SOLIDProof

Bring trust into your projects

Blockchain Security | Smart Contract Audits | KYC

MADE IN GERMANY

v1.0: 05. December, 2021

Audit

Security Assessment
06. December, 2021

For

NAKAMOTO .GAMES

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 security or functionality of the technology we agree to analyze.

Version	Date	Description
1.0	05. December 2021	• Layout project • Automated- /Manual-Security Testing • Summary
1.1	06. December 2021	• Reaudit

Network
Polygon

Website
https://nakamoto.games/

Telegram
https://t.me/NakamotoGames

Twitter
https://twitter.com/NakamotoGames

Medium
https://medium.com/@nakamotogames

Description

Over **65% of the world has an internet connection.** 43% of Americans can't meet monthly living expenses. Developing countries fare far worse with many relying on remittances from overseas family members. Play-to-earn gives anyone with an internet connection the opportunity to earn cryptocurrency and generate a sustainable source of income. **Nakamoto Games** is building the premier **play-to-earn ecosystem** where players can earn in endless blockchain-based games and **developers can deploy games** to a broad user base.

Project Engagement

During the 03rd of November 2021, **NAKA Team** engaged Solidproof.io to audit smart contracts that they created. The engagement was technical in nature and focused on identifying security flaws in the design and implementation of the contracts. They provided Solidproof.io with access to their code repository and whitepaper.

Logo

Contract Link

v1.1

- TBA

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 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 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.

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 SolidProof 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 SolidProof 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.

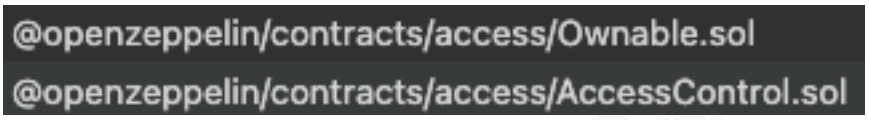
3. Best practices review, which is a review of the smart contracts to improve efficiency, effectiveness, clarify, maintainability, security, and control based on the established industry and academic practices, recommendations, and research.

4. Specific, itemized, actionable recommendations to help you take steps to secure your smart contracts.

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

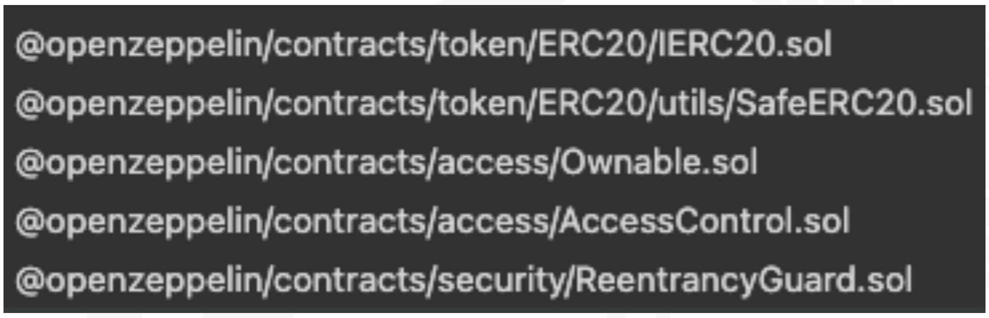
Imported packages:

ItemVault



```
@openzeppelin/contracts/access/Ownable.sol
@openzeppelin/contracts/access/AccessControl.sol
```

BalanceVault



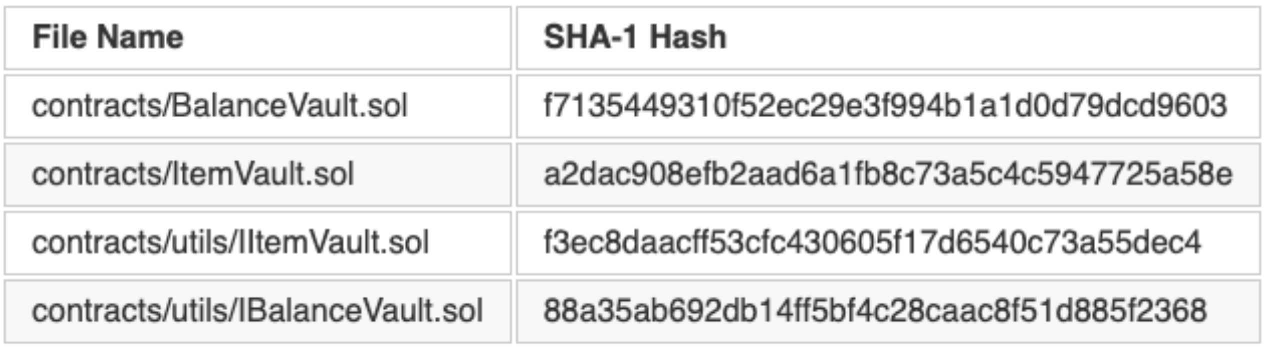
```
@openzeppelin/contracts/token/ERC20/IERC20.sol
@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol
@openzeppelin/contracts/access/Ownable.sol
@openzeppelin/contracts/access/AccessControl.sol
@openzeppelin/contracts/security/ReentrancyGuard.sol
```

Tested Contract Files

This audit covered the following files listed below with a SHA-1 Hash.

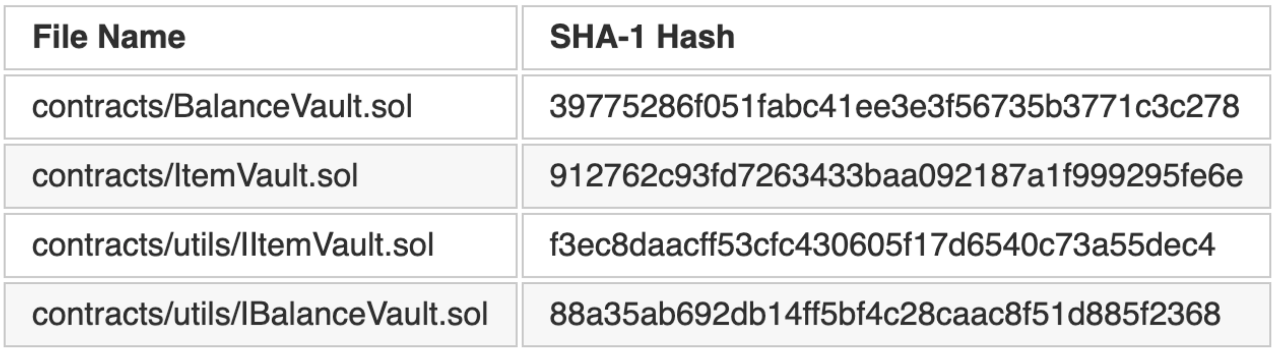
A file with a different Hash has been modified, intentionally or otherwise, after the security review. A different Hash could be (but not necessarily) an indication of a changed condition or potential vulnerability that was not within the scope of this review.

v1.0



File Name	SHA-1 Hash
contracts/BalanceVault.sol	f7135449310f52ec29e3f994b1a1d0d79dcd9603
contracts/ItemVault.sol	a2dac908efb2aad6a1fb8c73a5c4c5947725a58e
contracts/utils/IItemVault.sol	f3ec8daacff53cfc430605f17d6540c73a55dec4
contracts/utils/IBalanceVault.sol	88a35ab692db14ff5bf4c28caac8f51d885f2368

v1.1

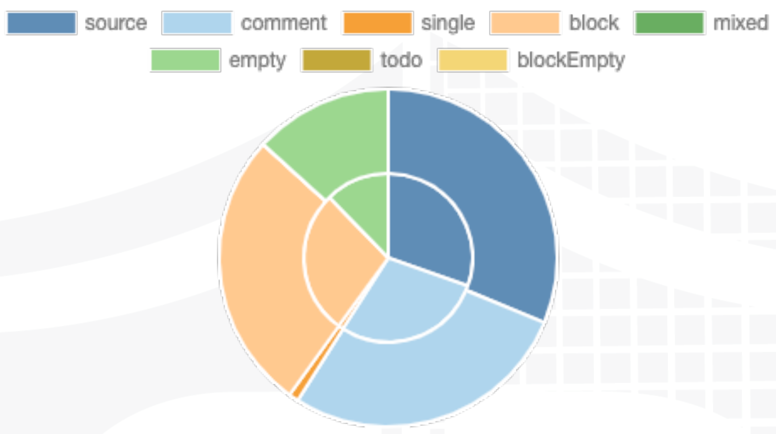


File Name	SHA-1 Hash
contracts/BalanceVault.sol	39775286f051fabc41ee3e3f56735b3771c3c278
contracts/ItemVault.sol	912762c93fd7263433baa092187a1f999295fe6e
contracts/utils/IItemVault.sol	f3ec8daacff53cfc430605f17d6540c73a55dec4
contracts/utils/IBalanceVault.sol	88a35ab692db14ff5bf4c28caac8f51d885f2368

Metrics

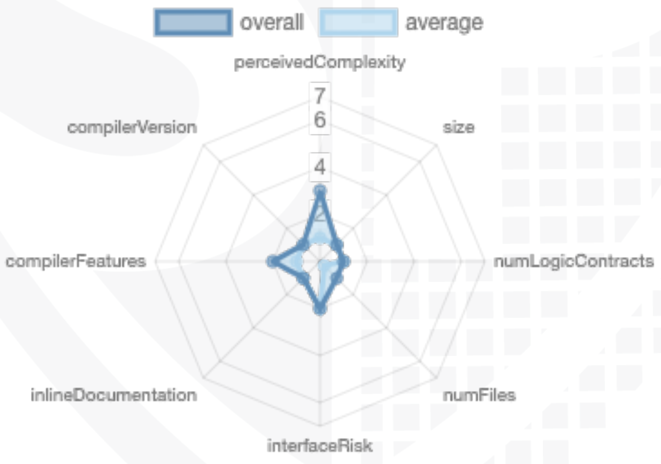
Source Lines

v1.0



Risk Level

v1.0



Capabilities

Components

Version	Contracts	Libraries	Interfaces	Abstract
1.0	2	0	2	0
1.1	2	0	2	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.

Version	Public	Payable
1.0	20	0
1.1	20	0

Version	External	Internal	Private	Pure	View
1.0	20	20	0	0	4
1.1	20	20	0	0	4

State Variables

Version	Total	Public
1.0	7	4
1.1	7	7

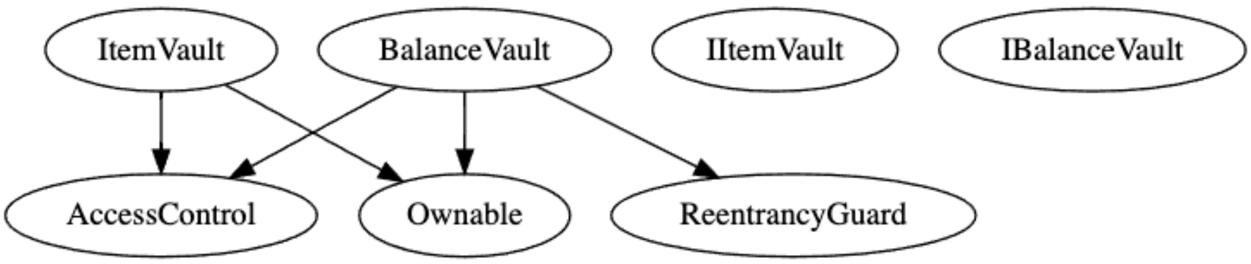
Capabilities

Version	Solidity Versions observed	Experimental Features	Can Receive Funds	Uses Assembly	Has Destroyable Contracts
1.0	^0.8.7			**** (0 asm blocks)	
1.1	`0.8.7` `^0.8.7`				

Version	Transfers ETH	Low-Level Calls	DelegateCall	Uses Hash Functions	ECRecover	New/ Create/ Create2
1.0				yes		
1.1				yes		

Inheritance Graph

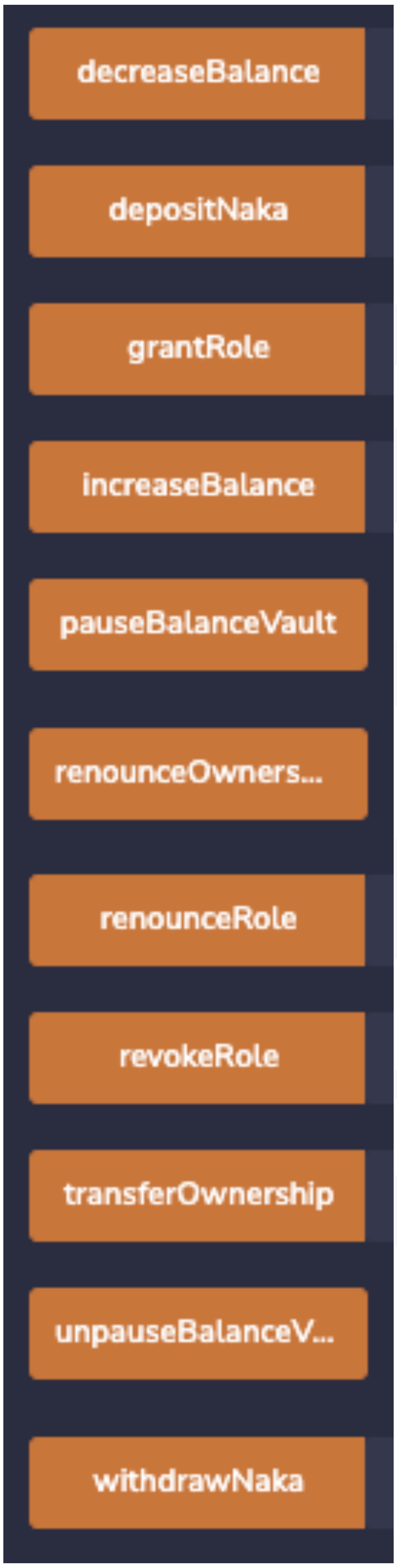
v1.0



Verify Claims

Write functions of contract

BalanceVault



decreaseBalance

depositNaka

grantRole

increaseBalance

pauseBalanceVault

renounceOwners...

renounceRole

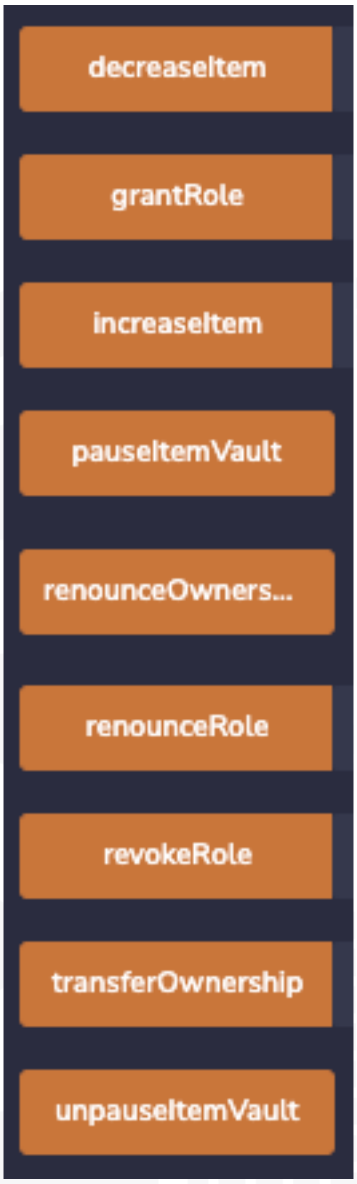
revokeRole

transferOwnership

unpauseBalanceV...

withdrawNaka

ItemVault



decreaseItem

grantRole

increaseItem

pauseItemVault

renounceOwners...

renounceRole

revokeRole

transferOwnership

unpauseItemVault

Deployer cannot mint any new tokens

Name	Exist	Tested	Verified
Deployer cannot mint	–	–	–

Deployer cannot burn or lock user funds

Name	Exist	Tested	Verified
Deployer cannot lock	–	–	–
Deployer cannot burn	–	–	–

Deployer cannot pause the contract

Name	Exist	Tested	Verified
Deployer cannot pause	✓	✓	✗

Comments:

v1.0

- Only Owner can pause ItemVault and BalanceVault

Overall checkup (Smart Contract Security)

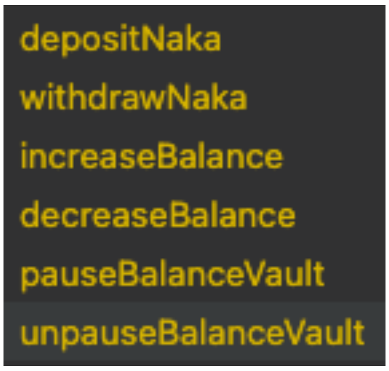
Tested	Verified
✓	✓

Legend

Attribute	Symbol
Verfified / Checked	✓
Partly Verified	⚑
Unverified / Not checked	✗
Not available	–

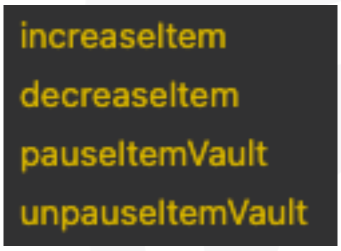
Modifiers

BalanceVault



depositNaka
withdrawNaka
increaseBalance
decreaseBalance
pauseBalanceVault
unpauseBalanceVault

ItemVault



increaseItem
decreaseItem
pauseItemVault
unpauseItemVault

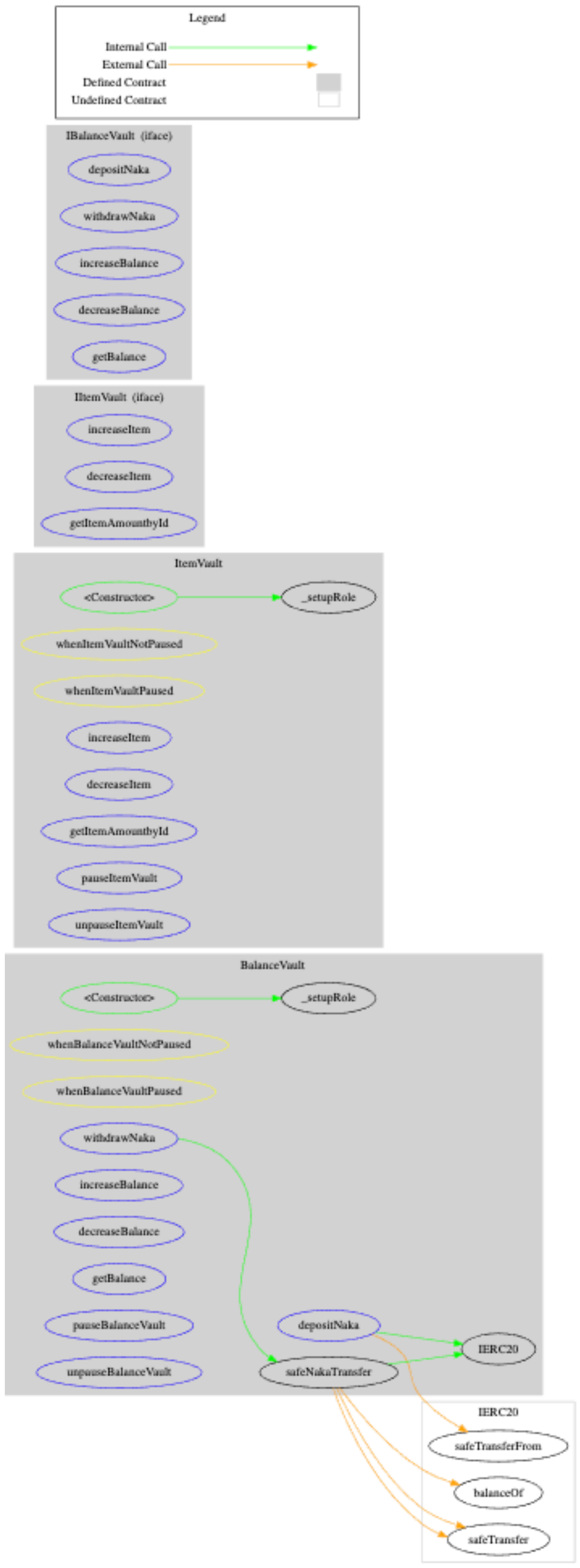
Comments

- BalanceVault
  - depositNaka
    - whenBalanceVaultNotPaused
  - withdrawNaka
    - whenBalanceVaultNotPaused
    - nonReentrant
  - increaseBalance
    - whenBalanceVaultNotPaused
    - onlyRole
      - VAULT_ADMIN
  - decreaseBalance
    - whenBalanceVaultNotPaused
    - onlyRole
      - VAULT_ADMIN
  - pauseBalanceVault
    - onlyOwner
    - whenBalanceVaultNotPaused
  - unpauseBalanceVault
    - onlyOwner
    - whenBalanceVaultPaused

- ItemVault
  - increaseItem

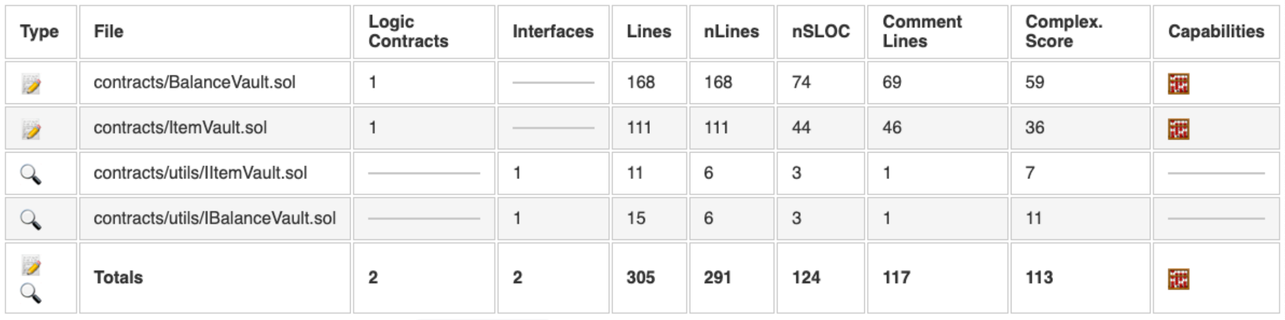
        - whenItemVaultNotPaused
        - onlyRole
            - VAULT_ADMIN
- decreaseItem
    - whenItemVaultNotPaused
    - onlyRole
        - VAULT_ADMIN
- pauseItemVault
    - whenItemVaultNotPaused
    - onlyOwner
- unpauseItemVault
    - onlyOwner
    - whenItemVaultPaused

CallGraph


Legend
Internal Call
External Call
Defined Contract
Undefined Contract
IBalanceVault (iface)
depositNaka
withdrawNaka
increaseBalance
decreaseBalance
getBalance
IItemVault (iface)
increaseItem
decreaseItem
getItemAmountbyId
ItemVault
<Constructor>
_setupRole
whenItemVaultNotPaused
whenItemVaultPaused
increaseItem
decreaseItem
getItemAmountbyId
pauseItemVault
unpauseItemVault
BalanceVault
<Constructor>
_setupRole
whenBalanceVaultNotPaused
whenBalanceVaultPaused
withdrawNaka
increaseBalance
decreaseBalance
getBalance
pauseBalanceVault
unpauseBalanceVault
depositNaka
safeNakaTransfer
IERC20
IERC20
safeTransferFrom
balanceOf
safeTransfer

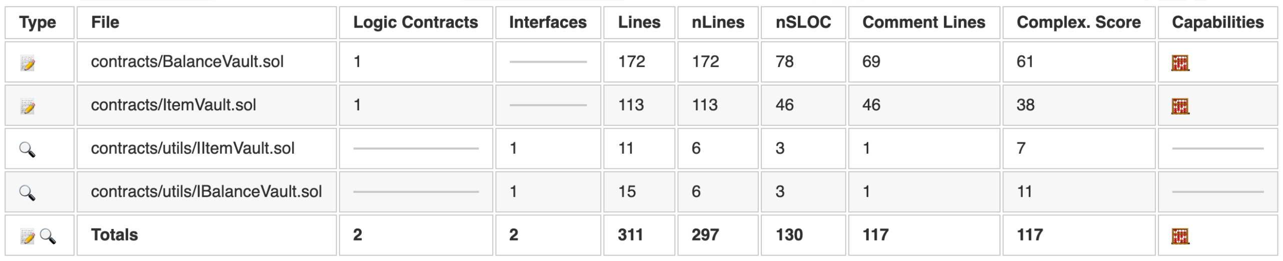
Source Units in Scope

v1.0



Type	File	Logic Contracts	Interfaces	Lines	nLines	nSLOC	Comment Lines	Complex. Score	Capabilities
📝	contracts/BalanceVault.sol	1	—	168	168	74	69	59	🧮
📝	contracts/ItemVault.sol	1	—	111	111	44	46	36	🧮
🔍	contracts/utils/IItemVault.sol	—	1	11	6	3	1	7	—
🔍	contracts/utils/IBalanceVault.sol	—	1	15	6	3	1	11	—
📝🔍	**Totals**	**2**	**2**	**305**	**291**	**124**	**117**	**113**	🧮

v1.1



Type	File	Logic Contracts	Interfaces	Lines	nLines	nSLOC	Comment Lines	Complex. Score	Capabilities
📝	contracts/BalanceVault.sol	1	—	172	172	78	69	61	🧮
📝	contracts/ItemVault.sol	1	—	113	113	46	46	38	🧮
🔍	contracts/utils/IItemVault.sol	—	1	11	6	3	1	7	—
🔍	contracts/utils/IBalanceVault.sol	—	1	15	6	3	1	11	—
📝🔍	**Totals**	**2**	**2**	**311**	**297**	**130**	**117**	**117**	🧮

Legend

Attribute	Description
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, ...)

Audit Results

AUDIT PASSED

Critical issues

- no critical issues found -

High issues

- no high issues found -

Medium issues

Issue	File	Type	Line	Description
#1	Balance Vault	Reentrancy vulnerabilities FIXED	77	Apply the [`check-effects-interactions pattern`](http://solidity.readthedocs.io/en/v0.4.21/security-considerations.html#re-entrancy).or nonReentrant modifier from OpenZeppelin

Low issues

Issue	File	Type	Line	Description
#1	Balance Vault	A floating pragma is set FIXED	3	The current pragma Solidity directive is „“^0.8.7””.
#2	ItemVa ult	A floating pragma is set FIXED	3	The current pragma Solidity directive is „“^0.8.7””.
#3	Balance Vault	Missing Zero Address Validation (missing-zero-check)	49	Check that the address is not zero
#4	Balance Vault	State variable visibility is not set FIXED	23	It is best practice to set the visibility of state variables explicitly

Informational issues

Issue	File	Type	Line	Description

#1	Balance Vault	Missing inheritance	-	Inherit from the missing interface or contract
#2	ItemVa ult	Missing inheritance	-	Inherit from the missing interface or contract
#3	Balance Vault	Wrong comment	85-90	Modify comment for withdrawNaka function
#4	ItemVa ult	Unnecessary require statement added	64, 80	There cannot be a underflow/ overflow because since pragma version 0.8.x solidity handles it under the hood with SafeMath library
#5	Balance Vault	Unnecessary require statement added	78, 94, 109, 124	There cannot be a underflow/ overflow because since pragma version 0.8.x solidity handles it under the hood with SafeMath library

Audit Comments

05. December 2021:

Read report for more information

06. December 2021:

- BalanceVault
    - > is changed to >= in line 148
    - _paused state variable was changed to _pausedBalanceVault
    - safeNakaTransfer was added to line 96
    - whenBalanceVaultNotPaused was removed from withDrawNata function
- ItemVault
    - _paused state variable was changed to _pausedItemVault

SWC Attacks

ID	Title	Relationships	Status
SWC-136	Unencrypted Private Data On-Chain	CWE-767: Access to Critical Private Variable via Public Method	PASSED
SWC-135	Code With No Effects	CWE-1164: Irrelevant Code	PASSED
SWC-134	Message call with hardcoded gas amount	CWE-655: Improper Initialization	PASSED
SWC-133	Hash Collisions With Multiple Variable Length Arguments	CWE-294: Authentication Bypass by Capture-replay	PASSED
SWC-132	Unexpected Ether balance	CWE-667: Improper Locking	PASSED
SWC-131	Presence of unused variables	CWE-1164: Irrelevant Code	PASSED
SWC-130	Right-To-Left-Override control character (U+202E)	CWE-451: User Interface (UI) Misrepresentation of Critical Information	PASSED
SWC-129	Typographical Error	CWE-480: Use of Incorrect Operator	PASSED
SWC-128	DoS With Block Gas Limit	CWE-400: Uncontrolled Resource Consumption	PASSED

SWC-127	Arbitrary Jump with Function Type Variable	CWE-695: Use of Low-Level Functionality	**PASSED**
SWC-125	Incorrect Inheritance Order	CWE-696: Incorrect Behavior Order	**PASSED**
SWC-124	Write to Arbitrary Storage Location	CWE-123: Write-what-where Condition	**PASSED**
SWC-123	Requirement Violation	CWE-573: Improper Following of Specification by Caller	**PASSED**
SWC-122	Lack of Proper Signature Verification	CWE-345: Insufficient Verification of Data Authenticity	**PASSED**
SWC-121	Missing Protection against Signature Replay Attacks	CWE-347: Improper Verification of Cryptographic Signature	**PASSED**
SWC-120	Weak Sources of Randomness from Chain Attributes	CWE-330: Use of Insufficiently Random Values	**PASSED**
SWC-119	Shadowing State Variables	CWE-710: Improper Adherence to Coding Standards	**PASSED**
SWC-118	Incorrect Constructor Name	CWE-665: Improper Initialization	**PASSED**
SWC-117	Signature Malleability	CWE-347: Improper Verification of Cryptographic Signature	**PASSED**

SWC-116	Timestamp Dependence	CWE-829: Inclusion of Functionality from Untrusted Control Sphere	**PASSED**
SWC-115	Authorization through tx.origin	CWE-477: Use of Obsolete Function	**PASSED**
SWC-114	Transaction Order Dependence	CWE-362: Concurrent Execution using Shared Resource with Improper Synchronization ('Race Condition')	**PASSED**
SWC-113	DoS with Failed Call	CWE-703: Improper Check or Handling of Exceptional Conditions	**PASSED**
SWC-112	Delegatecall to Untrusted Callee	CWE-829: Inclusion of Functionality from Untrusted Control Sphere	**PASSED**
SWC-111	Use of Deprecated Solidity Functions	CWE-477: Use of Obsolete Function	**PASSED**
SWC-110	Assert Violation	CWE-670: Always-Incorrect Control Flow Implementation	**PASSED**
SWC-109	Uninitialized Storage Pointer	CWE-824: Access of Uninitialized Pointer	**PASSED**
SWC-108	State Variable Default Visibility	CWE-710: Improper Adherence to Coding Standards	**PASSED**
SWC-107	Reentrancy	CWE-841: Improper Enforcement of Behavioral Workflow	**PASSED**
SWC-106	Unprotected SELFDESTRUCT Instruction	CWE-284: Improper Access Control	**PASSED**

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

Solid Proofed

Blockchain Security | Smart Contract Audits | KYC

MADE IN GERMANY