Blockchain Security | Smart Contract Audits | KYC
Development | Marketing

MADE IN GERMANY

AficionaDao

AUDIT

SECURITY ASSESSMENT

12. September, 2023

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	AficionaDao
Website	https://aficionadao.wtf/
About the project	We are born into a world where we have limitations placed upon us as to what constitutes value and as such, the means as to how we transact with one another. This isn't a story about the evils of fiat, nor the merits of crypto. We use Ether because it has gained the most widespread adoption.
Chain	Ethereum Network (eth)
Language	Solidity
Codebase	Token:https://etherscan.io/address/0x4ce9975a510D9b02ECCa7541bc94C5530C1B1929#code
Commit	N/A
Unit Tests	Not Provided

Social Medias

Telegram	https://t.me/aficionadao
Twitter	https://twitter.com/AficionaDAO
Facebook	N/A
Instagram	N/A
GitHub	N/A
Reddit	N/A
Medium	N/A
Discord	https://discord.gg/UbrHpjZ8d4
YouTube	N/A
TikTok	N/A
LinkedIn	N/A

Audit Summary

Version	Delivery Date	Change Log
v1.0	06. September 2023	· Layout Project · Automated/ Manual-Security Testing · Summary
v1.1	12. September 2023	· Reaudit

Note - The following audit report presents a comprehensive security analysis of the smart contract utilized in the project. 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 it was not functionally tested by us.

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
contracts/GovernorBravoInterfaces.sol	bc7a6d1a0829fee4b3cb845ff070f985c62f63aa
contracts/Timelock.sol	f5304eb756f09b1e15f519834bb9d04466984cd8
contracts/Governance.sol	af2468151a2b0e936a8345bf7a5a3d9006d2c1d0
contracts/Token.sol	879379b6b7b3c76777e61fbb555e3c516d066c47

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.

Dependency / Import Path	Count
@openzeppelin/contracts-upgradeable/proxy/utils/Initializable.sol	1
@openzeppelin/contracts-upgradeable/proxy/utils/UUPSUpgradeable.sol	1
@openzeppelin/contracts/access/Ownable.sol	1
@openzeppelin/contracts/security/Pausable.sol	1
@openzeppelin/contracts/token/ERC20/ERC20.sol	1
@openzeppelin/contracts/token/ERC20/extensions/ERC20Permit.sol	1
@openzeppelin/contracts/token/ERC20/extensions/ERC20Snapshot.sol	1
@openzeppelin/contracts/token/ERC20/extensions/ERC20Votes.sol	1
@openzeppelin/contracts/utils/math/SafeMath.sol	1
@uniswap/v2-core/contracts/interfaces/IUniswapV2Factory.sol	1
@uniswap/v2-periphery/contracts/interfaces/IUniswapV2Router02.sol	1

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.

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'sstate. They are declared at the contract level and can be accessed and modified by any function within the contract. State variables can be needed within visibility modifier, such as public, private or internal, which determines the access level of the variable.

Components

📝Contracts	📚Libraries	🔍Interfaces	🎨Abstract
7	0	3	0

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
48	5

External	Internal	Private	Pure	View
36	58	1	2	10

StateVariables

Total	🌐Public
49	49

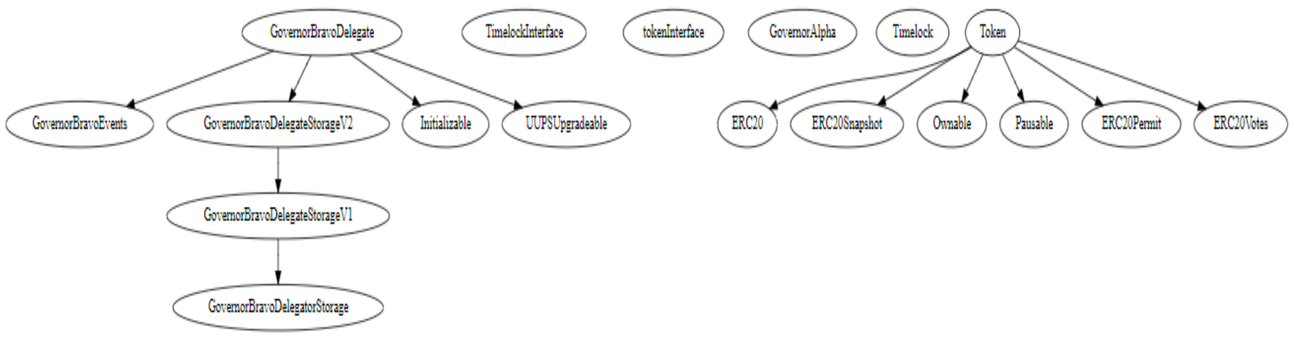
Capabilities

Solidity Versions observed	🧪 Experimental Features	💰 Can Receive Funds	🖥 Uses Assembly	💣 Has Destroyable Contracts
`0.8.19`	`ABIEncoderV2`	`Yes`	`Yes` (1 asm blocks)	`----------`

📤 Transfers ETH	⚡ Low-Level Calls	👥 Delegate Call	🧮 Uses Hash Functions	🔖 ECRecover	🌀 New/Create/Create2
			`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 methodsfrom another class. It shows the relationships between different contracts and how they are related to each other through inheritance.



Audit Information

Vulnerability & Risk Level

Risk represents the probability that a certain source threat will exploit the 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 executingthe 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 andsmart 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 codeand 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.

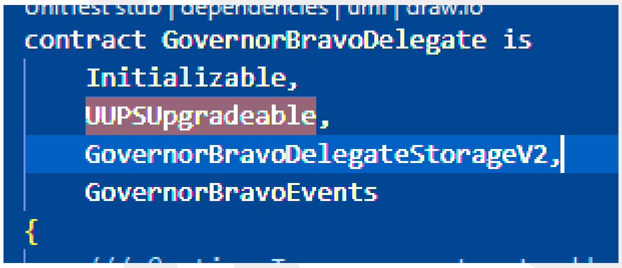
4. Concrete, itemized and actionable recommendations to help you secure your smart contracts.

Overall Security

Upgradeability

Contract is an upgradable	❌ **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 contract contains the proxy upgradable functionality through which he can upgrade the contract after initial deployment which is not recommended.

File/Line(s): L9-13
Codebase: Governance.sol



```
Unittest stub | dependencies | uml | draw.io
contract GovernorBravoDelegate is
    Initializable,
    UUPSUpgradeable,
    GovernorBravoDelegateStorageV2,
    GovernorBravoEvents
{
```

Ownership

The ownership is not renounced	**The ownership is not renounced**

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.
Example	N/A
Comment	N/A

***Note** – The contract cannot be considered as renounced till it is not deployed or having some functionality that can change the state of the contract.*

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

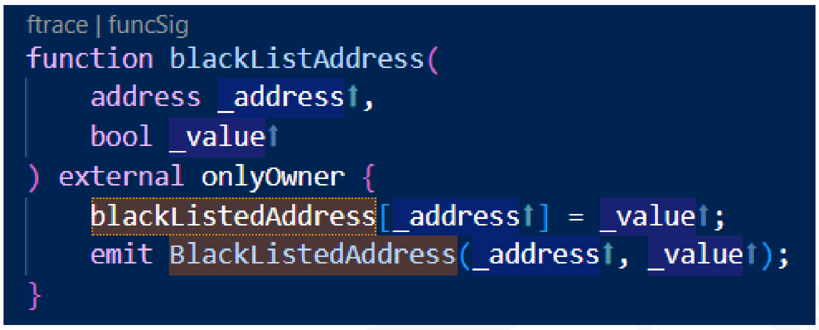
Blacklist addresses

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.

The owner can blacklist address.	**The owner can blacklist addresses.**
Description	If the owner or developers of the smart contract abuse their power to blacklist addresses without proper justification or transparency, it can lead to a decrease in trust and credibility in the project. For example, suppose an owner or developer blacklists an address without proper explanation or communication with stakeholders. In that case, it can create speculation and uncertainty among investors, potentially causing them to sell their tokens and decreasing the token's value. Furthermore, if the owner or developers have a significant number of tokens themselves and use their power to blacklist competitors or manipulate the market, it can lead to an unfair advantage and concentration of power, potentially harming the interests of other stakeholders. Therefore, it is important for projects and platforms to be transparent about their blacklisting practices and ensure that they are justified and in the best interest of their stakeholders. Investors should carefully research the project and its blacklisting practices and exercise caution before investing in any cryptocurrency or DeFi project to avoid potential losses.
Comment	The owner has the ability to blacklist addresses from transferring tokens for an unlimited period of time.

File/Line(s): L277-284

Codebase: Token.sol


```
ftrace | funcSig
function blackListAddress(
    address _address,
    bool _value
) external onlyOwner {
    blackListedAddress[_address] = _value;
    emit BlackListedAddress(_address, _value);
}
```

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 indeveloping and maintaining the contract.

Contract owner cannot set fees more than 25%	✅**Owner cannot set fees more than 25%.**
Description	The owner is not able to set the fees above 25%.
Comment	The burn tax and treasury tax cannot be set to more than 25%.

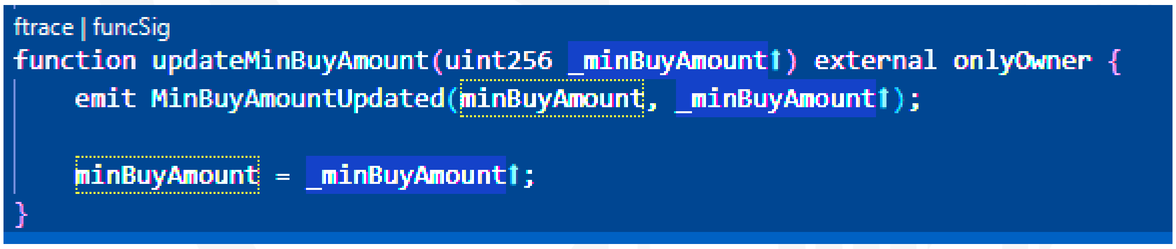
Lock User Funds

In a smart contract, locking refers to the process of restricting access to certain tokens or assets for a specified period of time. When token 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.

Contract owner can lock user funds.	❌ **The owner can lock user funds.**
Description	Locking the contract means that the owner is able to lock any funds of addresses that they are not able to transfer bought tokens anymore.
Comment	The owner can set any arbitrary value as max transaction amount which can lock the transfer of tokens if the value is set to zero. There must be a threshold where the max transfer amount should not be less than that. Also, If the minimum transfer amount is set so high then also no transfer of tokens will be possible.

File/Line(s): L312-316

Codebase: Token.sol



ftrace | funcSig

```
function updateMinBuyAmount(uint256 _minBuyAmount↑) external onlyOwner {
    emit MinBuyAmountUpdated(minBuyAmount, _minBuyAmount↑);

    minBuyAmount = _minBuyAmount↑;
}
```

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 the contract is controlled by a single entity 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:

File	Privileges
Token.sol	➢ The owner can take snapshots. ➢ The owner can update the pair path addresses. ➢ The owner can update the pair path token addresses. ➢ The owner can whitelist pair and user addresses from fees. ➢ The owner can blacklist users from transferring tokens. ➢ The owner can update the router and treasury address. ➢ The owner can update the minimum buy amount to any arbitrary value. Also, it can be set to zero. ➢ The owner can set any arbitrary value as burn tax. ➢ The owner can set any arbitrary value as slippage. ➢ The owner can set any arbitrary value as treasury tax.
Governance.sol	➢ The admin can set the voting delay time between 1 sec to 1 week. ➢ The admin can set the voting period between 1 sec to 2 weeks. ➢ The admin can set the proposal threshold between 1 to 1000000000000e18 tokens. ➢ The owner can set any address as a new pending admin. ➢ The pending admin can accept himself as an admin.
Timelock.sol	➢ The pending admin can accept himself as an admin. ➢ Only the admin can queue, execute, and cancel the transaction in the token.

Recommendations

To avoid potential hacking risks, it is advisable for the client to 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 cannot modify any state variables of the contract anymore. Make sure to set up everything before renouncing.

Audit Result

#1 | The owner can blacklist wallets from transferring tokens.

File	Severity	Location	Status
Token.sol	Medium	L277-284	ACK

Description – The owner has the ability to blacklist wallets from transferring tokens for an unlimited period of time which is not recommended.

Remediation – There must be some locking period and should not be blacklisted for an unlimited period. Add functionality for the blacklisting period.

#2 | The owner can lock tokens.

File	Severity	Location	Status
Token.sol	Medium	L225-231	ACK

Description – The owner can set any arbitrary value as a max transfer amount and minimum transfer amount that can lock the transfer functionality in the contract.

Remediation – There must be a certain threshold for the amount that can be set as the max and minimum transfer amount in the contract.

#3 | The owner can set fees more than 100%.

File	Severity	Location	Status
Token.sol	Medium	L233-237, L245-249	Fixed

Description – The burn and treasury tax can be set to more than 100% which is not recommended.

Remediation – Add functionality where the total fees in the token cannot be set to more than 25%.

Alleviation – Fees cannot be set to more than 25%.

#4 | The owner can mint unlimited number of tokens.

File	Severity	Location	Status
Token.sol	Medium	L76-78	Fixed

Description – The owner can mint the tokens after initial deployment which can manipulate the total supply of tokens and can change the price of tokens.

Remediation – It is recommended that the token should not be minted after initial deployment or if you still want to mint then there must be a total supply limit and cannot be minted more than the amount.

Alleviation – Functionality is removed from the contract.

#5 | Missing events arithmetic.

File	Severity	Location	Status
Token.sol	Low	L204-206, L208-210	Fixed
Governance.sol	Low	L63-106	Fixed
Timelock.sol	Low	L48-60	Fixed

Description – Emit all the critical parameter changes.

#6 | Missing zero and dead address check.

File	Severity	Location	Status
Token.sol	Low	L179-181, L183-189, L207-211, L213-217	Fixed

Description – Add a ‘require’ check that the address is not zero or dead.

#7 | Missing threshold.

File	Severity	Location	Status
Token.sol	Low	L219-249	ACK

Description – There is no threshold value present in the contract, through which the owner can add any arbitrary value as min buy, max buy, burn tax, and treasury tax which can lock the contract.

Remediation – It is recommended that there must be a certain threshold where the value lies and should not be so high or less that can simply lock the contract.

#8 | Remove unused code.

File	Severity	Location	Status
Token.sol	Low	L68-74	ACK

Description – This contract imports 'pausable' contract which contains the pausable functionality, but it is not used in the contract.

Remediation – It is recommended that there should not be any pausing functionality. Also, if the contracts only import the functionality and are not used will increase the gas fee. Remove unused code to avoid high gas fees.

#9 | Missing visibility.

File	Severity	Location	Status
Timelock.sol	Low	L49	Fixed

Description – Add 'public' or 'private' during initializing any variable in the contract.

#10 | NatSpec Documentation missing.

File	Severity	Location	Status
Token.sol	Informational	--	Fixed
Timelock.sol	Informational	--	Fixed
Governance.sol	Informational	--	Fixed

Description – If you started to comment on your code, also 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.
Partially Fixed	The issue is fixed but not fully remediated.

Solid Proofed

Blockchain Security | Smart Contract Audits | KYC
Development | Marketing

MADE IN GERMANY