SOLIDProof

Bring trust into your projects

Blockchain Security | Smart Contract Audits | KYC Development | Marketing

MADE IN GERMANY

GoldenAsset

AUDIT

SECURITY ASSESSMENT

26. September, 2024

FOR

Introduction

SolidProof.io is a brand of the officially registered company FutureVisions Deutschland, 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	GoldenAsset
Website	https://goldenasset.org/
About the project	TBA
Chain	TBA
Language	Solidity
Codebase	Provided as file.
Commit	N/A
Unit Tests	Not Provided

Social Medias

Telegram	TBA
Twitter	TBA
Facebook	N/A
Instagram	N/A
GitHub	N/A
Reddit	N/A
Medium	N/A
Discord	N/A
YouTube	N/A
TikTok	N/A
LinkedIn	N/A

Audit Summary

Version	Delivery Date	Change Log
v1.0	26. September 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
contracts\staking\RWAStaking.sol	28130e192453dc1759540133610a927b91388301
contracts\staking\Vesting.sol	481df374b03f7b3054caf4d33d119a0080efcc9b
contracts\staking\RewardHandler.sol	b39d7fac92c760f5f86efacc6c44191b08288f38
contracts\token\DiscountCodes.sol	b0dd750e508fb766b0a95fbf583b42d11d7729e7
contracts\token\RWATokenFactory.sol	defaf8a6c0273c266f792714e4fd0b5398c35d87
contracts\token\RWAToken.sol	d48bb993873d683b2088202dedaaf28b3d3e678b
contracts\token\Roles.sol	da1c15815059a392897761a70aca8f52c160f6c1
contracts\token\PendingManager.sol	aa35a097ba04b73e9618100843a8dfb90915b020
contracts\token\interface\IRWATokenFactory.sol	6342515050d7d8591e926a1240841bb3af339940
contracts\token\interface\IRWAToken.sol	440c970d7494b6614ce6f89452767ab86e174476
contracts\token\interface\IRoles.sol	a8df80a3708f5042030ad11e43b346216ba6873a
contracts\token\interface\IPendingManager.sol	377246e7f73a4c4ab573dfea2f03e5f15a12a159
contracts\staking\interface\IVesting.sol	1857aecb6255b0790cc8c57d136dd04f293780a9
contracts\token\interface\IDiscountCodes.sol	c188c18c036f251eb684be0d9542f78492160096d
contracts\staking\interface\IRewardHandler.sol	152e7bcaab837aa634f6bc98577eb6ad11efb9cd

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/access/AccessControl.sol	1
@openzeppelin/contracts/access/Ownable.sol	1
@openzeppelin/contracts/access/extensions/AccessControlEnumerable.sol	1
@openzeppelin/contracts/access/extensions/IAccessControlEnumerable.sol	1
@openzeppelin/contracts/token/ERC20/ERC20.sol	2
@openzeppelin/contracts/token/ERC20/IERC20.sol	6
@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol	2
@openzeppelin/contracts/utils/Pausable.sol	1
@openzeppelin/contracts/utils/ReentrancyGuard.sol	1
@openzeppelin/contracts/utils/math/Math.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
8	0	8	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
151	0

External	Internal	Private	Pure	View
134	120	7	2	69

StateVariables

Total	🌐 Public
52	45

Capabilities

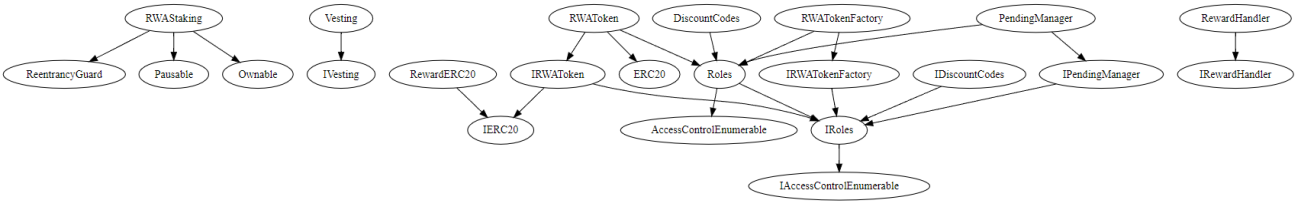
Solidity Versions observed	Experimental Features	Can Receive Funds	Uses Assembly	Has Destroyable Contracts
`^0.8.0` `^0.8.24`	`----------`			`----------`

Transfers ETH	Low-Level Calls	Delegate Call	Uses Hash Functions	ECRecover	New/Create/Create2

`Yes`			`Yes`		`yes`
					`→ NewContract:Vesting`
					`→ NewContract:RWAToken`

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 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 not upgradable	✅ **Deployer cannot update the contract with new functionalities.**
Description	The contract is not an upgradeable contract. The Deployer is not able to change or add any functionalities to the contract after deploying.
Comment	N/A

Ownership

Contract 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.
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 can mint new tokens.	❌ **The owner can mint new tokens.**
Description	The owner is able to mint new tokens once the contract is deployed.
Comment	The contract contains the functionality in which the minter of the contract can mint an unlimited amount of tokens to any wallet which is not recommended as this can manipulate the total supply of the token and price of the token will also be fluctuated because of this functionality. There must be a fixed max total supply so that the tokens total supply will not be changed.

File/Line(s): L318-320

Codebase: RWAToken.sol

```
function mint(address to, uint256 amount) external onlyMint {
    _mint(to, amount);
}
```

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 can burn tokens	❌ **The owner can burn tokens.**
Description	The owner is able burn tokens without any allowances.
Comment	The contract contains the functionality in which the minter role of the contract has the authority to burn the tokens from other wallets without any allowance, which is not recommended as this can cause the loss of funds for the user of the minter has manually burned their tokens from the contract. It is recommended that there must be a check in which the allowed tokens will only be burned by the minter.

File/Line(s): L327-329
Codebase: RWAToken.sol

```
function burn(address from, uint256 amount) external onlyMint {
    _burn(from, amount);
}
```

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 wallets.**
Description	The owner cannot blacklist wallets from transferring tokens.
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 indeveloping and maintaining the contract.

Contract owner cannot levy high taxes	✅ **The owner cannot set fees more than 25%.**
Description	The owner cannot set fees of more than 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 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 function.	❌ **The owner can lock function.**
Description	The owner can lock the claim.
Comment	The contract contains the functionality in which the owner of the contract can pause the redeem functionality mid-sale, which is not recommended as this can lock the user from claiming their tokens from the contract. There must be functionality in which the user will be able to claim their tokens from the contract without any restrictions or locking for an indefinite period of time.

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
RWAStaking	➢ The owner can update the reward token, duration, and distributor in the contract. ➢ The owner can update the distributor of the reward token in the contract. ➢ The owner can update the reward handler of the reward token. ➢ The reward distributor can deposit the reward tokens into the contract. ➢ The owner can claim the stuck tokens from the contract excluding the staked and reward tokens.
DiscountCodes	➢ The config role can set the investment discount for the RWA token. ➢ The config role can delete the present discount for the RWA token. ➢ The config role can update the redemption discount code for a specific RWA token. ➢ The config role can remove the discount code from the specific RWA token.
PendingManager	➢ The investor or configure role can cancel the pending investment from the contract. ➢ The config role can confirm the pending investment. ➢ The RWA token or config role can fill the pending redeems. ➢ The config role or redeemer can cancel the pending redemption. ➢ The config role can update any arbitrary value in the cancel pending invest threshold time. ➢ The config role can update any arbitrary value in the cancel pending redeem threshold time. ➢ The withdraw role can withdraw tokens from the

	contract.
RWAToken	➢ The minter can mint an unlimited amount of tokens to any recipient wallet. ➢ The minter can burn tokens from other wallets without any allowances. ➢ The config role or bot can update the exchange pair. ➢ The config role can manage the routing address, percentage of investment and redeem value in the contract. ➢ The config can remove the routing address. ➢ The admin can update the factory contract address. ➢ The config role can update the time interval for price updates. ➢ The config role can update the investment percentage for a given exchange pair. ➢ The config role can update the redemption percentage for a given exchange pair. ➢ The config role can deposit the exchange pair tokens to the contract. ➢ The withdrawal role can withdraw tokens and send them to the recipient's wallet.
RWATokenFactory	➢ The config role can create new tokens. ➢ The config role can pause and un-pause the factory contract. ➢ The config role can enable/disable whitelist settings in the contract. ➢ The config role or bot can change the user's whitelist status. ➢ The config role can update the discount code contract and pending manager contract address.

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

Critical Issues

No critical issues

High Issues

#1 | The owner can pause redeem.

File	Severity	Location	Status
RWAToken.sol	High	L194-309	Open

Description – The contract contains the functionality in which the owner of the contract can pause the redeem functionality mid-sale, which is not recommended as this can lock the user from claiming their tokens from the contract. There must be functionality in which the user will be able to claim their tokens from the contract without any restrictions or locking for an indefinite period of time.

Medium Issue

#1 | Missing Allowance check.

File	Severity	Location	Status
RewardHandler.sol	Medium	L108-116, L152-157	Open
Vesting.sol	Medium	L139-170	Open
PendingManager.sol	Medium	L370-418, L536-585	Open
RWAToken.sol	Medium	L125-129, L718-742	Open

Description – The contract contains the functionality in which the tokens will be transferred from the user's wallet to the contract address. It is important to have the allowance functionality added before using the transfer from function in the contract, as without any allowance, the user will not be able to send tokens to the contract address, and the functionality will fail. It is recommended that the tokens be added to the allowance before transferring tokens from other wallets.

Remediation — The approve function must be called before using transferFrom in the contract so that the user does not have to approve the contract first to transfer the tokens from their wallets.

#2 | Missing threshold.

File	Severity	Location	Status
RWAToken.sol	Medium	L544-574, L677-679	Open

Description – The config role or bot can update any arbitrary value in the invest price, redeem price, availableInvestvolume, and availableRedeemvolume, which is not recommended as this can lock the tokens if the available redeem value is set to zero for any particular exchange pair similarly the config or bot can update the redeem and invest price to any arbitrary value that can lock the invest and redeem functionality in the contract if the value is set to an excessive number. Also, The configure can set the last update since time to any arbitrary number.

Remediation – Add a check so that the invest and redeem price cannot be more than 25%, and the available invest and redeem volume cannot be zero or a very small number in the contract.

#3 | Missing 'isContract' check.

File	Severity	Location	Status
RWAToken.sol	Medium	L669-671	Open
RWATokenFactory.sol	Medium	L137-141, L147-151	Open

Description — The contract admin can update any arbitrary address as the factory address, but this is not recommended. If the address is set to any arbitrary address, the contract's functionality will fail. There must be a check so that the address can only be set to a contract address.

#4 | The withdrawer can withdraw pair tokens.

File	Severity	Location	Status
RWAToken.sol	Medium	L750-758	Open

Description — The contract contains the functionality in which the withdraw role address can withdraw tokens from the contract, including the exchange pair address token. This is not recommended as it can cause

the lock of redeem functionality if the amount of exchange pair tokens is not present in the contract.

Remediation – Add a check so that the withdrawer cannot withdraw the pair address tokens from the contract.

Low Issue

#1 | Floating pragma solidity version.

File	Severity	Location	Status
All	Low	--	Open

Description – Adding the constant version of solidity is recommended, as this prevents the unintentional deployment of a contract with an outdated compiler that contains unresolved bugs.

#2 | Missing events arithmetic.

File	Severity	Location	Status
PendingManager.sol	Low	L733-735, L740-742	Open
RWAToken.sol	Low	L669-671, L677-679	Open

Description – It is recommended to emit all the critical parameter changes.

#3 | Missing 'require' error message.

File	Severity	Location	Status
RWAStaking.sol	Low	L319	Open

Description – It is recommended to add the error message while using the require check inside a function or a modifier.

#4 | Remove Math library.

File	Severity	Location	Status
RWAStaking.sol	Low	L16	Open

Description – The compiler version above 0.8.0 has the ability to control arithmetic overflow/underflow. It is recommended to remove the unwanted code in order to avoid high gas fees.

Informational Issue

#1 | Natspec documentation missing.

File	Severity	Location	Status
All	Informational	--	Open

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.

Blockchain Security | Smart Contract Audits | KYC Development | Marketing

MADE IN GERMANY