- ✓ Should succeed to send ERC777 tokens through ERC20 interface to contract that does not implement ERC777 Receiver Hook
- ✓ Should fail to receive an unauthorized ERC777
- ✓ Should fail to send ERC777 through ERC777 interface if not owner
- ✓ Should fail to send ERC777 through ERC20 interface if not owner
- ✓ Can receive and send ERC20 tokens to another contract
- ✓ Can receive and send ERC20 tokens to an EOA
- ✓ Should fail to send ERC20 if called by not owner

- ✓ Should fail to send ERC20 to the zero Address
- ✓ Should be able to take out an ERC721 sent to the contract
- ✓ Should fail to move ERC721 from contract if not owner
- ✓ Should be able to modify the Vesting Contract Address
- ✓ Should revert if try to modify the Vesting Contract Address if not Owner
- ✓ Should revert if trying to set Vesting to the Zero Address
- ✓ Should emit an even NewTokenAllowed upon calling addERC777Token
- ✓ Should revert if trying to add a new ERC777 token if not Owner
- ✓ Should revert if trying to set ZeroAddress as allowed token
- ✓ Should eliminate an existing ERC777 Token from _allowedTokens and emit event TokenRetired
- ✓ Should not be able to eliminate an existing ERC777 Token from _allowedTokens if not Owner
- ✓ Should revert if trying to eliminate the ZeroAddress from allowed tokens
- ✓ Should be able to withdraw funds from Vesting Contract in time 1
- ✓ Should be able to withdraw funds from Vesting Contract in time 0
- ✓ Should be able to withdraw funds from Vesting Contract in time 0 even if not owner

Eggnog Token Vesting Unit Tests

constructor

- ✓ initializes the token address correctly

Claimable function

- ✓ estimates claimable tokens correctly

Whitelisted Addresses can withdraw

- ✓ allows direct claiming
- ✓ allows claiming by proxy
- ✓ emits Claimed event
- ✓ rejects claims from account with zero claimable balance
- ✓ rejects ether tokens

95 passing (8s)



6.6 Test Coverage

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Lines
contracts/	100	87.5	100	96.1	
EggnogDAOTreasury.sol	100	100	100	100	
EggnogGovernor.sol	100	100	100	100	
EggnogTimelockController.sol	100	100	100	100	
EggnogToken.sol	100	100	100	100	
EggnogVesting.sol	100	70	100	88.46	
contracts/extensions/	100	100	100	100	100
ERC777Permit.sol	100	100	100	100	100
ERC777Votes.sol	100	100	100	100	100
contracts/interfaces/	100	100	100	100	100
IEggnogDAOTreasury.sol	100	100	100	100	100
IEggnogGovernor.sol	100	100	100	100	100
IEggnogTimelockController.sol	100	100	100	100	100
IEggnogToken.sol	100	100	100	100	100
IEggnogVesting.sol	100	100	100	100	100
All files	100	91.67	100	97.83	

7. Executive Summary

Two (2) independent Chainsulting experts performed an unbiased and isolated audit of the smart contract codebase.

The main goal of the audit was to verify the claims regarding the security and functions of the smart contract. During the audit, no critical, no high, no medium, no low and 3 informational issues have been found, after the manual and automated security testing.

No necessary need for action, as the recommendations only further enhance the code's readability, not security.

8. Mainnet Contract

Pending Verification



9. About the Auditor

Chainsulting is a professional software development firm, founded in 2017 and based in Germany. They show ways, opportunities, risks and offer comprehensive web3 solutions. Their services include web3 development, security and consulting.

Chainsulting conducts code audits on market-leading blockchains such as Solana, Tezos, Ethereum, Binance Smart Chain, and Polygon to mitigate risk and instil trust and transparency into the vibrant crypto community. They have also reviewed and secured the smart contracts of 1Inch, POA Network, Unicrypt, LUKSO among numerous other top DeFi projects.

Chainsulting currently secures [\\$100 billion](#) in user funds locked in multiple DeFi protocols. The team behind the leading audit firm relies on their robust technical know-how in the web3 sector to deliver top-notch smart contract audit solutions, tailored to the clients' evolving business needs.

Check our website for further information: <https://chainsulting.de>

How We Work



1 -----

PREPARATION

Supply our team with audit ready code and additional materials



2 -----

COMMUNICATION

We setup a real-time communication tool of your choice or communicate via e-mails.



3 -----

AUDIT

We conduct the audit, suggesting fixes to all vulnerabilities and help you to improve.



4 -----

FIXES

Your development team applies fixes while consulting with our auditors on their safety.



5 -----

REPORT

We check the applied fixes and deliver a full report on all steps done.