SOLIDProof

Bring trust into your projects

Blockchain Security | Smart Contract Audits | KYC Development | Marketing

MADE IN GERMANY

GNUSai

AUDIT

SECURITY ASSESSMENT

28. February, 2024

FOR

Introduction

SolidProof.io is a brand of the officially registered company MAKE Network GmbH, based in Germany. We're mainly focused on Blockchain Security such as Smart Contract Audits and KYC verification for project teams.
Solidproof.io assess potential security issues in the smart contracts implementations, review for potential inconsistencies between the code base and the whitepaper/documentation, and provide suggestions for improvement.

Disclaimer

SolidProof.io reports are not, nor should be considered, an "endorsement" or "disapproval" of any particular project or team. These reports are not, nor should be considered, an indication of the economics or value of any "product" or "asset" created by any team. SolidProof.io do not cover testing or auditing the integration with external contract or services (such as Unicrypt, Uniswap, PancakeSwap etc'...)

SolidProof.io Audits do not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technology proprietors. SolidProof Audits should not be used in any way to make decisions around investment or involvement with any particular project. These reports in no way provide investment advice, nor should be leveraged as investment advice of any sort.

SolidProof.io Reports represent an extensive auditing process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology. Blockchain technology and cryptographic assets present a high level of ongoing risk. SolidProof's position is that each company and individual are responsible for their own due diligence and continuous security. SolidProof in no way claims any guarantee of the security or functionality of the technology we agree to analyze.

Project Overview

Summary

Project Name	GNUSai
Website	https://www.gnus.ai
About the project	GNUS.AI aims at providing a system and method for distributed general-purpose computing with crypto token payment system which offers a full system that integrates a slow Blockchain Cryptotoken with a fast Directed Acyclic Graph Blockchain Cryptotoken.
Chain	Ethereum, Polygon, BNB
Language	Solidity
Codebase Link	https://etherscan.io/address/0x614577036f0a024dbc1c88ba616b394dd65d105a
Commit	N/A
Unit Tests	Provided

Social Medias

Telegram	https://t.me/geniustokens
Twitter	https://www.twitter.com/gnusai
Facebook	https://www.facebook.com/gnusai
Instagram	https://www.instagram.com/gnus.ai
Github	N/A
Reddit	N/A
Medium	N/A
Discord	https://discord.gg/syqfajh
Youtube	https://www.youtube.com/@gnusai
TikTok	N/A
LinkedIn	https://www.linkedin.com/company/gnusai

Audit Summary

Version	Delivery Date	Changelog
v1.0	28. February 2024	• Layout Project • Automated- /Manual-Security Testing • Summary

Note - The following audit report presents a comprehensive security analysis of the smart contract utilized in the project that includes outside manipulation of the contract's functions in a malicious way. This analysis did not include functional testing (or unit testing) of the contract/s logic. We cannot guarantee 100% logical correctness of the contract as we did not functionally test it. This includes internal calculations in the formulae used in the contract.

File Overview

The Team provided us with the files that should be tested in the security assessment. This audit covered the following files listed below with an SHA-1 Hash.

File Name	SHA-1 Hash
GNUSai_Contracts/@gnus.ai/contracts-upgradeable-diamond/utils/AddressUpgradeable.sol	e3d80a4b81ebd820a04da292e5c243fdf246c709
GNUSai_Contracts/@gnus.ai/contracts-upgradeable-diamond/utils/StringsUpgradeable.sol	3e115540fb70052d0575012a25b27aca73cd5499
GNUSai_Contracts/@gnus.ai/contracts-upgradeable-diamond/utils/structs/EnumerableSetUpgradeable.sol	c97114087166a5d7b17d47d6d0b159d90f3b71d3
GNUSai_Contracts/@gnus.ai/contracts-upgradeable-diamond/utils/introspection/ERC165Upgradeable.sol	d7ba2dbe845bd721c585b36d7d7f6b0a164697cf
GNUSai_Contracts/@gnus.ai/contracts-upgradeable-diamond/utils/introspection/ERC165StorageStorage.sol	54530df00a9171a1d35eda4fac23aa5932ebc0aa
GNUSai_Contracts/@gnus.ai/contracts-upgradeable-diamond/utils/introspection/IERC165Upgradeable.sol	2d0810e11ce1ff7e2e37e665b27d3e9244b9d979
GNUSai_Contracts/@gnus.ai/contracts-upgradeable-diamond/utils/introspection/ERC165StorageUpgradeable.sol	bbd2f9d55da717c41e92521ad86902276ceaa436
GNUSai_Contracts/@gnus.ai/contracts-upgradeable-diamond/utils/ContextUpgradeable.sol	f61179f34f714762de40294a314a28e9583913d9
GNUSai_Contracts/@gnus.ai/contracts-upgradeable-diamond/access/AccessControlEnumerableStorage.sol	a93557421bab8362a78236408c9e6e7118908fe1
GNUSai_Contracts/@gnus.ai/contracts-upgradeable-diamond/access/IAccessControlEnumerableUpgradeable.sol	efdbfbbe5cf91fbae48bf80d59fff0194c15c7a8
GNUSai_Contracts/@gnus.ai/contracts-upgradeable-diamond/access/AccessControlStorage.sol	f0fee06c5a63cfdf8cb7e6ace0e23fd47210d0c8
GNUSai_Contracts/@gnus.ai/contracts-upgradeable-diamond/access/AccessControlUpgradeable.sol	3c32f9a6987e59277bc9367e73260794c4fc5140
GNUSai_Contracts/@gnus.ai/contracts-upgradeable-diamond/access/AccessControlEnumerableUpgradeable.sol	843957a6483e8a57f76e7234a35de3c37a0b2265
GNUSai_Contracts/@gnus.ai/contracts-upgradeable-diamond/access/IAccessControlUpgradeable.sol	e068485d78f0f34b88425c6d6e2be65076bd7e6d
GNUSai_Contracts/@gnus.ai/contracts-upgradeable-diamond/proxy/utils/InitializableStorage.sol	44c5cd6d96fd7e322e2c176f2a1ebb4e4d28fff0
GNUSai_Contracts/@gnus.ai/contracts-upgradeable-diamond/proxy/utils/Initializable.sol	d0643fd2b4e4eda59773dcdda5ca38f35af29028
GNUSai_Contracts/@gnus.ai/contracts-upgradeable-diamond/token/ERC1155/IERC1155Upgradeable.sol	8384a0ed80449087fcf3033ce3224c7ace8cc679
GNUSai_Contracts/contracts-starter/contracts/interfaces/IERC173.sol	110f93b8acffb1eecd7f982bca938a4fddfed602
GNUSai_Contracts/contracts-starter/contracts/interfaces/IERC165.sol	fee9caeeb44643a7eabbbd6668c7099b0179eac1
GNUSai_Contracts/contracts-starter/contracts/interfaces/IDiamondLoupe.sol	f4ae29f75765a0685f9b1ae7be1240ab2e3013de
GNUSai_Contracts/contracts-starter/contracts/interfaces/IDiamondCut.sol	24bbe8ec0947a6be20a8025d5991fb0e3e00d319
GNUSai_Contracts/contracts-starter/contracts/Diamond.sol	76c36c06c70e60e8abad23df7361fd08b1e9d71b
GNUSai_Contracts/contracts-starter/contracts/libraries/LibDiamond.sol	955e24559efdb1dbef829d470a21addb5e479185
GNUSai_Contracts/contracts-starter/contracts/facets/DiamondLoupeFacet.sol	60890ec2f7c6c0a3c8073eae4ecf7a003b352422
GNUSai_Contracts/contracts-starter/contracts/facets/DiamondCutFacet.sol	bf1e45e408d826947019a224d42c55eec31b490b
GNUSai_Contracts/contracts/GeniusAccessControl.sol	d911ef7da2e77305e5320a5838dda0b9fc0ce050
GNUSai_Contracts/contracts/GeniusOwnershipFacet.sol	c6743d5e548074f5cbf7fc2082bad7d3d8be85b8
GNUSai_Contracts/contracts/GNUSConstants.sol	fe78fe514bdc9d6d326336e9d73de89f1a4b68b3
GNUSai_Contracts/contracts/GeniusDiamond.sol	0ec9bd24e32a4b0a869fbe51dcfe4caad556372b

Please note: Files with a different hash value than in this table have been modified after the security check, either intentionally or unintentionally. A different hash value may (but need not) be an indication of a changed state or potential vulnerability that was not the subject of this scan.

Imported packages

Used code from other Frameworks/Smart Contracts (direct imports).

Dependency / Import Path	Count
@gnus.ai/contracts-upgradeable-diamond/access/AccessControlEnumerableUpgradeable.sol	1
@gnus.ai/contracts-upgradeable-diamond/proxy/utils/Initializable.sol	1
@gnus.ai/contracts-upgradeable-diamond/token/ERC1155/IERC1155Upgradeable.sol	1
@gnus.ai/contracts-upgradeable-diamond/utils/introspection/ERC165StorageUpgradeable.sol	1
contracts-starter/contracts/Diamond.sol	1
contracts-starter/contracts/facets/DiamondCutFacet.sol	1
contracts-starter/contracts/facets/DiamondLoupeFacet.sol	1
contracts-starter/contracts/interfaces/IERC173.sol	1
contracts-starter/contracts/libraries/LibDiamond.sol	3

Note for Investors: We only audited contracts mentioned in the scope above. All contracts related to the project apart from that are not a part of the audit, and we cannot comment on its security and are not responsible for it in any way

Audit Information

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. The 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 aspossible.
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

Auditing Strategy and Techniques Applied

Throughout the review process, care was taken to check the repository for security-related issues, code quality, and compliance with specifications and best practices. To this end, our team of experienced pen-testers and smart contract developers reviewed the code line by line and documented any issues discovered.

We check every file manually. We use automated tools only so that they help us achieve faster and better results.

Methodology

The auditing process follows a routine series of steps:

1. Code review that includes the following:
    a. Reviewing the specifications, sources, and instructions provided to SolidProof to ensure we understand the size, scope, and functionality of the smart contract.
    b. Manual review of the code, i.e., reading the source code line by line to identify potential vulnerabilities.
    c. Comparison to the specification, i.e., verifying that the code does what is described in the specifications, sources, and instructions provided to SolidProof.

2. Testing and automated analysis that includes the following:
    a. Test coverage analysis determines whether test cases cover code and how much code is executed when those test cases are executed.
    b. Symbolic execution, which is analysing a program to determine what inputs cause each part of a program to execute.

3. Review best practices, i.e., review smart contracts to improve efficiency, effectiveness, clarity, maintainability, security, and control based on best practices, recommendations, and research from industry and academia.

4. Concrete, itemized and actionable recommendations to help you secure your smart contracts.

Overall Security

Upgradeability

Contract is an upgradeable	❌ Deployer can update the contract with new functionalities
Description	The deployer can replace the old contract with a new one with new features. Be aware of this, because the owner can add new features that may have a negative impact on your investments.
Comment	The contracts are designed on the basis of the Diamond pattern which is upgradeable by nature

Ownership

The ownership is not renounced	❌ **The owner is not renounce**
Description	The owner has not renounced the ownership that means that the owner retains control over the contract's operations, including the ability to execute functions that may impact the contract's users or stakeholders. This can lead to several potential issues, including: • Centralizations • The owner has significant control over contract's operations
Comment	The owner has the privilege to update the contract

Note - If the contract is not deployed, then we would consider the ownership not to be renounced. Moreover, if there are no ownership functionalities, ownership is automatically considered renounced.

Ownership Privileges

These functions can be dangerous. Please note that abuse can lead to financial loss. We have a guide where you can learn more about these Functions.

Minting tokens

Minting tokens refer to the process of creating new tokens in a cryptocurrency or blockchain network. This process is typically performed by the project's owner or designated authority, who has the ability to add new tokens to the network's total supply.

Contract owner cannot mint new tokens	✅ **The owner cannot mint new tokens**
Description	The owner is not able to mint new tokens once the contract is deployed.
Comment	N/A

Burning tokens

Burning tokens is the process of permanently destroying a certain number of tokens, reducing the total supply of a cryptocurrency or token. This is usually done to increase the value of the remaining tokens, as the reduced supply can create scarcity and potentially drive up demand.

Contract owner cannot burn tokens	✅ **The owner cannot burn tokens**
Description	The owner is not able burn tokens without any allowances.
Comment	N/A

Blacklist addresses

Blacklisting addresses in smart contracts is the process of adding a certain address to a blacklist, effectively preventing them from accessing or participating in certain functionalities or transactions within the contract. This can be useful in preventing fraudulent or malicious activities, such as hacking attempts or money laundering.

Contract owner cannot blacklist addresses	✅ **The owner cannot blacklist addresses**
Description	The owner is not able blacklist addresses to lock funds.
Comment	N/A

Fees and Tax

In some smart contracts, the owner or creator of the contract can set fees for certain actions or operations within the contract. These fees can be used to cover the cost of running the contract, such as paying for gas fees or compensating the contract's owner for their time and effort in developing and maintaining the contract.

Contract owner cannot set fees more than 25%	✅ **The owner cannot levy unfair taxes**
Description	The owner is not able to set the fees above 25%
Comment	N/A

Lock User Funds

In a smart contract, locking refers to the process of restricting access to certain tokens or assets for a specified period of time. When tokens or assets are locked in a smart contract, they cannot be transferred or used until the lock-up period has expired or certain conditions have been met.

Owner cannot lock the contract	✅ **The owner cannot lock the contract**
Description	The owner is not able to lock the contract by any functions or updating any variables.
Comment	N/A

External/Public functions

External/public functions are functions that can be called from outside of a contract, i.e., they can be accessed by other contracts or external accounts on the blockchain. These functions are specified using the function declaration's external or public visibility modifier.

State variables

State variables are variables that are stored on the blockchain as part of the contract's state. They are declared at the contract level and can be accessed and modified by any function within the contract. State variables can be defined with a visibility modifier, such as public, private, or internal, which determines the access level of the variable.

Components

📝 Contracts	📚 Libraries	🔍 Interfaces	🎨 Abstract
5	8	8	7

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
46	4

External	Internal	Private	Pure	View
32	116	7	9	56

StateVariables

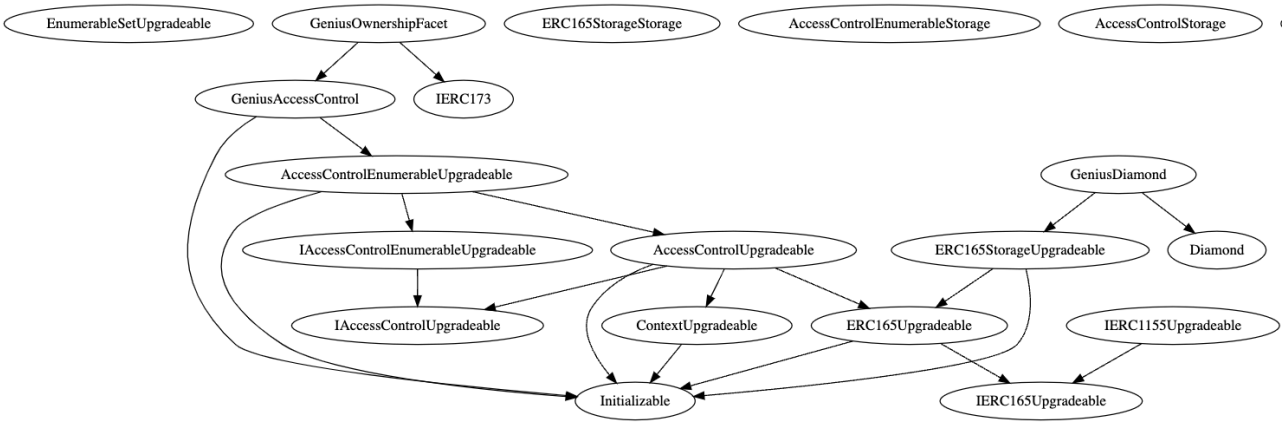
Total	🌐 Public
10	2

Capabilities

Solidity Versions observed	Delegate Call	💰 Can Receive Funds	🖥️ Uses Assembly	💣 Has Destroyable Contracts
^0.8.1 ^0.8.0 ^0.8.2	Yes	Yes		

Inheritance Graph

An inheritance graph is a graphical representation of the inheritance hierarchy among contracts. In object-oriented programming, inheritance is a mechanism that allows one class (or contract, in the case of Solidity) to inherit properties and methods from another class. It shows the relationships between different contracts and how they are related to each other through inheritance.



Centralization Privileges

Centralization can arise when one or more parties have privileged access or control over the contract's functionality, data, or decision-making. This can occur, for example, if a single entity controls the contract or if certain participants have special permissions or abilities that others do not.

In the project, there are authorities that have access to the following functions:

Files	Privileges
All	• There are no state changing ownership privileges that could harm user funds

Recommendations

To avoid potential hacking risks, the client should manage the private key of the privileged account with care. Additionally, we recommend enhancing the security practices of centralized privileges or roles in the protocol through a decentralized mechanism or smart-contract-based accounts, such as multi-signature wallets.

Here are some suggestions of what the client can do:

- Consider using multi-signature wallets: Multi-signature wallets require multiple parties to sign off on a transaction before it can be executed, providing an extra layer of security e.g. Gnosis Safe
- Use of a timelock at least with a latency of e.g. 48-72 hours for awareness of privileged operations
- Introduce a DAO/Governance/Voting module to increase transparency and user involvement
- Consider Renouncing the ownership so that the owner can no longer modify any state variables of the contract. Make sure to set up everything before renouncing.

Audit Results

Critical issues

No critical issues

High issues

No high issues

Medium issues

No medium issues

Low issues

No low issues

Informational issues

#1 | NatSpec documentation missing

File	Severity	Location	Status
All	Informational	N/A	Open

Description - If you started to comment on your code, comment on all other functions, variables, etc.

Legend for the Issue Status

Attribute or Symbol	Meaning
Open	The issue is not fixed by the project team.
Fixed	The issue is fixed by the project team.
Acknowledged(ACK)	The issue has been acknowledged or declared as part of business logic.

Blockchain Security | Smart Contract Audits | KYC Development | Marketing

MADE IN GERMANY