



SOLIDProof
Bring trust into your projects

Blockchain Security | Smart Contract Audits | KYC

MADE IN GERMANY

v1.0: 14. January, 2022

V1.1: 20. January, 2022

Audit

Security Assessment
28. February, 2022

For



UPBOTS

Disclaimer	3
Description	5
Project Engagement	5
Logo	5
Contract Link	5
Methodology	7
Used Code from other Frameworks/Smart Contracts (direct imports)	8
Tested Contract Files	9
Source Lines	10
Risk Level	10
Capabilities	11
Scope of Work	14
Inheritance Graph	14
CallGraph	17
Source Units in Scope	18
Critical issues	20
High issues	20
Medium issues	20
Low issues	20
Informational issues	20
Audit Comments	20
SWC Attacks	21

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	14. January 2022	<ul style="list-style-type: none">• Layout project• Automated- /Manual-Security Testing• Summary
1.1	20. January 2022	<ul style="list-style-type: none">• Reaudit
1.2	28. February 2022	<ul style="list-style-type: none">• Reaudit

Network

Binance Smart Chain (BEP20)

Website

<https://upbots.com/>

Telegram

<https://t.me/Upbots>

https://t.me/Upbots_announcement

Twitter

<https://twitter.com/UpBotscom>

Facebook

<https://www.facebook.com/UpBotscom>

LinkedIn

<https://www.linkedin.com/company/upbots/about/?viewAsMember=true>

Instagram

<https://www.instagram.com/upbotscom/>

YouTube

<https://www.youtube.com/channel/UCFjbtgzDJDIVSS9AaBfLKA/videos>

Discord

<https://discord.gg/wCrdMYEVjd>

Description

No matter your skill or experience, UpBots is your gateway to crypto. A trading platform where everyone wins or nobody does

Project Engagement

During the 13th of January 2022, **UpBots 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

v1.0

- <https://bscscan.com/address/0xf08508f84d66D532F146CEd0a62924aDEc68d613#code>

v1.1

- VaultFactory
 - <https://bscscan.com/address/0x4f42D6705a281302640EbCff2569c670bb4259E8#code>
- Vault
 - <https://bscscan.com/address/0xF37135e75Da1b24443D8b84793bf0D40435acCCf#code>

v1.2

- VaultFactory
 - <https://bscscan.com/address/0x893ff6B13c2f8419e4Af1809ad382eb5A8087588#code>
- Vault
 - <https://bscscan.com/address/0x9FCeb49B884A89fcB3D4AEf8ea22fb022DDd0f82#code>

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

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:

Dependency / Import Path	Count
@openzeppelin/contracts/access/Ownable.sol	1
@openzeppelin/contracts/token/ERC20/ERC20.sol	1
@openzeppelin/contracts/token/ERC20/IERC20.sol	1



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/vault_factory.sol	47fe4634124a417b2e3ffa58dbc69bb3cbfc0dcd
contracts/Vault.sol	81169fa93e0b2a9fc66cd3e2d5f67f59721edcaf
contracts/interfaces/lib/Utils.sol	98c954bc6fa9687a2bfc728343be9f4e38b13ee6
contracts/interfaces/iparaswap.sol	f292247b471c5fa387386dbd11b849ca4209f160
contracts/interfaces/uniswapv2.sol	230b2cbd39ae3bf2b49a877d1c7375e9f7331592

v1.1

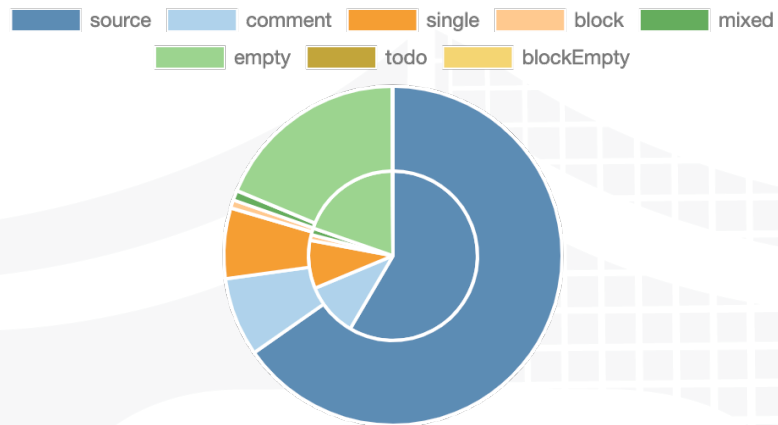
File Name	SHA-1 Hash
contracts/vault_factory.sol	646010ab380a91ca96f572a0205c4d9460eb903e
contracts/Vault.sol	35ef579428d34f33906df4ec176e6cdeb4a7f615
contracts/interfaces/lib/Utils.sol	62e37c77ca4b80817629cefed0f8a5e0da030f00
contracts/interfaces/iparaswap.sol	7fb7ff66d3a9ebc574d32e1a88ca3639205520c4
contracts/interfaces/uniswapv2.sol	e19567c355c69a61e3468f0d77927ece61a440ac

v1.2

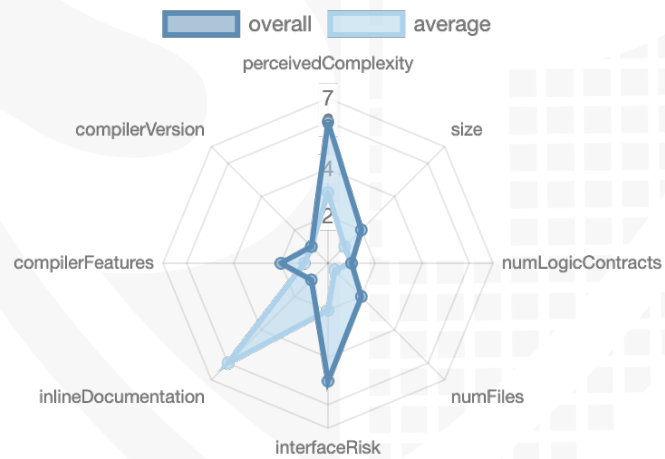
File Name	SHA-1 Hash
contracts/vault_factory.sol	339096568586bc7aa1479aa05fd5225b91eadcda
contracts/Vault.sol	780853bfb3d2fac6044213e9d3e75fe4a668b9e4
contracts/interfaces/lib/Utils.sol	62e37c77ca4b80817629cefed0f8a5e0da030f00
contracts/interfaces/iparaswap.sol	7fb7ff66d3a9ebc574d32e1a88ca3639205520c4
contracts/interfaces/uniswapv2.sol	e19567c355c69a61e3468f0d77927ece61a440ac

Metrics

Source Lines v1.0



Risk Level v1.0



Capabilities

Components

Version	Contracts	Libraries	Interfaces	Abstract
1.0	2	1	4	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	77	21
1.1	76	21

Version	External	Internal	Private	Pure	View
1.0	58	37	3	5	22
1.1	58	36	3	5	22

State Variables

Version	Total	Public
1.0	25	20
1.1	24	20
1.2	23	20

Capabilities

Version	Solidity Versions observed	Experimental Features	Can Receive Funds	Uses Assembly	Has Destroyable Contracts
1.0	0.8.10	ABIEncoderV2	yes	yes (3 asm blocks)	
1.1	0.8.10 0.8.9	ABIEncoderV2	yes	yes (3 asm blocks)	

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

1.0	yes					yes → New Contract:V ault
-----	-----	--	--	--	--	------------------------------------



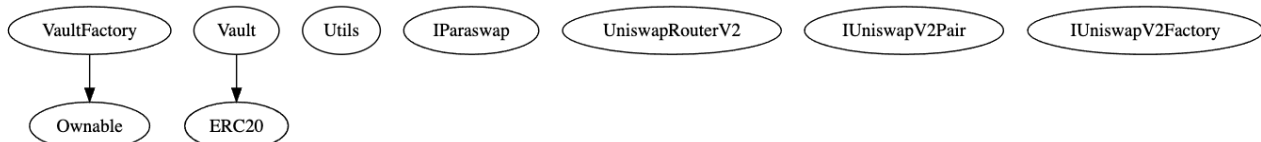
Scope of Work

The above token Team provided us with the files that needs to be tested (Github, Bscscan, Etherscan, files, etc.). The scope of the audit is the main contract (usual the same name as team appended with .sol).

We will verify the following claims:

1. Correct implementation of Token standard
2. Deployer cannot mint any new tokens
3. Deployer cannot burn or lock user funds
4. Deployer cannot pause the contract
5. Overall checkup (Smart Contract Security)

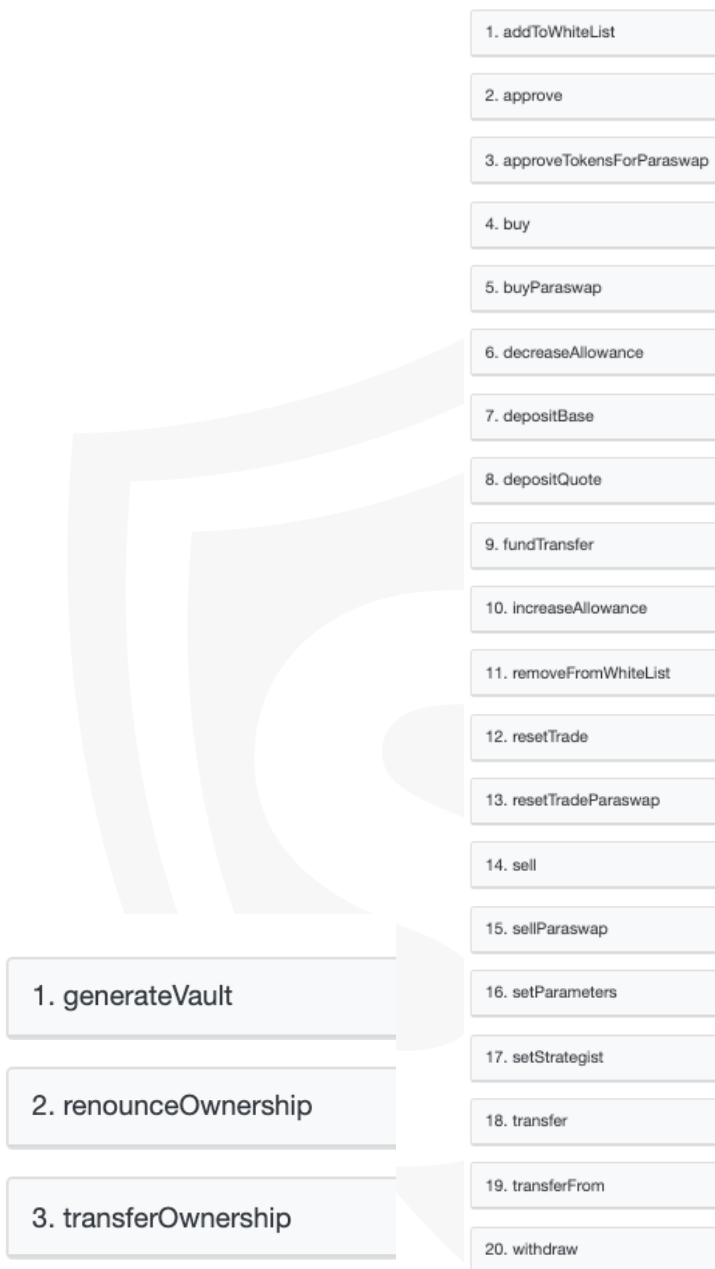
Inheritance Graph v1.0



Write functions of contract

VaultFactory

Vault



Modifier

- VaultFactory
 - onlyOwner
 - generateVault
- Vault
 - Only strategist
 - setParameters
 - fundTransfer
 - approveTokensForParaswap
 - resetTrade
 - resetTradeParaswap
 - addToWhiteList

- removeFromWhiteList
- setStrategist
- OnlyWhitelisted
 - Buy
 - Sell
 - buyParaswap
 - sellParaswap

Comments

v1.1

- Following state variables can be set without any limitations
 - percentDev
 - Max to $(2^{16}) - 1$
 - percentUpbotsFee
 - Max to $(2^{16}) - 1$
 - percentBurn
 - Max to $(2^{16}) - 1$
 - percentStakers
 - Max to $(2^{16}) - 1$
 - maxCap
 - Max to $(2^{256}) - 1$

If a function is not listed above, the function can be called without any address restrictions

CallGraph

v1.0


















v1.1


















Source Units in Scope
















v1.0

Type	File	Logic Contracts	Interfaces	Lines	nLines	nSLOC	Comment Lines	Complex. Score	Capabilities
	contracts/vault_factory.sol	1	————	59	49	34	5	44	
	contracts/Vault.sol	1	————	628	617	397	75	387	
	contracts/interfaces/lib/Utils.sol	1	————	87	87	68	14	1	————
	contracts/interfaces/iparaswap.sol	————	1	168	37	30	1	81	
	contracts/interfaces/uniswapv2.sol	————	3	213	22	17	1	92	
  	Totals	3	4	1155	812	546	96	605	  

v1.1

Type	File	Logic Contracts	Interfaces	Lines	nLines	nSLOC	Comment Lines	Complex. Score	Capabilities
	contracts/vault_factory.sol	1	————	61	51	34	6	44	
	contracts/Vault.sol	1	————	630	617	399	75	388	
	contracts/interfaces/lib/Utils.sol	1	————	87	87	68	14	1	————
	contracts/interfaces/iparaswap.sol	————	1	168	37	30	1	81	
	contracts/interfaces/uniswapv2.sol	————	3	213	22	17	1	92	
  	Totals	3	4	1159	814	548	97	606	  

v1.2

Type	File	Logic Contracts	Interfaces	Lines	nLines	nSLOC	Comment Lines	Complex. Score	Capabilities
	contracts/vault_factory.sol	1	————	61	51	34	6	44	
	contracts/Vault.sol	1	————	630	617	399	75	388	
	contracts/interfaces/lib/Utils.sol	1	————	87	87	68	14	1	————
	contracts/interfaces/iparaswap.sol	————	1	168	37	30	1	81	
	contracts/interfaces/uniswapv2.sol	————	3	213	22	17	1	92	
  	Totals	3	4	1159	814	548	97	606	  

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

- no medium issues found -

Low issues

Issue	File	Type	Line	Description
#1	VaultFactory	Require message missing	32, 33, 34, 35, 36, 37	Provide an error message to require statement
#2	Vault	Require message missing	514	Provide an error message to require statement
#3	Vault	Local variables shadowing	51	Rename the local variables that shadow another component

Informational issues

- no low informational found -

Audit Comments

14. January 2022:

- [Read whole report for more information](#)

20. January 2022:

- [Read whole report for more information](#)

28. February 2022:

- [Read whole report for more information](#)

SWC Attacks

ID	Title	Relationships	Status
SW C-1 36	Unencrypted Private Data On-Chain	CWE-767: Access to Critical Private Variable via Public Method	PASSED
SW C-1 35	Code With No Effects	CWE-1164: Irrelevant Code	PASSED
SW C-1 34	Message call with hardcoded gas amount	CWE-655: Improper Initialization	PASSED
SW C-1 33	Hash Collisions With Multiple Variable Length	CWE-294: Authentication Bypass by Capture-replay	PASSED
SW C-1 32	Unexpected Ether balance	CWE-667: Improper Locking	PASSED
SW C-1 31	Presence of unused variables	CWE-1164: Irrelevant Code	PASSED
SW C-1 30	Right-To-Left-Override control character (U+202E)	CWE-451: User Interface (UI) Misrepresentation of Critical Information	PASSED
SW C-1 29	Typographical Error	CWE-480: Use of Incorrect Operator	PASSED
SW C-1 28	DoS With Block Gas Limit	CWE-400: Uncontrolled Resource Consumption	PASSED

SW C-1 27	Arbitrary Jump with Function Type	CWE-695: Use of Low-Level Functionality	PASSED
SW C-1 25	Incorrect Inheritance Order	CWE-696: Incorrect Behavior Order	PASSED
SW C-1 24	Write to Arbitrary Storage Location	CWE-123: Write-what-where Condition	PASSED
SW C-1 23	Requirement Violation	CWE-573: Improper Following of Specification by Caller	PASSED
SW C-1 22	Lack of Proper Signature Verification	CWE-345: Insufficient Verification of Data Authenticity	PASSED
SW C-1 21	Missing Protection against Signature Replay Attacks	CWE-347: Improper Verification of Cryptographic Signature	PASSED
SW C-1 20	Weak Sources of Randomness from Chain	CWE-330: Use of Insufficiently Random Values	PASSED
SW C-11 9	Shadowing State Variables	CWE-710: Improper Adherence to Coding Standards	PASSED
SW C-11 8	Incorrect Constructor Name	CWE-665: Improper Initialization	PASSED
SW C-11 7	Signature Malleability	CWE-347: Improper Verification of Cryptographic Signature	PASSED

SW C-11 6	Timestamp Dependence	CWE-829: Inclusion of Functionality from Untrusted Control Sphere	PASSED
SW C-11 5	Authorization through tx.origin	CWE-477: Use of Obsolete Function	PASSED
SW C-11 4	Transaction Order Dependence	CWE-362: Concurrent Execution using Shared Resource with Improper Synchronization ('Race Condition')	PASSED
SW C-11 3	DoS with Failed Call	CWE-703: Improper Check or Handling of Exceptional Conditions	PASSED
SW C-11 2	Delegatecall to Untrusted Callee	CWE-829: Inclusion of Functionality from Untrusted Control Sphere	PASSED
SW C-11 1	Use of Deprecated Solidity Functions	CWE-477: Use of Obsolete Function	PASSED
SW C-11 0	Assert Violation	CWE-670: Always-Incorrect Control Flow Implementation	PASSED
SW C-1 09	Uninitialized Storage Pointer	CWE-824: Access of Uninitialized Pointer	PASSED
SW C-1 08	State Variable Default Visibility	CWE-710: Improper Adherence to Coding Standards	PASSED
SW C-1 07	Reentrancy	CWE-841: Improper Enforcement of Behavioral Workflow	PASSED
SW C-1 06	Unprotected SELFDESTRUCT Instruction	CWE-284: Improper Access Control	PASSED

SW C-1 05	Unprotected Ether Withdrawal	CWE-284: Improper Access Control	PASSED
SW C-1 04	Unchecked Call Return Value	CWE-252: Unchecked Return Value	PASSED
SW C-1 03	Floating Pragma	CWE-664: Improper Control of a Resource Through its Lifetime	PASSED
SW C-1 02	Outdated Compiler Version	CWE-937: Using Components with Known Vulnerabilities	PASSED
SW C-1 01	Integer Overflow and Underflow	CWE-682: Incorrect Calculation	PASSED
SW C-1 00	Function Default Visibility	CWE-710: Improper Adherence to Coding Standards	PASSED

The logo features the words "Solid Proofed" in a white, elegant script font. The word "Solid" is positioned above "Proofed". Behind the text is a faint, stylized shield emblem with a grid-like pattern, rendered in a darker shade of blue. The entire composition is set against a solid blue background.

Solid
Proofed

Blockchain Security | Smart Contract Audits | KYC

A small horizontal bar representing the German flag, with black, red, and gold stripes.

MADE IN GERMANY