



SOLIDProof
Bring trust into your projects

Blockchain Security | Smart Contract Audits | KYC

MADE IN GERMANY

GalixCity

Audit

Security Assessment

17. June, 2022

For



SolidProof_io



@solidproof_io

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
Inheritance Graph	12
CallGraph	13
Scope of Work/Verify Claims	14
Modifiers and public functions	21
Source Units in Scope	25
Critical issues	26
High issues	26
Medium issues	26
Low issues	26
Informational issues	27
Audit Comments	27
SWC Attacks	28

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	17. June 2022	<ul style="list-style-type: none">• Layout project• Automated- /Manual-Security Testing• Summary

Network

Binance Smart Chain (BEP20)

Website

<http://galixcity.io/>



Description

TBA

Project Engagement

During the 16th of June 2022, **GalixCity 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

- Github
 - <https://github.com/kogi-labs/nemo-smartcontracts>
 - Commit: b41b5c28627331cbaea5af37c95c3619361b5be7

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-upgradeable/access/AccessControlEnumerableUpgradeable.sol	3
@openzeppelin/contracts-upgradeable/access/OwnableUpgradeable.sol	4
@openzeppelin/contracts-upgradeable/proxy/utils/Initializable.sol	7
@openzeppelin/contracts-upgradeable/security/PausableUpgradeable.sol	2
@openzeppelin/contracts-upgradeable/security/ReentrancyGuardUpgradeable.sol	3
@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol	1
@openzeppelin/contracts-upgradeable/token/ERC20/extensions/ERC20PausableUpgradeable.sol	1
@openzeppelin/contracts-upgradeable/token/ERC20/extensions/ERC20SnapshotUpgradeable.sol	1
@openzeppelin/contracts-upgradeable/token/ERC20/extensions/draft-ERC20PermitUpgradeable.sol	1
@openzeppelin/contracts-upgradeable/token/ERC20/utils/SafeERC20Upgradeable.sol	3
@openzeppelin/contracts-upgradeable/token/ERC721/IERC721ReceiverUpgradeable.sol	1
@openzeppelin/contracts-upgradeable/token/ERC721/IERC721Upgradeable.sol	2
@openzeppelin/contracts-upgradeable/token/ERC721/extensions/IERC721MetadataUpgradeable.sol	1
@openzeppelin/contracts-upgradeable/utils/AddressUpgradeable.sol	3
@openzeppelin/contracts-upgradeable/utils/ContextUpgradeable.sol	3
@openzeppelin/contracts-upgradeable/utils/CountersUpgradeable.sol	1
@openzeppelin/contracts-upgradeable/utils/StringsUpgradeable.sol	1
@openzeppelin/contracts-upgradeable/utils/cryptography/draft-EIP712Upgradeable.sol	1
@openzeppelin/contracts-upgradeable/utils/introspection/ERC165Upgradeable.sol	1
@openzeppelin/contracts-upgradeable/utils/math/MathUpgradeable.sol	1
@openzeppelin/contracts-upgradeable/utils/math/SafeMathUpgradeable.sol	3
@openzeppelin/contracts-upgradeable/utils/structs/EnumerableSetUpgradeable.sol	1

Tested Contract Files

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

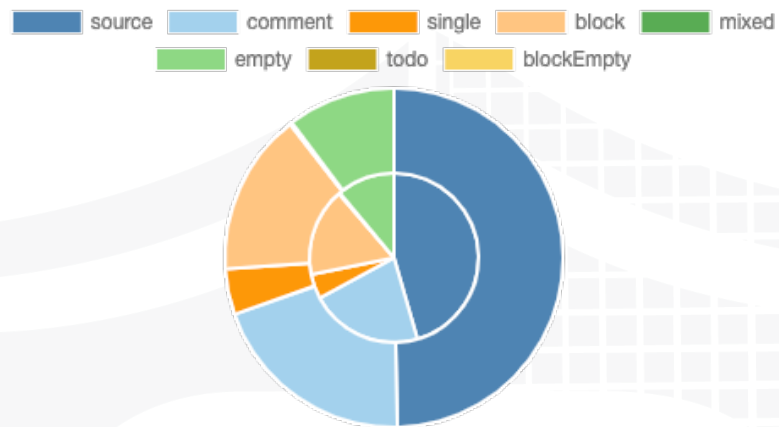
A file with a different Hash has been modified, intentionally or otherwise, after the security review. A different Hash could be (but not necessarily) an indication of a changed condition or potential vulnerability that was not within the scope of this review.

v1.0

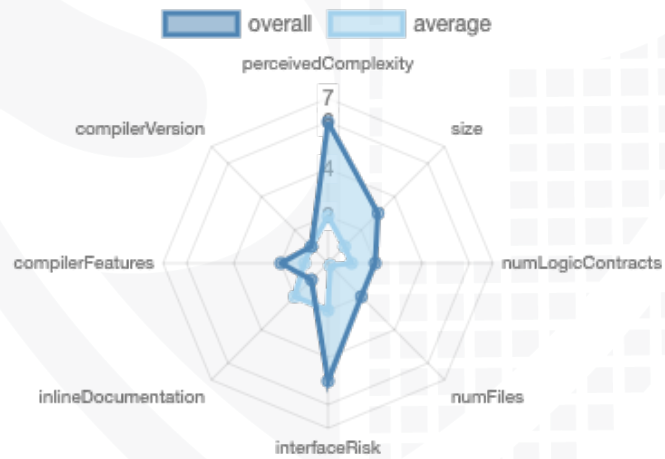
File Name	SHA-1 Hash
contracts/GalixStakingPool.sol	8b7f3f9ebb2c69e2263b8fde4722ffc24f0e26c9
contracts/common/interfaces/IGalixERC20.sol	6c1aa180c8ebcc2c510b535408727e98f1135ae9
contracts/common/interfaces/IERC20Mintable.sol	62db31e4c2abefabaf05c7d1629ec42fa2006738
contracts/common/interfaces/IWETH.sol	989b3d4c3dd0cd4386a69264eb58bf307b5b4f4b
contracts/common/interfaces/IGalixERC721.sol	32da5c31bf05431f839399e06698edddad1205ea
contracts/common/SafeERC20Upgradeable.sol	cda2bf20fe64124f64482d3c54d1eae4e35d182c
contracts/common/ERC721Upgradeable.sol	1e9467a466cfb6ab3ca8df7e5b3e20e29dd2a997
contracts/common/ERC721URIStorageUpgradeable.sol	640c7713f5e122fcbc12ab01e5288f0e37d4adbe
contracts/common/ERC721PausableUpgradeable.sol	88955a7d3f839e4a97f9482820f4eb1b6a392c25
contracts/common/meta-transactions/ContextMixin.sol	1825b502a3289905d271b0e740451b002ab22d89
contracts/common/meta-transactions/NativeMetaTransaction.sol	a328fafbadd4d9a2d8e91caa2b48c93145aa0bd7
contracts/common/meta-transactions/EIP712Base.sol	6a5856071ff1452e009768dfcd6693f1c756450
contracts/common/ERC721BurnableUpgradeable.sol	8e5b391ae3081942da81eef3a417adbb45cf45ff
contracts/GalixGovernance.sol	64d3fd3406863699ba5c9495650eca414ecb6964
contracts/GalixERC721.sol	a234b00ab8b4226ecf477941455e3ebea047e0c8
contracts/GalixERC20.sol	e93a9b413b8470d8b7a1df9e6108bbcff810677d

Metrics

Source Lines v1.0



Risk Level v1.0



Capabilities

Components

Version	Contracts	Libraries	Interfaces	Abstract
1.0	9	1	4	4

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	119	2

Version	External	Internal	Private	Pure	View
1.0	71	163	2	1	55

State Variables

Version	Total	Public
1.0	50	31

Capabilities

Version	Solidity Versions observed	Experimental Features	Can Receive Funds	Uses Assembly	Has Destroyable Contracts
1.0	<code>^0.8.2</code> <code>^0.8.0</code>		yes	yes (4 asm blocks)	

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

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. 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)

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	✓	✓	✓

ERC721				
Function	Description	Exist	Tested	Verified
BalanceOf	Count all NFTs assigned to an owner	✓	✓	✓
OwnerOf	Find the owner of an NFT	✓	✓	✓
SafeTransferFrom	Transfers the ownership of an NFT from one address to another address	✓	✓	✓
SafeTransferFrom	See above - Difference is that this function has an extra data parameter	✓	✓	✓
TransferFrom	Transfer ownership of an NFT	✓	✓	✓
Approve	Change or reaffirm the approved address for an NFT	✓	✓	✓
SetApprovalForAll	Enable or disable approval for a third party ("operator") to manage all of `msg.sender`'s assets	✓	✓	✓
GetApproved	Get the approved address for a single NFT	✓	✓	✓
IsApprovedForAll	Query if an address is an authorized operator for another address	✓	✓	✓
SupportsInterface	Query if a contract implements an interface	✓	✓	✓
Name	Provides information about the name	✓	✓	✓
Symbol	Provides information about the symbol	✓	✓	✓
TokenURI	Provides information about the TokenUri	✓	✓	✓

Write functions of contract v1.0

GALIXGOVERNANCE	GALIXERC20	GALIXERC721	GALIXSTAKINGPOOL
addRewardPool	addMinter	addMinter	exit
addStakeOperator	approve	approve	getReward
addTokenOperator	burn	awardItem	initialize
grantRole	burnFrom	awardItemBySignature	notifyRewardAmount
initialize	createSnapshot	awardItems	pause
recoverWrongTokens	decreaseAllowance	awardItemWithFeeBySigna...	recoverWrongTokens
removeRewardPool	freeze	burn	renounceOwnership
removeStakeOperator	grantRole	grantRole	setRewardDistribution
removeTokenOperator	increaseAllowance	initialize	stake
renounceOwnership	initialize	lock	transferOwnership
renounceRole	mint	pause	unpause
revokeRole	mintFrom	removeMinter	withdraw
setInit	pause	renounceOwnership	
stakeDistribute	permit	renounceRole	
tokenMint	removeMinter	revokeRole	
transferOwnership	renounceOwnership	safeTransferFrom	
	renounceRole	safeTransferFrom	
	revokeRole	setApprovalForAll	
	transfer	setBaseURI	
	transferFrom	setLandLpAddress	
	transferOwnership	setProxyRegistryAddress	
	unfreeze	setTransferLock	
	unpause	transferFrom	
		transferOwnership	
		unlock	
		unpause	

Deployer cannot mint any new tokens

Name	Exist	Tested	Status
Deployer cannot mint	✓	✓	✗

Comments:

v1.0

- Minter role can mint new tokens



Deployer cannot burn or lock user funds

Name	Exist	Tested	Status
Deployer cannot lock	✓	✓	✗
Deployer cannot burn	✓	✓	✗

Comments:

v1.0

- Owner can lock user funds by
 - Freezing addresses
- Tokens
 - can be burned by the owner without allowance
 - can be burned by msg.sender

Deployer cannot pause the contract

Name	Exist	Tested	Status
Deployer cannot pause	✓	✓	✗

Comments:

v1.0

- Owner can pause contract



Overall checkup (Smart Contract Security)

Tested	Verified
✓	✓

Legend

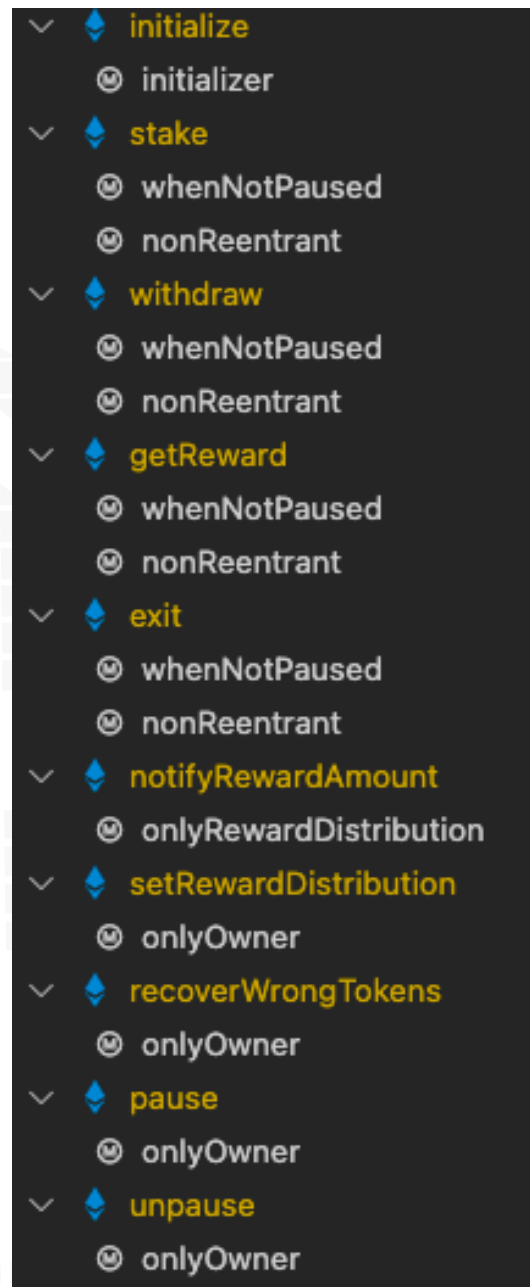
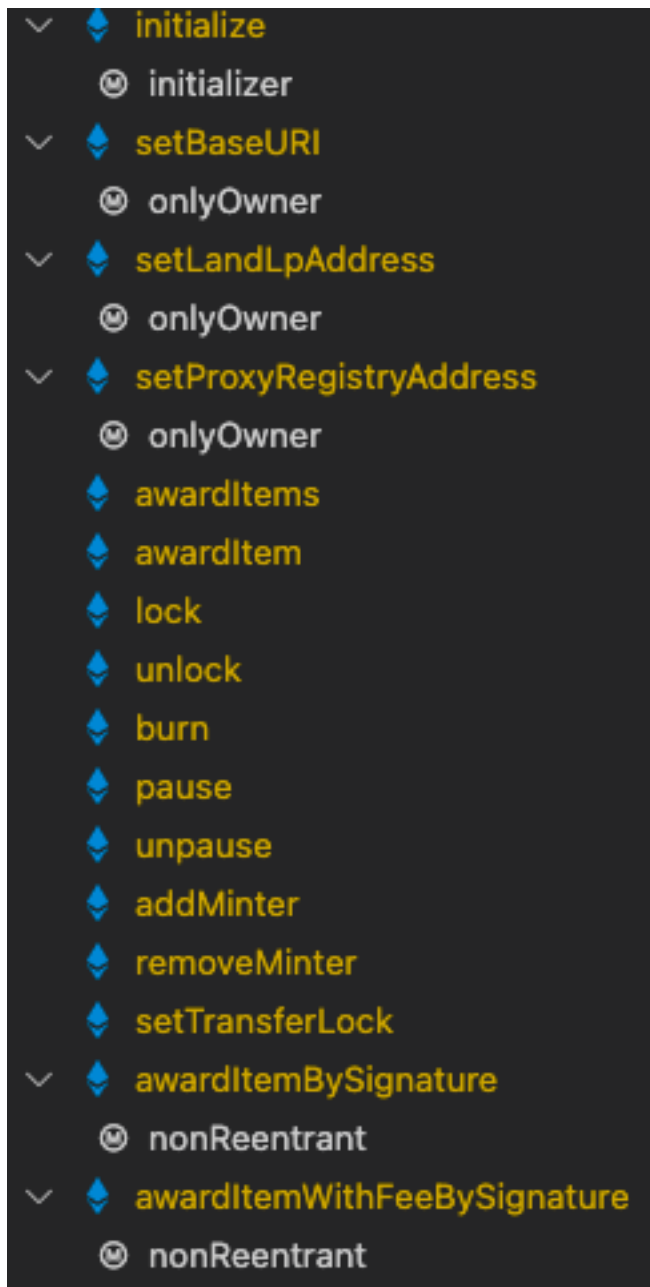
Attribute	Symbol
Verified / Checked	✓
Partly Verified	⚠
Unverified / Not checked	✗
Not available	—

Modifiers and public functions

v1.0

- initialize
 - initializer
- createSnapshot
- mintFrom
- mint
- burnFrom
- addMinter
- removeMinter
- pause
- unpause
- freeze
- unfreeze
- burn

- initialize
 - initializer
- setInit
 - onlyAdmin
- recoverWrongTokens
 - onlyAdmin
- addTokenOperator
 - onlyAdmin
- removeTokenOperator
 - onlyAdmin
- addStakeOperator
 - onlyAdmin
- removeStakeOperator
 - onlyAdmin
- addRewardPool
 - onlyAdmin
- removeRewardPool
 - onlyAdmin
- tokenMint
 - onlyTokenOperator
- stakeDistribute
 - onlyStakeOperator



Comments

- Deployer can set following addresses
 - GalixERC721
 - __baseURI
 - liquidityProviderAddress
 - proxyRegistryAddress
 - GalixGovernance
 - token
 - daoFund
 - stakePool
 - GalixStakingPool
 - rewardDistribution
- Existing Modifiers






























- GalixGovernance
 - notContract
 - onlyAdmin
 - onlyTokenOperator
 - onlyStakeOperator
- GalixStakingPool
 - onlyRewardDistribution
- There are several authorities which are authorized to call some functions, that means, if the owner is renounced, another address is still authorized to call functions
 - Be aware of this
- GalixERC20
 - Minter Role can
 - freeze/unfreeze addresses
 - burn from address without any allowance
 - mint new tokens
 - Admin role can
 - pause/unpause contract
 - add/remove minter role
 - create new snapshots
 - Freeze role
 - Was used to blacklist addresses
- GalixERC721
 - Minter role can
 - Mint new nfts
 - Lock/unlock nft id's
 -
 - Admin role can
 - add/remove minter role
 - Can set transferLock
 -
 - Pauser role can
 - Pause/unpause contract
- GalixGovernance
 - Owner can
 - recover wrong token from contract. We recommend to prevent passing token address to the "recoverWrongToken" function
 - Grant/revoke
 - token operator role
 - Stake operator role
 - Add/remove reward pool addresses

- The staking is only tracked with the event but not as a state variable
- GalixStakingPool
 - Users can withdraw immediately after staking. We recommend to add a delay for the withdraw function (at least 24 hours) that nobody can use multi call contracts to stake and withdraw it at the same time. Same for the rewards.
 - Owner can
 - Pause/unpause contract
 - Recover wrong tokens

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

Source Units in Scope

v1.0

Type	File	Logic Contracts	Interfaces	Lines	nLines	nSLOC	Comment Lines	Complex. Score	Capabilities
	contracts/GalixStakingPool.sol	1	—	233	229	145	43	136	—
	contracts/common/interfaces/IGalixERC20.sol	—	1	32	11	3	7	35	—
	contracts/common/interfaces/IERC20Mintable.sol	—	1	14	5	3	1	15	—
	contracts/common/interfaces/IWETH.sol	—	1	11	6	3	1	10	
	contracts/common/interfaces/IGalixERC721.sol	—	1	28	18	9	6	23	—
	contracts/common/SafeERC20Upgradeable.sol	1	—	98	76	36	30	24	
	contracts/common/ERC721Upgradeable.sol	1	—	424	389	164	171	149	
	contracts/common/ERC721URISStorageUpgradeable.sol	1	—	85	81	40	28	29	—
	contracts/common/ERC721PausableUpgradeable.sol	1	—	43	39	19	15	17	
	contracts/common/meta-transactions/ContextMixin.sol	1	—	26	22	18	2	16	
	contracts/common/meta-transactions/NativeMetaTransaction.sol	1	—	101	85	67	8	33	
	contracts/common/meta-transactions/EIP712Base.sol	1	—	75	66	47	11	32	
	contracts/common/ERC721BurnableUpgradeable.sol	1	—	35	35	18	13	19	
	contracts/GalixGovernance.sol	1	—	136	132	102	10	105	
	contracts/GalixERC721.sol	3	—	384	340	233	64	248	
	contracts/GalixERC20.sol	1	—	182	182	95	66	110	
	Totals	14	4	1907	1716	1002	476	1001	

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

High issues

No high issues

Medium issues

No medium issues

Low issues

Issue	File	Type	Line	Description
#1	All	A floating pragma is set	At the top of the source code	The current pragma Solidity directive is „^0.8.2“.
#2	GalixERC721	Missing Zero Address Validation (missing-zero-check)	58, 158, 162, 177, 215, 220,	Check that the address is not zero
#3	GalixGovernance	Missing Zero Address Validation (missing-zero-check)	79-81, 88, 93, 97, 101, 105, 109, 113	Check that the address is not zero
#4	GalixStakingPool	Missing Zero Address Validation (missing-zero-check)	54, 55, 56, 201, 209	Check that the address is not zero
#5	GalixERC20	Local variables shadowing	34	Rename the local variables that shadow another component
#6	GalixERC721	Local variables shadowing	58, 307	Rename the local variables that shadow another component

#7	GalixStakingPool	Missing Events Arithmetic	202	Emit an event for critical parameter changes
----	------------------	---------------------------	-----	--

Informational issues

Issue	File	Type	Line	Description
#1	GalixStakingPool	Error message is missing	136	Provide an error message for require statement
#2	Main	NatSpec documentation missing	-	If you started to comment your code, also comment all other functions, variables etc.
#3	Commons	Remove unused contracts	See description	We recommend you to remove unused contracts, libraries etc.
#4	GalixERC20	Remove comments	125-181	Unnecessary comments at the bottom of the file.
#5	GalixERC721	Remove comments	332-383	Unnecessary comments at the bottom of the file.

Audit Comments

We recommend you to use the special form of comments (NatSpec Format, Follow link for more information <https://docs.soliditylang.org/en/v0.5.10/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.

17. June 2022:

- Owner can deploy a new version of the contract which can change any limit and give owner new privileges
- Read whole report and modifiers section 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 Arguments	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 Variable	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 Attributes	CWE-330: Use of Insufficiently Random Values	PASSED
SW C-11 9	Shadowing State Variables	CWE-710: Improper Adherence to Coding Standards	NOT 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	NOT 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

 Solid
Proofed

Blockchain Security | Smart Contract Audits | KYC


MADE IN GERMANY