HACKEN

SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

Customer: Civic Technologies
Date: June 23, 2023

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

Document

Name	Smart Contract Code Review and Security Analysis Report for Civic Technologies
Approved By	Paul Fomichov | Lead Smart Contract Auditor at Hacken OU
Type	ERC3525 token
Platform	EVM
Language	Solidity
Methodology	Link
Website	https://www.civic.com/
Changelog	13.04.2023 - Initial Review 21.04.2023 - Second Review 23.06.2023 - Third Review

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

Table of contents

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

Introduction

Hacken OÜ (Consultant) was contracted by Civic Technologies (Customer) to conduct a Smart Contract Code Review and Security Analysis. This report presents the findings of the security assessment of the Customer's smart contracts.

Scope

The scope of the project includes the following smart contracts from the provided repository:

Initial review scope

Repository	https://github.com/identity-com/on-chain-identity-gateway/tree/develop/ethereum/smart-contract/
Commit	aa70a5c2f
Whitepaper	https://github.com/identity-com/gateway-whitepaper/blob/main/gateway-whitepaper.pdf
Functional Requirements	https://github.com/identity-com/on-chain-identity-gateway/blob/develop/ethereum/README.md
Technical Requirements	https://github.com/identity-com/on-chain-identity-gateway/blob/develop/ethereum/README.md
Contracts	File: interfaces/IGatewayToken.sol SHA3: 2e6ec3c969f9728d3bfa8659557cf5c5aaa979fe1d349593cc52af19c979384f File: interfaces/IParameterizedAccessControl.sol SHA3: ab775387d0b5a0fe33bf45e43645df594852d03f6334f1216d0890d29b9045f9 File: FlagsStorage.sol SHA3: bc68a94deef4de471bde0cedf6b3af9487977ba7f366e1f6d70d2410f153f433 File: FlexibleNonceForwarder.sol SHA3: c6a7cb55a1a1de8a2778e1cda68b897275c33b8e9b1c25f301d101359f87360a File: MultiERC2771Context.sol SHA3: 78f4c4bf1ca35ad7b95de7ad73810a007da725b7c00ad6a535407e71a2e9c135 File: TokenBitMask.sol SHA3: 3ddb9b799cb97b41155461a3c0e0aefd9a897126df2661193ba0dc1b91aa0832 File: TokenBitMask.sol SHA3: 3ddb9b799cb97b41155461a3c0e0aefd9a897126df2661193ba0dc1b91aa0832 File: library/Charge.sol SHA3: a462b838a72491bc335a872973709e5dcc571b9b475fdf76561c6e8a9b88dc90 File: Gated.sol SHA3: 3589e4a087de12dab8d7f67059c6ea5766de77c01404df7097adc336a241be09 File: Forwarder.sol SHA3: b7f66fd9562ec26c9c390915996dea8f87647639a8f647e26450c24c95cbc8d3

	File: interfaces/IGatewayTokenVerifier.sol SHA3: cf7a88d3c475d2f9f83ecc49457d03839efd479ffd79b0552636ccfaa46e6905 File: interfaces/IFlagsStorage.sol SHA3: 69fa17c537dabf76654de4256111cc3736a39fe7ee1b748060e410b686035cda File: interfaces/IForwarder.sol SHA3: 03b4b5c14afa62ec510c7a859153d27651422fb024f630a2bf53f91275fe8677 File: interfaces/IParameterizedAccessControl.sol SHA3: ab775387d0b5a0fe33bf45e43645df594852d03f6334f1216d0890d29b9045f9 File: interfaces/IERC721Expirable.sol SHA3: 0af56c7d0e1fc82ce2c21675ffa755de2b1b5361f97828ba0354e1787a428cc0 File: interfaces/IERC721Freezable.sol SHA3: 8eb6c8b09022c902b8a598854bc8f168690d71aa9a8eb1f480cdf27d01eddf7e File: interfaces/IGatewayTokenVerifier.sol SHA3: cf7a88d3c475d2f9f83ecc49457d03839efd479ffd79b0552636ccfaa46e6905 File: interfaces/IERC721Revokable.sol SHA3: 9a782892ad08a12b10f80a7268b1141016b8696cd854229457a25e5838c2df2f

Second review scope

Repository	https://github.com/identity-com/on-chain-identity-gateway/tree/develop/ethereum/smart-contract/
Commit	55547cc6
Whitepaper	https://github.com/identity-com/gateway-whitepaper/blob/main/gateway-whitepaper.pdf
Functional Requirements	https://github.com/identity-com/on-chain-identity-gateway/blob/develop/ethereum/README.md
Technical Requirements	https://github.com/identity-com/on-chain-identity-gateway/blob/develop/ethereum/README.md
Contracts	File: FlagsStorage.sol SHA3: c4d4f928b846cbfabf3e1a2526334f8ed58580196c85b270f07b56cef1770cd0 File: FlexibleNonceForwarder.sol SHA3: ad65ff40a85a941f9437ba6c9229f290b8eb47202b62c08c7f32f5603605f289 File: Gated.sol SHA3: 02cc2069b02bfe7b8c01dcc0b393bd5644e463126dce4432935b23cf3a974ffe File: GatewayToken.sol SHA3: 543a914ae634e3ad21bbc0f7c772acf8e5ab43565b71b1fea6811d3bff3830fd File: MultiERC2771Context.sol SHA3: 1ad3328c2980841c0fbe03c9b301f6eaf286b850e714578704e4f573498fe36a File: ParameterizedAccessControl.sol SHA3: c51e4ff435b6335732a16e393421a628b234d84cef39a695fee5c40e2719e46e File: TokenBitMask.sol SHA3: cfa2ed32fc10b58d166dcef792d0b395c56fa7fa672031d5f2267426055ab6f5

	File: interfaces/IERC721Expirable.sol SHA3: 97e5b4dbbb4f60163638be823826ee35895ef65439f6b951a5053fb15e2b98ad File: interfaces/IERC721Freezable.sol SHA3: 65eec00fa50e04322e432aea1e0f8e64e9f9b424191c87518f34a7c4a8ee8115 File: interfaces/IERC721Revokable.sol SHA3: a041c20c9f560149a73e92add3cffa5158fc63d95ea69bdcc5b405daafb87c86 File: interfaces/IFlagsStorage.sol SHA3: 0868dfa0fcaf12b4b62e67872baebca7aa7475b342bd3d401de4c2720901b9af File: interfaces/IForwarder.sol SHA3: cbda536359d178e4446c3e7e69e8476e99198d1a754d3abcf0edc09c6639eb73 File: interfaces/IGatewayToken.sol SHA3: 3f540f005122e233821223eea5b8dfd0b3d66a8331b855dbd015606cb5e67ddc File: interfaces/IGatewayTokenVerifier.sol SHA3: 37ebc4734e69d9faab9098da98a1c122ccf131badafa5292942e1bf00eb94f7a File: interfaces/IParameterizedAccessControl.sol SHA3: 819c9014797ecf7f84ca0aa0f7b0ce50ae08575d36a4f0258766789cfa5d5aff File: library/BitMask.sol SHA3: 07caf24b3c069eeaa33508a756ab0a14919bafb2f64b530e18876f58299c8de8 File: library/Charge.sol SHA3: 4bd2b5def4c785640daef6332e4db2b90d3eb6e48fab23e492bf70dbfd80327d File: library/CommonErrors.sol SHA3: 6a2f9de91d6e94c0074c906ce65548d275a3cfb0b5b70b567fd67ea2e010d2d6

Third review scope

Repository	https://github.com/identity-com/on-chain-identity-gateway/tree/develop/ethereum/smart-contract/
Commit	c7016ca7
Whitepaper	https://github.com/identity-com/gateway-whitepaper/blob/main/gateway-whitepaper.pdf
Functional Requirements	https://github.com/identity-com/on-chain-identity-gateway/blob/develop/ethereum/README.md
Technical Requirements	https://github.com/identity-com/on-chain-identity-gateway/blob/develop/ethereum/README.md
Contracts	File: ChargeHandler.sol SHA3: a4e71c797000fb6ec52f633582982c8dc9f875b5cc99ee816c413b6a1b8696f1 File: FlagsStorage.sol SHA3: c4d4f928b846cbfabf3e1a2526334f8ed58580196c85b270f07b56cef1770cd0 File: FlexibleNonceForwarder.sol SHA3: 9340372e096632993f324e81d1fbc7796b6e7cf970ccd0400b6ceee396f5a737 File: Gated.sol

	SHA3: 2feb933ee4fbf5cb39d3f25b01ac3ab51b02eeffa79e8061516e49761a8cda56 File: GatewayToken.sol SHA3: 57e0f0e8303ea5e7e2588df91bff92335881ee602025e70f7cbb912257968032 File: MultiERC2771Context.sol SHA3: 1ad3328c2980841c0fbe03c9b301f6eaf286b850e714578704e4f573498fe36a File: ParameterizedAccessControl.sol SHA3: c51e4ff435b6335732a16e393421a628b234d84cef39a695fee5c40e2719e46e File: TokenBitMask.sol SHA3: b2a25e98c7e756a58a0789113801bc9a5fa5d3065a7963f85c71ad37efb77c60 File: interfaces/IERC721Expirable.sol SHA3: 5346de063a8ec503a5c9dd455bb9c6c6e1c2508e0bc6e57cfef3f07c6980388c File: interfaces/IERC721Freezable.sol SHA3: 65eec00fa50e04322e432aea1e0f8e64e9f9b424191c87518f34a7c4a8ee8115 File: interfaces/IERC721Revokable.sol SHA3: a041c20c9f560149a73e92add3cffa5158fc63d95ea69bdcc5b405daafb87c86 File: interfaces/IFlagsStorage.sol SHA3: 0868dfa0fcaf12b4b62e67872baebca7aa7475b342bd3d401de4c2720901b9af File: interfaces/IForwarder.sol SHA3: cbda536359d178e4446c3e7e69e8476e99198d1a754d3abcf0edc09c6639eb73 File: interfaces/IGatewayToken.sol SHA3: 363422f5a86b7a2a8884dd1bd57b30b319b00a01a9fb6c1b906dabab16667c4c File: interfaces/IGatewayTokenVerifier.sol SHA3: cf7a88d3c475d2f9f83ecc49457d03839efd479ffd79b0552636ccfaa46e6905 File: interfaces/IParameterizedAccessControl.sol SHA3: 819c9014797ecf7f84ca0aa0f7b0ce50ae08575d36a4f0258766789cfa5d5aff File: library/BitMask.sol SHA3: 07caf24b3c069eeaa33508a756ab0a14919bafb2f64b530e18876f58299c8de8 File: library/Charge.sol SHA3: 32bdaacdca663ecfdb0cacd29ae05a4b389a33e0124d2780ddb2edbab6621704 File: library/CommonErrors.sol SHA3: 6a2f9de91d6e94c0074c906ce65548d275a3cfb0b5b70b567fd67ea2e010d2d6

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

Severity Definitions

Risk Level	Description
Critical	Critical vulnerabilities are usually straightforward to exploit and can lead to the loss of user funds or contract state manipulation by external or internal actors.
High	High vulnerabilities are usually harder to exploit, requiring specific conditions, or have a more limited scope, but can still lead to the loss of user funds or contract state manipulation by external or internal actors.
Medium	Medium vulnerabilities are usually limited to state manipulations but cannot lead to asset loss. Major deviations from best practices are also in this category.
Low	Low vulnerabilities are related to outdated and unused code or minor Gas optimization. These issues won't have a significant impact on code execution but affect code quality

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

Executive Summary

The score measurement details can be found in the corresponding section of the scoring methodology.

Documentation quality

The total Documentation Quality score is **10** out of **10**.

- Functional requirements are provided and complete.
- Technical description is provided.

Code quality

The total Code Quality score is **10** out of **10**.

- The development environment is configured.
- The code follows the official Solidity Style Guide.

Test coverage

Code coverage of the project is **96.58%** (branch coverage).

- Deployment and basic user interactions are covered with tests.
- Some utility functions are not covered by tests.

Security score

As a result of the audit, the code contains **no** issues. The security score is **10** out of **10**.

All found issues are displayed in the “Findings” section.

Summary

According to the assessment, the Customer's smart contract has the following score: **9.9**. The system users should acknowledge all the risks summed up in the risks section of the report.

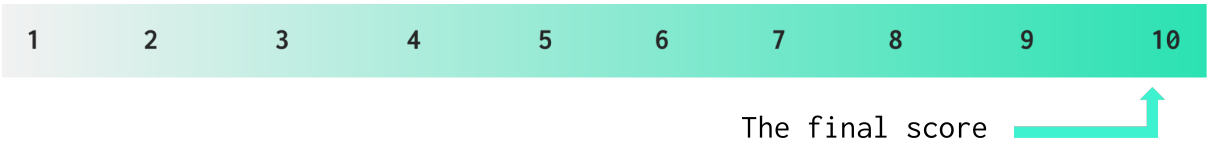

Table. The distribution of issues during the audit

Review date	Low	Medium	High	Critical
13 April 2023	13	2	1	0
21 April 2023	0	0	0	0
23 June 2023	0	0	0	0

Risks

- The Super Admin role has permissions for performing any change in the state of the protocol. It makes it very centralized and can impact security. Although it can not be assured by the code, it is recommended to use a multisig wallet with at least ⅗ transactions signing policy for the Super Admin role.

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

System Overview

The EVM Gateway Protocol is a standard that enables smart contracts on the Ethereum Virtual Machine (EVM) to implement access control constraints using Gateway Tokens (GTs). GTs are non-transferable, semi-fungible tokens that conform to the ERC-20 interface and are issued by "Gatekeepers" who verify the eligibility of users for a specific GT. The Gateway Protocol is open and permissionless, allowing anyone to become a gatekeeper and issue GTs. Gatekeepers operate within a Gatekeeper Network, a group of gatekeepers that work together to issue GTs for a particular use case. The client smart contract specifies the trusted Gatekeeper Network, and if a user has a valid GT from that network, the transaction can proceed. GTs have features such as expiry dates, freeze/unfreeze options, and revocation, giving gatekeepers more control over the token's lifecycle. The Gateway Protocol is designed to be easily integrated into existing smart contracts, and it does not prescribe a specific mechanism for obtaining GTs, allowing gatekeepers to design their own user onboarding flows.

The scope is composed by the following contracts:

- GatewayToken - ERC3525 Token contract, takes care of minting and handling the token states
- MultiERC2771Context - Extends ContextUpgradeable with ERC2771 trustedForwarders
- ParametrizedAccessControl - Network level access control
- TokenBitMask - An internal smart contract for Gateway Token implementation that stores KYC flags per identity token in a bitmask
- FlagsStorage - The main contract to store KYC-related flags for Gateway Token System
- FlexibleNonceForwarder - ERC2771 Forwarder contract extended to allow for flexible nonces
- Gated - Utility contract to check if msg.sender has a valid token on a specific network

Privileged roles

- Super admins of the *FlagsStorage* contract can arbitrarily add, delete and modify the flags stored. It is therefore entitled to impersonate or change the logic of critical components of the system at will.
- Super admins of the *GatewayToken* contract can withdraw locked funds, change metadata descriptor, update forwarders and update flags storage.

Recommendations

- Some uint variables do not have their size explicitly specified. It is better to properly specify the size of the variable to avoid readability issues (contracts/interfaces/IGatewayToken.sol,

contracts/library/BitMask.sol, contracts/library/CommonErrors.sol, contracts/GatewayToken.sol, contracts/MultiERC2771Context.sol).

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

Checked Items

We have audited the Customers' smart contracts for commonly known and specific vulnerabilities. Here are some items considered:

Item	Type	Description	Status
Default Visibility	SWC-100 SWC-108	Functions and state variables visibility should be set explicitly. Visibility levels should be specified consciously.	Passed
Integer Overflow and Underflow	SWC-101	If unchecked math is used, all math operations should be safe from overflows and underflows.	Not Relevant
Outdated Compiler Version	SWC-102	It is recommended to use a recent version of the Solidity compiler.	Passed
Floating Pragma	SWC-103	Contracts should be deployed with the same compiler version and flags that they have been tested thoroughly.	Passed
Unchecked Call Return Value	SWC-104	The return value of a message call should be checked.	Passed
Access Control & Authorization	CWE-284	Ownership takeover should not be possible. All crucial functions should be protected. Users could not affect data that belongs to other users.	Passed
SELFDESTRUCT Instruction	SWC-106	The contract should not be self-destructible while it has funds belonging to users.	Not Relevant
Check-Effect-Interaction	SWC-107	Check-Effect-Interaction pattern should be followed if the code performs ANY external call.	Passed
Assert Violation	SWC-110	Properly functioning code should never reach a failing assert statement.	Passed
Deprecated Solidity Functions	SWC-111	Deprecated built-in functions should never be used.	Passed
Delegatecall to Untrusted Callee	SWC-112	Delegatecalls should only be allowed to trusted addresses.	Not Relevant
DoS (Denial of Service)	SWC-113 SWC-128	Execution of the code should never be blocked by a specific contract state unless required.	Passed

Race Conditions	SWC-114	Race Conditions and Transactions Order Dependency should not be possible.	Passed
Authorization through tx.origin	SWC-115	tx.origin should not be used for authorization.	Passed
Block values as a proxy for time	SWC-116	Block numbers should not be used for time calculations.	Passed
Signature Unique Id	SWC-117 SWC-121 SWC-122 EIP-155 EIP-712	Signed messages should always have a unique id. A transaction hash should not be used as a unique id. Chain identifiers should always be used. All parameters from the signature should be used in signer recovery. EIP-712 should be followed during a signer verification.	Passed
Shadowing State Variable	SWC-119	State variables should not be shadowed.	Passed
Weak Sources of Randomness	SWC-120	Random values should never be generated from Chain Attributes or be predictable.	Not Relevant
Incorrect Inheritance Order	SWC-125	When inheriting multiple contracts, especially if they have identical functions, a developer should carefully specify inheritance in the correct order.	Passed
Calls Only to Trusted Addresses	EEA-Level-2 SWC-126	All external calls should be performed only to trusted addresses.	Passed
Presence of Unused Variables	SWC-131	The code should not contain unused variables if this is not justified by design.	Passed
EIP Standards Violation	EIP	EIP standards should not be violated.	Passed
Assets Integrity	**Custom**	Funds are protected and cannot be withdrawn without proper permissions or be locked on the contract.	Passed
User Balances Manipulation	**Custom**	Contract owners or any other third party should not be able to access funds belonging to users.	Not Relevant
Data Consistency	**Custom**	Smart contract data should be consistent all over the data flow.	Passed

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

Flashloan Attack	**Custom**	When working with exchange rates, they should be received from a trusted source and not be vulnerable to short-term rate changes that can be achieved by using flash loans. Oracles should be used.	Not Relevant
Token Supply Manipulation	**Custom**	Tokens can be minted only according to rules specified in a whitepaper or any other documentation provided by the Customer.	Not Relevant
Gas Limit and Loops	**Custom**	Transaction execution costs should not depend dramatically on the amount of data stored on the contract. There should not be any cases when execution fails due to the block Gas limit.	Passed
Style Guide Violation	**Custom**	Style guides and best practices should be followed.	Passed
Requirements Compliance	**Custom**	The code should be compliant with the requirements provided by the Customer.	Passed
Environment Consistency	**Custom**	The project should contain a configured development environment with a comprehensive description of how to compile, build and deploy the code.	Passed
Secure Oracles Usage	**Custom**	The code should have the ability to pause specific data feeds that it relies on. This should be done to protect a contract from compromised oracles.	Not Relevant
Tests Coverage	**Custom**	The code should be covered with unit tests. Test coverage should be sufficient, with both negative and positive cases covered. Usage of contracts by multiple users should be tested.	Passed
Stable Imports	**Custom**	The code should not reference draft contracts, which may be changed in the future.	Passed

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

Findings

■■■■ Critical

No critical severity issues were found.

■■■ High

H01. Inconsistent Data

The function *execute()* from FlexibleNonceForwarder accepts Ether and does not verify if the transaction value (*msg.value*) is the same as the request value (*req.value*) and does not return unused Ether to the caller. If the transaction value is not enough to perform the forwarding, it will use funds from the contract (previously collected funds that were not used).

Path: ./contracts/FlexibleNonceForwarder.sol : execute()

Recommendation: Check if the *msg.value* is equal to the *req.value* and return unused Ether to the caller.

Found in: aa70a5c2

Status: Fixed (Revised commit: 55547cc6)

■■ Medium

M01. Only EOA Allowed

The function *withdraw()* restricts any incoming smart contract calls by performing high level *payable(address).transfer()*.

This causes inability to call the functionality from multisig wallets, DAO accounts or any smart contract.

Path: ./contracts/GatewayToken.sol : withdraw()

Recommendation: Remove the restriction or provide alternative ways for smart contract interactions.

Found in: aa70a5c2

Status: Fixed (Revised commit: 55547cc6)

M02. Best Practice Violation

Forwarder contract is a direct implementation of OpenZeppelin's MinimalForwarder contract, with no added functionalities.

Its use in production is discouraged by OpenZeppelin itself.

Path: ./contracts/Forwarder.sol

Recommendation: Remove the contract and only use FlexibleNonceForwarder.

Found in: aa70a5c2

Status: Fixed (Revised commit: 55547cc6)

Low

L01. Redundant Override Keyword

override keyword is used on state variables when not needed.

Path: ./contracts/FlagsStorage.sol : superAdmin, supportedFlagsMask, flagIndexes

Recommendation: Remove redundant code.

Found in: aa70a5c2

Status: Fixed (Revised commit: 55547cc6)

L02. Functions that Can Be Declared External

public functions that are never called by the contract should be declared “external” to save Gas.

Notice: it is also applicable to the *initialize* function in upgradable contracts. There is no advantage in declaring them public if the contract is not inherited.

Paths: ./contracts/FlagsStorage.sol : initializer(), updateSuperAdmin(), addFlag(), addFlags(), removeFlag(), removeFlags(), isFlagSupported()

./contracts/FlexibleNonceForwarder : execute(), getNonce()

./contracts/GatewayToken.sol : initialize(), setMetadataDescriptor(), addForwarder(), removeForwarder(), transferDAOManager(), getTokenBitmask(), setBitmask()

./contracts/ParameterizedAccessControl.sol : setSuperAdmin()

Recommendation: Use the external attribute for functions never called from the contract.

Found in: aa70a5c2

Status: Fixed (Revised commit: 55547cc6)

L03. Reading Array Length in a Loop

Array length should be saved in a local variable instead of being computed in each loop cycle during the condition check.

Paths: ./contracts/FlagsStorage.sol : addFlags(), removeFlags()

./contracts/GatewayToken.sol : _getTokenIdsByOwnerAndNetwork()

Recommendation: Save the array length in a variable and use that variable in the for loop condition.

Found in: aa70a5c2

Status: Fixed (Revised commit: 55547cc6)

L04. Variable Can Be Set Immutable

Variable *_blockAgeTolerance* can be declared immutable to save Gas on computations.

Path: ./contracts/FlexibleNonceForwarder.sol

Recommendation: Declare mentioned variables as mentioned.

Found in: aa70a5c2

Status: Fixed (Revised commit: 55547cc6)

L05. Floating Pragma

Locking the pragma helps ensure that contracts do not accidentally get deployed using, for example, an outdated compiler version that might introduce bugs that affect the contract system negatively. The project uses *floating pragma ^0.8.9*.

Path: all contracts

Recommendation: Consider locking the pragma version whenever possible and avoid using a floating pragma in the final deployment.

Found in: aa70a5c2

Status: Fixed (Revised commit: 55547cc6)

L06. Zero Address Check

Address parameters are being used without checking against the possibility of *0x0*.

This can lead to unwanted external calls to *0x0*.

Path: ./contracts/GatewayToken.sol : initialize()

Recommendation: Implement zero address checks.

Found in: aa70a5c2

Status: Fixed (Revised commit: 55547cc6)

L07. Best Practice Violation

Since OpenZeppelin's contracts v4.6.0 it is recommended to use *_disableInitializers()* in the constructor instead of the *initializer* modifier.

Paths: ./contracts/FlagsStorage.sol : constructor(),
./contracts/GatewayToken.sol : constructor(),
./contracts/MultiERC2771Context.sol : constructor()

Recommendation: Use *_disableInitializers* instead of *initializer* on constructor as recommended by OpenZeppelin docs.

Found in: aa70a5c2

Status: Fixed (Revised commit: 55547cc6)

L08. Unused Variable

Unused variables should be removed from the contracts. Unused variables are allowed in Solidity and do not pose a direct security issue. It is best practice to avoid them as they can cause an increase in computations (and unnecessary Gas consumption) and decrease readability.

The variable *controller* is never used.

Path: ./contracts/GatewayToken.sol

Recommendation: Remove unused variable.

Found in: aa70a5c2

Status: Fixed (Revised commit: 55547cc6)

L09. Function Can Be Pure

The function *transfersRestricted()* does not read or modify the variables of the state and, due to that, should be declared pure.

This can lower Gas taxes.

Path: ./contracts/GatewayToken.sol : transfersRestricted()

Recommendation: Change function state mutability to pure.

Found in: aa70a5c2

Status: Fixed (Revised commit: 55547cc6)

L10. Redundant Function

withdraw() function is currently needed because of the ERC3525 payable *approve()* function, which can lead to funds lock if users send funds by mistake to the contract.

A cleaner way to solve the issue would be to override the ERC3525 *approve()* function, letting it revert if *msg.value* is greater than zero. This way, funds will not reach the contract in the first place.

Path: ./contracts/GatewayToken.sol : withdraw()

Recommendation: Override the ERC3525 *approve()* function, and extend it by triggering a revert if msg.value is greater than zero.

Found in: aa70a5c2

Status: Fixed (Revised commit: 55547cc6)

L11. Empty Constructor

In the contract *Forwarder* the constructor is empty, which makes it redundant due to default Solidity behavior to create an empty constructor if it is not included in code.

This makes redundant parts of code.

Path: ./contracts/Forwarder.sol

Recommendation: Remove redundant parts of code.

Found in: aa70a5c2

Status: Fixed (Revised commit: 55547cc6)

L12. Unfinalized NatSpec

In the contract *Forwarder* the NatSpec placed before the contract definition is not finalized.

Path: ./contracts/Forwarder.sol

Recommendation: Finish the NatSpec comment block.

Found in: aa70a5c2

Status: Fixed (Revised commit: 55547cc6)

L13. Style Guide Violation

The function ordering is not following the official guidelines.

Path: ./contracts/GatewayToken.sol

Recommendation: Follow the official Solidity guidelines.

Found in: aa70a5c2

Status: Fixed (Revised commit: 55547cc6)

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

Disclaimers

Hacken Disclaimer

The smart contracts given for audit have been analyzed based on best industry practices at the time of the writing of this report, with cybersecurity vulnerabilities and issues in smart contract source code, the details of which are disclosed in this report (Source Code); the Source Code compilation, deployment, and functionality (performing the intended functions).

The report contains no statements or warranties on the identification of all vulnerabilities and security of the code. The report covers the code submitted and reviewed, so it may not be relevant after any modifications. Do not consider this report as a final and sufficient assessment regarding the utility and safety of the code, bug-free status, or any other contract statements.

While we have done our best in conducting the analysis and producing this report, it is important to note that you should not rely on this report only — we recommend proceeding with several independent audits and a public bug bounty program to ensure the security of smart contracts.

English is the original language of the report. The Consultant is not responsible for the correctness of the translated versions.

Technical Disclaimer

Smart contracts are deployed and executed on a blockchain platform. The platform, its programming language, and other software related to the smart contract can have vulnerabilities that can lead to hacks. Thus, the Consultant cannot guarantee the explicit security of the audited smart contracts.