SOLIDProof
Bring trust into your projects

Blockchain Security | Smart Contract Audits | KYC Development | Marketing

MADE IN GERMANY

BELUGA Arbitrum Core

Audit

Security Assessment
02. June, 2023

For

SolidProof_io

@solidproof_io

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	26. May 2023	• Layout project • Automated- /Manual-Security Testing • Summary
1.1	02. June 2023	• Reaudit

Network
Arbitrum One

Website
https://beluga.so

Twitter
https://twitter.com/belugadex

Telegram
https://t.me/belugadex

Discord
https://discord.com/invite/7GRPZ6tW23

Description

As the DeFi ecosystem grows, we realise that there is a need for a simple swap product with a great user experience that aligns with Beluga's vision to simplify DeFi.

Project Engagement

During the 23 of May 2023, **BELUGA Team** engaged Solidproof.io to audit smart contracts 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

Proxy -

https://arbiscan.io/address/0x7668bcbf650ae69297e411d2a8ec91e07dd91c0b#code

https://arbiscan.io/address/0x48945a091108bbbd54829b632b1df94bb50f81d7#code

https://arbiscan.io/address/0x7fbdEb84D5966c1C325D8CB2E01593D74c9A41Cd

https://arbiscan.io/address/0x15A024061c151045ba483e9243291Dee6Ee5fD8A

https://arbiscan.io/address/0x6621E58c692239874515a54Cc1D374a4101e884C

Implementations -

https://arbiscan.io/address/0xfb928d22c0f807da938ba7403a936ed31749de8d#code

https://arbiscan.io/address/0x6a02c9666b2efea6522e9249b36a168ad56d0653#code

https://arbiscan.io/address/0xea59051f3c517ec67399065f7bd79471ac323040#code

https://arbiscan.io/address/0x4316ec4d15a562e381359d6144c35c675951c120#code

v1.1

Implementation -

MasterBelugaV3 - https://arbiscan.io/address/0x2C4CAE3Cb50912AeEB6eE2812Ec6d0AACA32cDaF

VeBela - https://arbiscan.io/address/0xc2cCD02E9E1F74C0860bcf375Fea94990b91A32a

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:

Dependency / Import Path	Count
@openzeppelin/contracts-upgradeable/access/OwnableUpgradeable.sol	1
@openzeppelin/contracts-upgradeable/proxy/utils/Initializable.sol	1
@openzeppelin/contracts-upgradeable/security/PausableUpgradeable.sol	1
@openzeppelin/contracts-upgradeable/security/ReentrancyGuardUpgradeable.sol	1
@openzeppelin/contracts-upgradeable/token/ERC20/utils/SafeERC20Upgradeable.sol	1
@openzeppelin/contracts/token/ERC20/ERC20.sol	1
@openzeppelin/contracts/token/ERC20/IERC20.sol	1
@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol	3
@openzeppelin/contracts/token/ERC721/IERC721Receiver.sol	1
@openzeppelin/contracts/utils/Address.sol	1
@openzeppelin/contracts/utils/structs/EnumerableSet.sol	1

Tested Contract Files

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

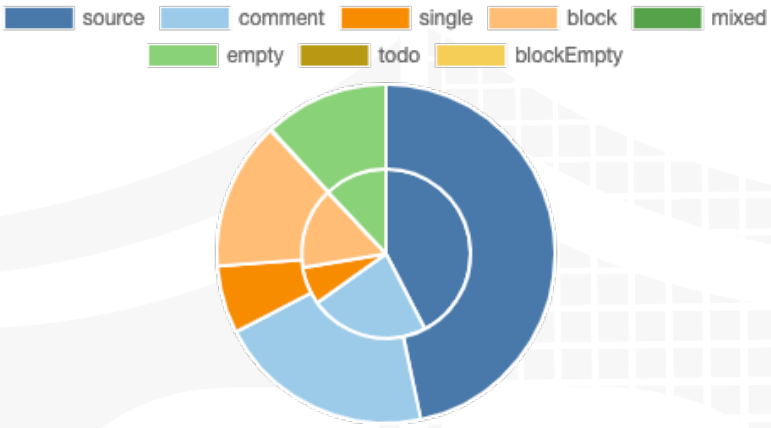
A file with a different Hash has been modified, intentionally or otherwise, after the security review. A different Hash could be (but not necessarily) an indication of a changed condition or potential vulnerability that was not within the scope of this review.

v1.0

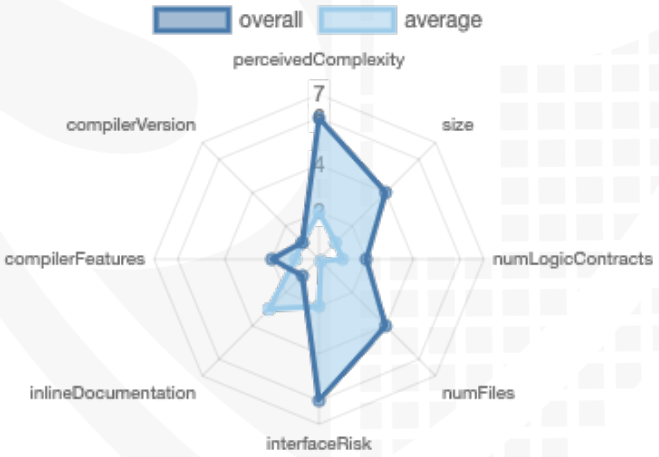
File Name	SHA-1 Hash
contracts/VeBela.sol	9c814ce30c83438a876129553ae5a0b219e58d10
contracts/MasterBelugaV3.sol	48e4f22e86c504efd7376cd347a80a006d64bfc7
contracts/PoolProxyV2.sol	3791ed0cfed2f4021e96162a43de93715a270270
contracts/Pool.sol	efd33a8052b94f07ba3e96bdca7c4fcff2b6d0c1

Metrics

Source Lines
v1.0



Risk Level
v1.0



Capabilities

Components

📝Contracts	📚Libraries	🔍Interfaces	🎨Abstract
4	0	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.

🌐Public	💰Payable
95	6

External	Internal	Private	Pure	View
79	86	17	2	40

StateVariables

Total	🌐Public
36	23

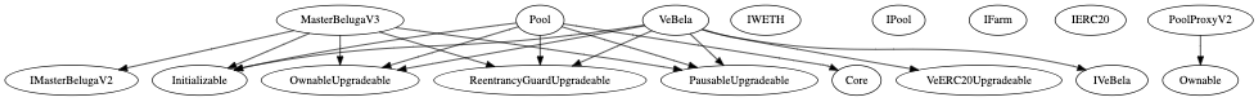
Capabilities

Solidity Versions observed	🧪 Experimental Features	💰 Can Receive Funds	🖥️ Uses Assembly	💣 Has Destroyable Contracts
`0.8.9` `^0.8.0`		`yes`	—	—

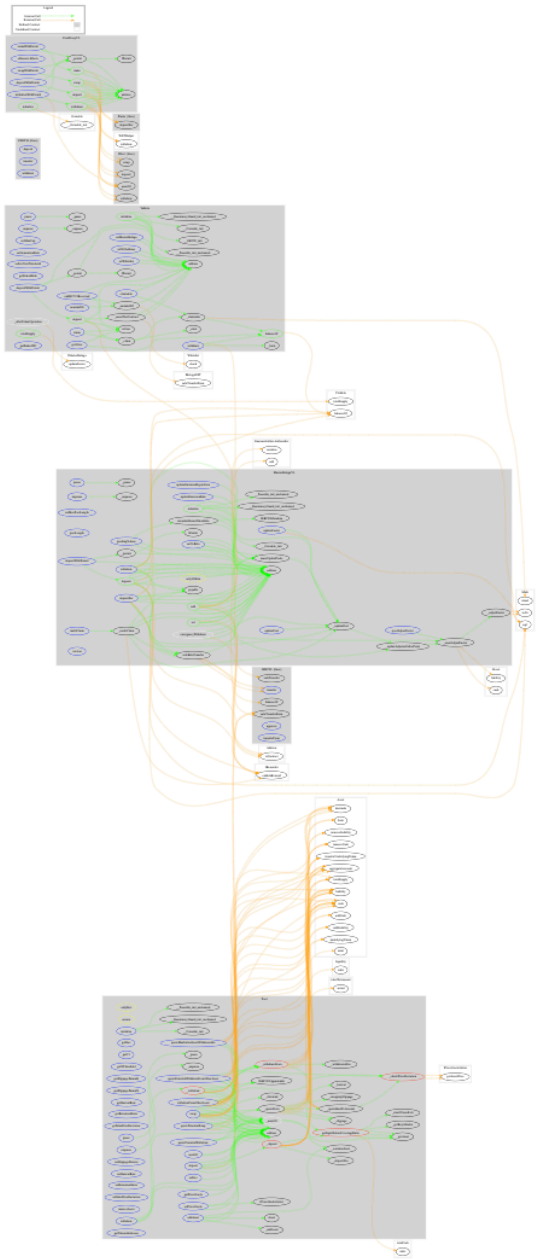
📩 Transfers ETH	⚡ Low-Level Calls	👥 DelegateCall	🧮 Uses Hash Functions	✒️ ECRecover	🌀 New/Create/Create2
`yes`	—	—	—	—	`yes` → `NewContract:WETHelper`

♻️ TryCatch	Σ Unchecked
—	—

Inheritance Graph
v1.0



CallGraph
v1.0



Scope of Work/Verify Claims

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. Are the contracts upgradeable
2. Overall checkup (Smart Contract Security)

Are the contracts upgradeable

Name	
Is contract an upgradeable?	Yes

Comments:

v1.0

- Owner can deploy a new version of the contracts which can change any limit and give owner new privileges
    - Be aware of this and do your own research for the contract which is the contract pointing to

✓	✓

Legend

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

Modifiers and public functions
v1.1

MasterBelugaV3

- pause
  - onlyOwner
- unpause
  - onlyOwner
- setMaxPoolLength
  - onlyOwner
- add
  - onlyOwner
- set
  - onlyOwner
- massUpdatePools
- updatePool
- depositFor
  - nonReentrant
  - whenNotPaused
- deposit
  - nonReentrant
  - whenNotPaused
- depositWithPermit
- multiClaim
  - nonReentrant
  - whenNotPaused
- withdraw
  - nonReentrant
  - whenNotPaused
- updateEmissionRate
  - onlyOwner
- updateEmissionRepartition
  - onlyOwner
- setVeBela
  - onlyOwner
- updateFactor
  - onlyVeBela

VeBela

- pause
  - onlyOwner
- unpause
  - onlyOwner
- setMasterBeluga
- setNftAddress
  - onlyOwner
- setWhitelist
  - onlyOwner
- setMaxCap
  - onlyOwner
- setGenerationRate
  - onlyOwner
- setInvVoteThreshold
  - onlyOwner
- deposit
  - nonReentrant
  - whenNotPaused
- depositWithPermit
- claim
  - nonReentrant
  - whenNotPaused
- withdraw
  - nonReentrant
  - whenNotPaused
- onERC721Received
  - nonReentrant
  - whenNotPaused
- unstakeNft
  - nonReentrant
  - whenNotPaused

Pool

- pause
  - onlyDev
- unpause
  - onlyDev
- setDev
  - onlyOwner
- setSlippageParams
  - onlyOwner
- setHaircutRate
  - onlyOwner
- setRetentionRatio
  - onlyOwner
- setMaxPriceDeviation
  - onlyOwner
- setPriceOracle
  - onlyOwner
- removeAsset
  - onlyOwner
- addAsset
  - onlyOwner
- deposit
  - ensure
  - nonReentrant
- withdraw
  - ensure
  - nonReentrant
  - whenNotPaused
- withdrawFromOtherAsset
  - ensure
  - nonReentrant
  - whenNotPaused
- swap
  - ensure
  - nonReentrant
  - whenNotPaused

Note: Imported contracts from official packages were not listed down below

Ownership/Authority Privileges

- *MasterBelugaV3.sol* -
  - The owner can pause/unpause the deposits
  - Set max pool length
  - Add a new Lp to the pool
  - Update the pool's allocation points
  - Update the Emission rate to any arbitrary value
  - Set VeBela address, and according to the code, this address can either be an EOA or a Contract.
  - The Veela address can set the user factor for any user address.

- *VeBela.sol* -
  - The owner can pause/unpause the deposits, claims, unstake
  - Set master Beluga, and NFT Address
  - Whitelist addresses
  - Set/Change the max cap, max votes threshold, and generation rate to any arbitrary value, as there is no constant upper limit

- *Pool.sol* -
  - The dev address can pause/unpause the deposits and withdraws
  - The owner can set Dev address
  - Set Slippage Parameters to any arbitrary value
  - Set Pool's haircut Rate and Retention Ratio
  - Set max price Deviation
  - Set price Oracle address
  - Add and Remove assets from the pool

Please check if an OnlyOwner or similar restrictive modifier has been forgotten.

Source Units in Scope
v1.0

File	Logic Contracts	Interfaces	Lines	nLines	nSLOC	Comment Lines	Complex. Score
contracts/VeBela.sol	1	————	409	398	204	125	166
contracts/MasterBelugaV3.sol	1	————	734	670	421	171	301
contracts/PoolProxyV2.sol	1	4	193	117	95	1	156
contracts/Pool.sol	1	————	952	874	424	333	348
Totals	**4**	**4**	**2288**	**2059**	**1144**	**630**	**971**

Legend

Attribute	Description
Lines	total lines of the source unit
nLines	normalised lines of the source unit (e.g. normalises functions spanning multiple lines)
nSLOC	normalised 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

Critical issues

No critical issues

High issues

No high issues

Medium issues

Medium Issues Fixed

Issue	File	Type	Line	Description	Status
#1	MasterBelugaV3.sol	Owner can disable withdrawals	—	The owner can pause withdrawals and claims which is not recommended as it may result in the lock of user funds, and they will not be able to withdraw.	**Fixed**
#2	VeBela.sol	Owner can disable withdrawals	—	The owner can pause withdrawals and claims which is not recommended as it may result in the lock of user funds, and they will not be able to withdraw.	**Fixed**

Low issues

Issue	File	Type	Line	Description	Status
#1	MasterBelugaV3	Missing Events Arithmetic	692	Emit an event for critical parameter changes	**Open**
#2	Pool.sol	Missing Events Arithmetic	368	Emit an event for critical parameter changes	**Open**
#3	Pool.sol	Divide by zero	285, 763	If the retention ratio is 10**18 then it will be divided by zero which is not allowed	**Open**
#4	Pool.sol	Restrict Dead Address	304, 442	Check that the DEAD address is restricted to be set	**Open**

#5	Pool.sol	Weak Implementation	253	The x must not be below xThreshold, and the c1 should not be smaller than x other wise, the funciton call will be reverted.	**Open**

Informational issues

Issue	File	Type	Line	Description	Status
#1	VeBela.sol	Missing Range	164	We recommend to implement a range here because if the maxCap is nearly the max then the "getVotes" function will be locked.	**Open**

Audit Comments

We recommend you to use the special form of comments (NatSpec Format, Follow link for more information https://docs.soliditylang.org/en/latest/natspec-format.html) for your contracts to provide rich documentation for functions, return variables and more. This helps investors to make clear what those variables, functions etc. do.

02. June 2023:

- There is still an owner (Owner still has not renounced ownership)
- In the pool contract, there should be a minimum value for price deviation because the function on line 394 will never pass if the max deviation is set to zero.
- Unit tests with 95% code coverage were not provided to SolidProof, so we cannot ensure complete functional correctness of the code's logic.
- We recommend **BELUGA** team conduct unit and fuzz tests thoroughly to rule out the possibilities of unwanted logical and calculation errors.
- Read the whole report and modifiers section for more information

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**

SolidProof_io

@solidproof_io

Solid Proofed

Blockchain Security | Smart Contract Audits | KYC Development | Marketing

MADE IN GERMANY