SOLIDProof

Bring trust into your projects

Blockchain Security | Smart Contract Audits | KYC Development | Marketing

MADE IN GERMANY

Dracula Finance

Audit

Security Assessment

10. May, 2023

For

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	27. April 2023 - 01. May 2023	• Layout project • Automated- /Manual-Security Testing • Summary
1.1	09. May 2023	• Reaudit

Network
ZkSync

Website
https://draculafi.xyz/

Discord
https://discord.gg/draculafi

Twitter
https://twitter.com/dracula_fi

Medium
https://medium.com/@draculafinance

Description

Introducing DraculaFi, a cutting-edge DeFi protocol built on the Zksync Era, revolutionizing the yield farming landscape. We've taken inspiration from the proven ve(3,3) model and have innovated it by introducing a game-changing mechanism that aims to provide increased rewards and stability for our users.Introducing DraculaFi, a cutting-edge DeFi protocol built on the Zksync Era, revolutionizing the yield farming landscape. We've taken inspiration from the proven ve(3,3) model and have innovated it by introducing a game-changing mechanism that aims to provide increased rewards and stability for our users.

Project Engagement

During the Date of 26 April 2023, **DraculaFi 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

- Provided as files

v1.1

- Provided as files from private repo

Note - This Audit report consists of a security analysis of the Dracula Finance smart contracts. This analysis did not include functional testing (or unit testing) of the contract's logic.

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.

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/Reentrancy.sol	c9ee869a29679ba0511ade594b95ab01450073d6
contracts/Gauges/Gauge.sol	36fd89b456bb8e9e90c69f62d936acb5096040d2
contracts/Gauges/GaugeFactory.sol	d5f950661889236ea45ae86da45c1c060813748b
contracts/Gauges/Bribe.sol	75d2660f39cf0e5cebba77d204d64663cdba9d4c
contracts/Gauges/BribeFactory.sol	5a16f1115a28043e5ef53d63774b12bae28ad8e9
contracts/Gauges/MultiRewardsPoolBase.sol	1564005565f33f73815c51c8c45162a5df3d4d7d
contracts/Dex/DraculaPair.sol	9e912593090d37c3ea631609cdfccf49c508673a
contracts/Dex/PairFees.sol	dcc37736c2da51befc8d55e5e225867624e783fd
contracts/Dex/DraculaFactory.sol	0f795ae5987e487c32945304daf9f469e91df59e
contracts/Dex/DraculaRouter01.sol	f664310df05163c8130da43c8d63c60cf2883138
contracts/Dex/SwapLibrary.sol	70cfe5203406c004791991949d58b9533d4f8c74
contracts/lib/Base64.sol	d3f3e16fd8366f8db2a44afb70c2b825666ad45f

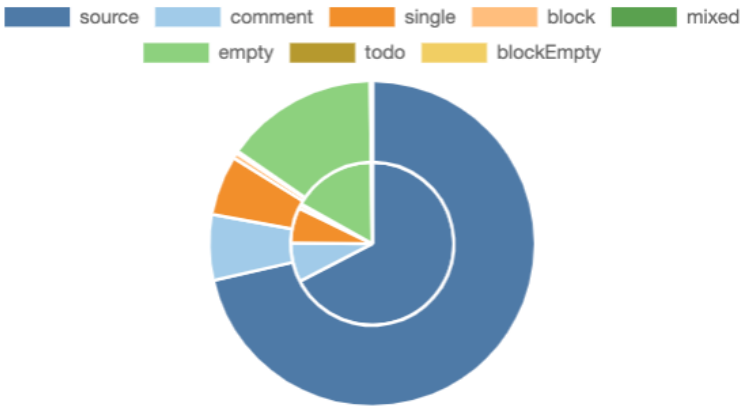
contracts/lib/Math.sol	f88846857276fd5bb830bda1c1817be3db3311eb
contracts/lib/CheckpointLib.sol	c3f1986da7ce4da679effee2eb1d72adbb5b492c
contracts/lib/Address.sol	2950f09da0cdc94a10a65a0adaa36c857e8011d6
contracts/lib/SafeERC20.sol	cfbc8307cc35ccca9bcf8ee6c5cc4291d27b0f84
contracts/Ve/Ve.sol	7bc08c46c75ed9ba343c4ddef9f37c87e227475b
contracts/Ve/DraculaMinter.sol	1bc0ec562b95347c6125e06664f879c882e18af4
contracts/Ve/BribeBond.sol	5777cd838a9fa8a478ce5484ad5ecb65258ea937
contracts/Ve/VeLogo.sol	66d5a60d6d9b39a98e7b5588f64210aa36cc7e4a
contracts/Ve/DraculaVoter.sol	c78832f4b7c08d1b40c0a50d22ce6f6b0777f333
contracts/Ve/VeDist.sol	662f2a413e53e8f718fec0a6b6e176345d674ed1
contracts/Multicall.sol	1f36fabb302131a1c9ddf46735ecd644d71e65c1
contracts/Controller.sol	a722ad8f4a5b87217876b24402e39636dce174ed
contracts/GovernanceTreasury.sol	3cef2cea04036e19c4051583061275b3fe9d094d
contracts/interface/IController.sol	350fd926ba5141150eeddb0c16adbcfadb57c7e0
contracts/interface/IPair.sol	0fc0d056fcf3e78e00560ae5b9048eb4bf84e941

contracts/interface/IERC165.sol	fc6e24730a8dc9dbf0acd13c5e56ef0e577e04cd
contracts/interface/IUnderlying.sol	8d7a6b44d7fb806d926edab59f597d65ce5146a4
contracts/interface/IPresaleFang.sol	cee01789849f1dd36703d81001c1b0c5c4c6f692
contracts/interface/IBribeBond.sol	7b62a400f8899cc578891f17096142a63bc21d89
contracts/interface/IGauge.sol	f676d26e1b58f4052147d9d6ef169ecb6aa9e05c
contracts/interface/IMinter.sol	b86498d9fc2c4c5e5d6fe94e6b83bb5583696c65
contracts/interface/IVoter.sol	069382b4149fab7f7e33ad042e604eb42e2a195e
contracts/interface/IRouter.sol	927a3b5a571d0da3b5307a7049d980b24dc0f6a7
contracts/interface/IVeLogo.sol	7e8ba5bf62ec00881c94fd0fc8ecd0e863a9dc05
contracts/interface/IVe.sol	5b3101131d1dc03114226ce01a16fb3841d0b0b4
contracts/interface/IWETH.sol	59191297a8d9963ee898fc580afa9ff561f158f1
contracts/interface/IERC721.sol	4adc321015b46400eb8ebdd1c264ead7061d96b1
contracts/interface/IGaugeFactory.sol	b5b241c23f69019474dfdb1eeb51b827019dbc55
contracts/interface/IFactory.sol	79d6811ef73997b5efef70bb77af9093eca6d21a
contracts/interface/ICallee.sol	150b43ea2b09207d316c9b15d6592bb67e90580a

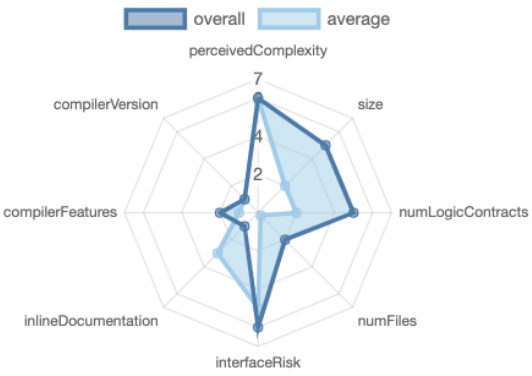
contracts/interface/IMultiRewardsPool.sol	79671468199573ca6a4c628c4449b73bafa5d548
contracts/interface/IERC721Receiver.sol	5a42c9d84843eafbc02d38a071f8fd7f27ceea83
contracts/interface/IBribeFactory.sol	d897d4c7a9728ed34eb8a3b028f9f72f76cef37d
contracts/interface/IBribe.sol	4eda6b22dda047d379e91e06d887c4a99e59c5b6
contracts/interface/IERC20.sol	745d7ed51480931749d67a175212af5ddb188972
contracts/interface/IERC721Metadata.sol	33175aa1e3830d48f5215adca8253b1e0606884d
contracts/interface/IVeDist.sol	a71fd7e7444d2383c3d8f248f165160884dd7f2f
contracts/Token/Dracula.sol	9e079d541b40383f39fb16bc6d200283e7f92527

Metrics

Source Lines
v1.0



Risk Level
v1.0



Capabilities

Components

📝Contracts	📚Libraries	🔍Interfaces	💊Abstract
19	5	24	2

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
378	9

External	Internal	Private	Pure	View
346	370	1	35	193

StateVariables

Total	🌐Public
213	150

Capabilities

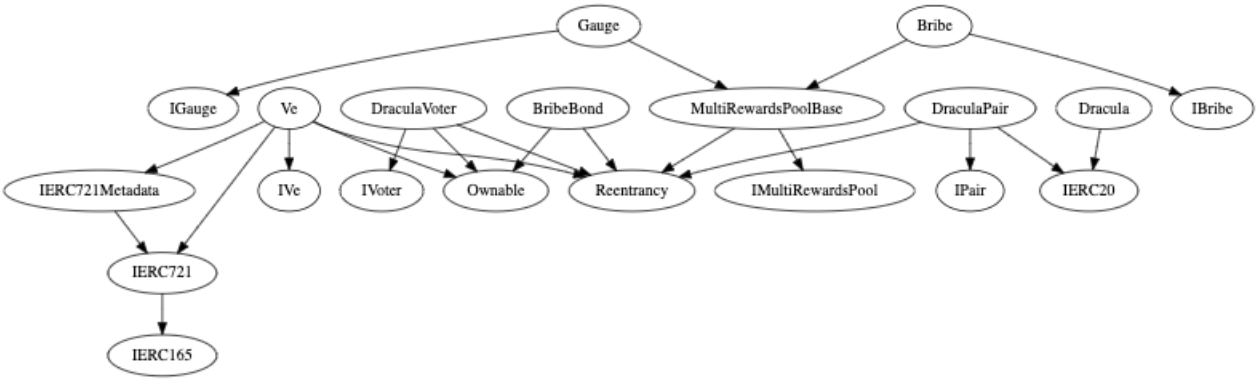
Solidity Versions observed	🧪 Experimental Features	💰 Can Receive Funds	🖥️ Uses Assembly	💣 Has Destroyable Contracts
`^0.8.15`		`yes`	`yes` (3 asm blocks)	—

🚙 Transfers ETH	⚡ Low-Level Calls	👥 DelegateCall	🧱 Uses Hash Functions	🪚 ECRecover	🌀 New/Create/Create2
`yes`	—	—	`yes`	`yes`	`yes` → `NewContract:Gauge` → `NewContract:Bribe` → `NewContract:PairFees`

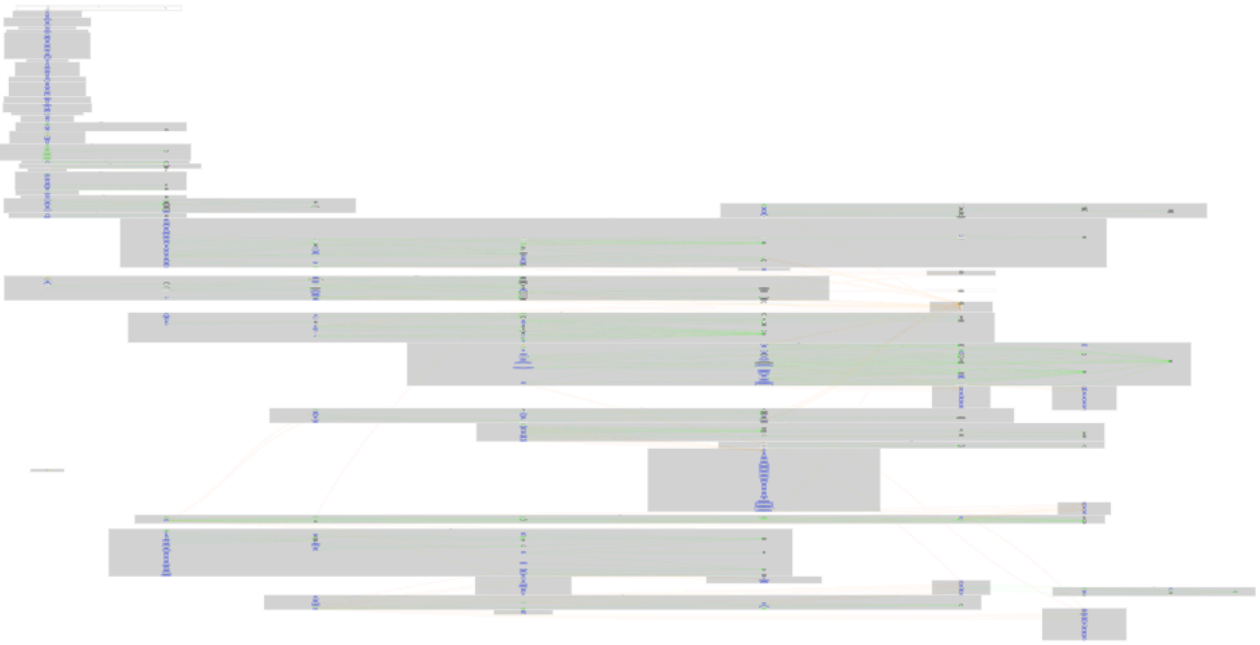
♻️ TryCatch	Σ Unchecked
`yes`	`yes`

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. Is contract an upgradeable
2. Correct implementation of Token standard
3. Overall checkup (Smart Contract Security)

Is contract an upgradeable

Name	
Is contract an upgradeable?	No

Correct implementation of Token standard

ERC20				
Function	**Description**	**Exist**	**Tested**	**Verified**
TotalSupply	Provides information about the total token supply	✓	✓	✓
BalanceOf	Provides account balance of the owner's account	✓	✓	✓
Transfer	Executes transfers of a specified number of tokens to a specified address	✓	✓	✓
TransferFrom	Executes transfers of a specified number of tokens from a specified address	✓	✓	✓
Approve	Allow a spender to withdraw a set number of tokens from a specified account	✓	✓	✓
Allowance	Returns a set number of tokens from a spender to the owner	✓	✓	✓

Overall checkup (Smart Contract Security)

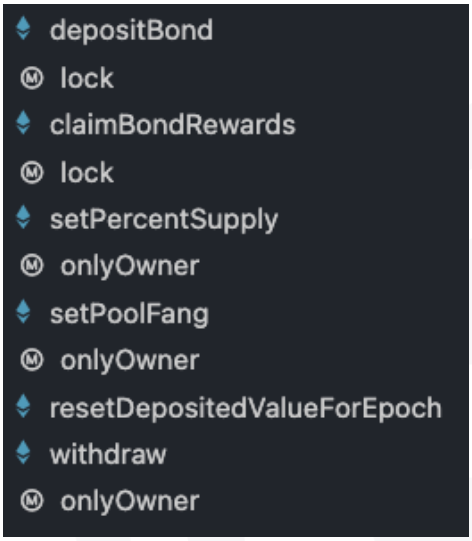
Tested	Verified
✓	✓

Legend

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

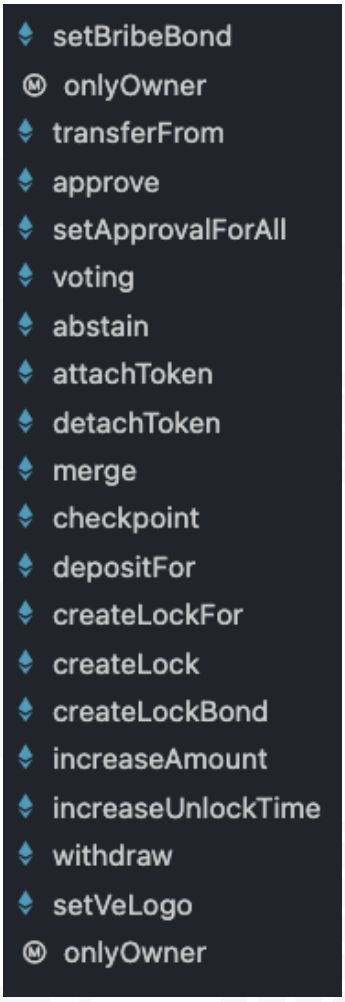
Modifiers and public functions
v1.1

BribeBond

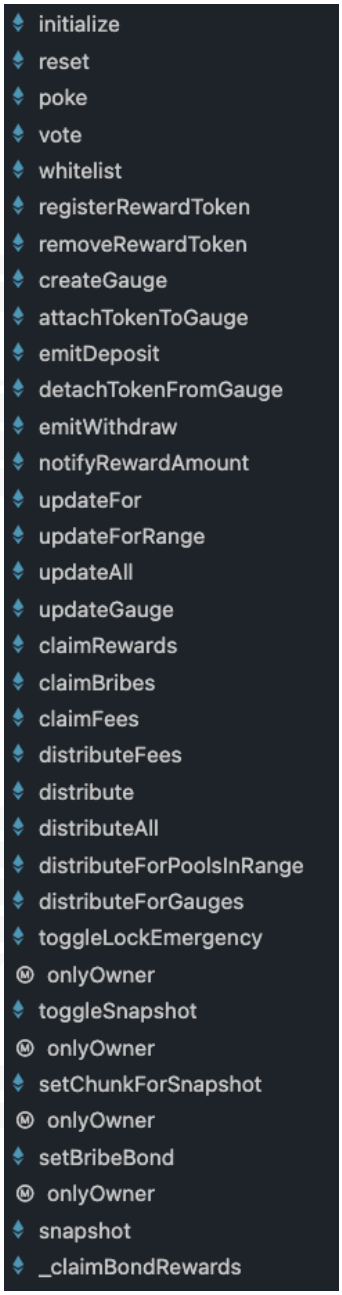


Dracula

Ve.sol



DraculaVoter



Note:

❖ General fork from SolidLizard

- Contracts inside are the same as the SolidLizard contracts on Arbitrum
  - https://solidlizard.gitbook.io/solidlizard/security/contracts
  - Differences between DraculaFi and SolidLizard contracts are the following:
    - Emergency lock and Snapshot functionality has been added to the voter cotnract.
    - Ve contract has an added snapshot functionality
    - BribeBond implementation has been added for deposits and reward claims

Ownership/Authorized Privileges

❖ *BribeBond.sol*

- Set percent supply to any arbitrary value, and it will affect the limit per day for the deposit. Thus, if the owner wants then the limit can be set to zero as well, and then the deposits will be stopped
- Set native token (FANG) pool address
- Owner can withdraw any type of tokens from the contract not including the stable ones.

❖ *Ve.sol*

- Set Bribe Bond address
- Set Ve Logo (NFT address)

❖ *DraculaVoter.sol*

- Enable/Disable the lock of voting functionalities in the contract manually
- Enable/Disable snapshots
- Set BribeBond Address

❖ *DraculaVoter.sol*

- The minter address will be the deployer of this contract.
- The minter address can mint unlimited number of Tokens
- The minter address can transfer minting rights to another address

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/Reentrancy.sol	1	—	15	15	10	2	3
contracts/Gauges/Gauge.sol	1	—	165	152	123	8	95
contracts/Gauges/GaugeFactory.sol	1	—	41	30	23	1	34
contracts/Gauges/Bribe.sol	1	—	93	85	60	10	54
contracts/Gauges/BribeFactory.sol	1	—	23	21	15	1	19
contracts/Gauges/MultiRewardsPoolBase.sol	1	—	599	539	410	52	194
contracts/Dex/DraculaPair.sol	1	—	831	734	577	73	356
contracts/Dex/PairFees.sol	1	—	40	36	23	7	13
contracts/Dex/DraculaFactory.sol	1	—	104	95	76	4	59
contracts/Dex/DraculaRouter01.sol	1	—	896	652	583	20	422
contracts/Dex/SwapLibrary.sol	1	—	394	321	287	5	125
contracts/lib/Base64.sol	1	—	73	73	51	7	167
contracts/lib/Math.sol	1	—	47	43	36	1	11
contracts/lib/CheckpointLib.sol	1	—	45	41	30	5	7
contracts/lib/Address.sol	1	—	80	72	25	42	22
contracts/lib/SafeERC20.sol	1	—	79	66	37	21	14
contracts/Ve/Ve.sol	1	—	1082	987	636	233	346
contracts/Ve/DraculaMinter.sol	1	—	214	209	146	36	119
contracts/Ve/BribeBond.sol	1	—	160	153	109	14	88
contracts/Ve/VeLogo.sol	1	—	104	98	88	5	45
contracts/Ve/DraculaVoter.sol	1	—	788	730	558	72	450
contracts/Ve/VeDist.sol	1	—	412	368	321	4	166
contracts/Multicall.sol	1	—	68	56	39	6	37
contracts/Controller.sol	1	—	46	46	34	1	23
contracts/GovernanceTreasury.sol	1	—	40	40	30	1	24
contracts/interface/IController.sol	—	1	11	6	3	1	7
contracts/interface/IPair.sol	—	1	69	13	8	2	25
contracts/interface/IERC165.sol	—	1	24	23	3	18	3
contracts/interface/IUnderlying.sol	—	1	17	6	3	1	13
contracts/interface/IPresaleFang.sol	—	1	31	22	17	1	9
contracts/interface/IBribeBond.sol	—	1	13	8	4	1	7
contracts/interface/IGauge.sol	—	1	11	6	3	1	7
contracts/interface/IMinter.sol	—	1	9	6	3	1	5
contracts/interface/IVoter.sol	—	1	40	8	4	1	23
contracts/interface/IRouter.sol	—	1	265	12	8	1	77
contracts/interface/IVeLogo.sol	—	1	13	6	3	1	3
contracts/interface/IVe.sol	—	1	68	29	20	6	31
contracts/interface/IWETH.sol	—	1	31	6	3	1	26
contracts/interface/IERC721.sol	—	1	155	74	42	110	17
contracts/interface/IGaugeFactory.sol	—	1	20	6	3	1	5

contracts/interface/IMinter.sol		1	9	6	3	1	5
contracts/interface/IVoter.sol		1	40	8	4	1	23
contracts/interface/IRouter.sol		1	265	12	8	1	77
contracts/interface/IVeLogo.sol		1	13	6	3	1	3
contracts/interface/IVe.sol		1	68	29	20	6	31
contracts/interface/IWETH.sol		1	31	6	3	1	26
contracts/interface/IERC721.sol		1	155	74	42	110	17
contracts/interface/IGaugeFactory.sol		1	20	6	3	1	5
contracts/interface/IFactory.sol		1	27	6	3	1	15
contracts/interface/ICallee.sol		1	12	6	3	1	3
contracts/interface/IMultiRewardsPool.sol		1	34	6	3	1	27
contracts/interface/IERC721Receiver.sol		1	26	20	3	15	3
contracts/interface/IBribeFactory.sol		1	9	6	3	1	3
contracts/interface/IBribe.sol		1	16	6	3	1	9
contracts/interface/IERC20.sol		1	97	31	21	61	15
contracts/interface/IERC721Metadata.sol		1	26	15	4	14	9
contracts/interface/IVeDist.sol		1	9	6	3	1	5
contracts/Token/Dracula.sol	1		101	87	70	3	41
Totals	**26**	**24**	**7573**	**6082**	**4570**	**877**	**3281**

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

Issue	File	Type	Line	Description	Status
#1	BribeBond.sol	Impossible Claim	56, 141	The stable coin address can be modified and if it is done then no user will be able to claim their deposited tokens. Although, the contract says that it is a Stable coin address but there is no actual check in existence to verify that the owner is setting the address of a stable coin. Moreover, if it is set to some other coin then the deposited coins can never be claimed.	Fixed
#2	BribeBond.sol	Owner can drain contract	56, 156	The owner can withdraw the deposited stable from the contract because the stable address in controllable by the owner and the check on L157 can be bypassed by changing the stable address.	Fixed

Low issues

Issue	File	Type	Line	Description	Status
#1	BribeBond.sol	Missing Zero Address Validation (missing-zero-check)	52, 56	Check that the address is not zero	Fixed

#2	Ve.sol	Missing Zero Address Validation (missing-zero-check)	123	Check that the address is not zero	Fixed
#3	DraculaVoter.sol	Missing Zero Address Validation (missing-zero-check)	563	Check that the address is not zero	Fixed
#4	BribeBond.sol	Missing Events Arithmetic	48-56	Emit an event for critical parameter changes. The contract has no events	Fixed

Informational issues

Issue	File	Type	Line	Description
#1	All	NatSpec documentation missing	—	If you started to comment your code, also comment all other functions, variables etc.

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 that variables, functions etc. do.

Alleviation

Comments from the Dracula Finance team over the fork declaration of the contracts -

“Yes the contracts are forked from solidlizard protocol. I've updated the pragma versions and removed some safeMath on the dex part, but didn't touch anything except that. The real modification of the fork are in the ve, voter, minter contracts.”

10. May 2023:

- We recommend DraculaFinance team to conduct unit and fuzz tests thoroughly to rule out possibilities of an unwanted logical and calculation errors.
- There is still an owner (Owner still has not renounced ownership)
- Owner can deploy a new version of the contract which can change any limit and give owner new privileges

- The low issues that exist in the SolidLizard codebase still exist in the forked code.
- We recommend using a multisig wallet for the owner address to prevent any risk of the loss of private key
- Do your own research here
- Read 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	NOT 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