SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

Customer: Civic Technologies, Inc.
Date: 31 Oct, 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, Inc
Approved By	Przemyslaw Swiatowiec | Lead Solidity SC Auditor at Hacken OÜ Kornel Światłowski | SC Auditor at Hacken OÜ Roman Tiutiun | SC Auditor at Hacken OÜ
Tags	ERC3525
Platform	EVM
Language	Solidity
Methodology	Link
Website	https://www.civic.com/
Changelog	23.10.2023 - Initial Review 31.10.2023 - Second Review

Table of contents

Introduction

Hacken OÜ (Consultant) was contracted by Civic Technologies, Inc. (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.

System Overview

GATEWAY PROTOCOL is a standard that allows smart contracts to add access control constraints, requiring that a user has a valid Gateway Token (GT) in order to interact with the smart contract. with the following contracts:

- *GatewayToken.sol* – the contract is responsible for managing Identity.com KYC gateway tokens; those tokens represent completed KYC with attached identity. Gateway tokens using ERC721 standard with custom extensions.
- *ChargeHandler.sol* – is an internal library used by the Gatekeeper to handle charges made to the gatekeeper on gateway token issuance or refresh.
- *FlagsStorage.sol* - is the main contract to store KYC-related flags for Gateway Token System. KYC flags are identifiable by short identifiers in bytes32 strings. After adding flags those bit indexes could be used by GatewayToken implementations to associate flags per token.
- *Gated.sol* - the modifier utility that allows function to only be called by a sender that has a valid gateway token.
- *FlexibleNonceForwarder.sol* - is the typed data standard specifies a method in which data to be signed (a transaction, in this case) is structured according to a schema, which is agreed between the signer and verifier. While the primary motivation behind this standard is to increase the security of user-facing wallets, by allowing them to display the data to be signed in a human-readable format, it also has benefits for the gateway protocol.
- *GatedERC2771.sol* - an abstract contract designed to work with a gateway token contract and a gatekeeper network, both of which are set during the contract's deployment.
- *GatedERC2771Upgradeable.sol* - is an abstract contract designed to work with a gateway token contract and a gatekeeper network.
- *MultiERC2771Context.sol* - is a Context variant with ERC2771 support for multiple trusted forwarders.
- *MultiERC2771ContextUpgradeable.sol* - is an abstract contract that extends the Context contract from OpenZeppelin. The contract is designed to support multiple trusted forwarders in the context of ERC2771, a standard for meta transactions.

- *ParameterizedAccessControl.sol* - contract module implements role-based access control mechanisms. This is a lightweight version that doesn't allow enumerating role members except through off-chain means by accessing the contract event logs.
- *TokenBitMask.sol* - contract is an internal smart contract for Gateway Token implementation that stores KYC flags per identity token in a bitmask.
- *BitMask.sol* - is a library that provides functions for manipulating bits in a 256-bit unsigned integer.

Privileged roles

The *superAdmin* of the *FlagsStorage.sol contract:*

- updateSuperAdmin() - transferring ownership to new super admin.
- *addFlag()* - adding a new flag into the gateway token system.
- *addFlags()* - adding multiple flags into the gateway token system.
- *removeFlag()* - removing existing flag from gateway token system.
- *removeFlags()* - removing multiple existing flags from gateway token system.

The *GatewayToken.sol* contract control access to critical functionalities and has the following privileges:

- *DAO_MANAGER_ROLE* - the DAO Manager has the ability to transfer their role to a new address, add or remove network authorities and gatekeepers, and create new networks. They can also set the contract to be governed by a DAO.
- *GATEKEEPER_ROLE* - has authority to mint, burn, freeze, and unfreeze tokens. They can also set the expiration of tokens and the bitmask for tokens. Essentially, they have control over the lifecycle of tokens.
- *NETWORK_AUTHORITY_ROLE* - has authority to manage the network. They can add or remove gatekeepers, set network features, and rename the network. They also have the ability to add or remove other network authorities.
- *onlySuperAdmin()* - modifier checks if the caller of the function is the super admin in the following functions:
  - *setMetadataDescriptor()*
  - *addForwarder()*
  - *removeForwarder()*
  - *updateFlagsStorage()*
  - *updateFlagsStorage()*

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**.

Code quality

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

Test coverage

Code coverage of the project is **93.84%** (branch coverage)

- Deployment and basic user interactions are covered with tests.
- MultiERC2771Context.sol contact is not covered well.

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: **9.8**. The system users should acknowledge all the risks summed up in the risks section of the report.

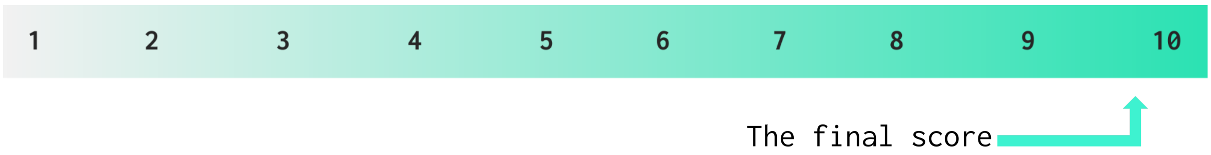

Table. The distribution of issues during the audit

Review date	Low	Medium	High	**Critical**
23 October 2023	2	1	0	0
31 October 2023	0	0	0	0

Risks

- The *FlagsStorage* contract is highly centralized, with the super admin having the power to add or remove flags, update the super admin, and authorize contract upgrades. This could be a risk if the super admin's private key is compromised.

- The system design necessitates trust in both the *Forwarder* and *Gatekeeper* components. The *Gatekeeper* holds the crucial responsibility of validating charge data. Failing to validate this data could result in potential charges to other users for gateway tokens, as users are required to set allowances for charges.
- To ensure backward compatibility, the *REMOVE_GATEKEEPER_INVALIDATES_TOKENS* feature exclusively affects newly generated tokens post-contract upgrade.
- The setting of the *CHARGE_CALLER_ROLE* must be approached with utmost caution. If a malicious user gains access to the *handleCharge* function, they could potentially steal tokens that have been authorized for expenditure by the *ChargeHandler* contract.

Findings

■■■■ Critical

No critical severity issues were found.

■■■ High

No high severity issues were found.

■■ Medium

M01. Super admin can delete all other super admins causing inability to manage GatewayToken contract

Impact	High
Likelihood	Low

The *revokeSuperAdmin()* function is designed to remove the super admin role from a specified address. This function is exclusively executable by super admin. However, a critical scenario could arise where the last super admin decides to revoke their own role. This would result in a situation where no other addresses can be assigned the super admin role. Such a situation would severely impede the effective management and modification of the *GatewayToken* contract.

Path: ./contracts/ParameterizedAccessControl.sol: revokeSuperAdmin()

Recommendation: It is recommended to implement a safeguard mechanism to prevent the last super admin from revoking their own role. This would ensure that there is always at least one address with the super admin role, enabling the ongoing management and modification of the *GatewayToken* contract.

Found in: 4b7bc57e40497b44e694b75b76603f046de7bbaa

Status: Fixed (Revised commit: d62d94)

Remediation: The *revokeSuperAdmin* function was updated, so super admins cannot remove themselves.

■ Low

L01. Missing zero address validation

Impact	Low
Likelihood	Low

A function fails to validate whether an address is set to zero before accepting it as an argument.

This oversight could result in accidentally adding an incorrect address to the contract, potentially leading to unexpected behavior or vulnerabilities. Proper address validation should be implemented to prevent this risk.

Paths: ./contracts/GatewayToken.sol: updateFlagsStorage()
./contracts/GatedERC2771.sol: constructor().
./contracts/GatedERC2771Upgradeable.sol:
__GatedERC2771Upgradeable_init().

Recommendation: It is recommended to implement zero address validation.

Found in: 4b7bc57e40497b44e694b75b76603f046de7bbaa

Status: Fixed (Revised commit: 71a6ad5).

Remediation: Zero address checks were introduced.

L02. ERC-2771 implementation differ from proposal - underflow risk

Impact	Low
Likelihood	Low

The implementation of *ERC-2771*, specifically in the *MultiERC2771Context* and *MultiERC2771ContextUpgradeable* contracts, deviates from the accepted proposal. According to the *ERC-2771* standard, Transaction Signer extraction from the Recipient contract should involve the following steps:

1. Check that the *Forwarder* is trusted.
2. Extract the *Transaction Signer* address from the last 20 bytes of the call data and use that as the original sender of the transaction (instead of *msg.sender*).
3. If the *msg.sender* is not a trusted forwarder **(or if the *msg.data* is shorter than 20 bytes)**, then return the original *msg.sender* as it is.

In the Civic Technologies implementation, checks for *msg.data* length (if it is shorter than 20 bytes) are omitted. This omission can result in an underflow error when *msg.data's* length is less than 20. The affected functions include:

_msgSender():

```
if (isTrustedForwarder(msg.sender)) {
  assembly {
    // sub(calldatasize(), 20) == calldatasize() - 20
    sender := shr(96, calldataload(sub(calldatasize(), 20)))
  }
```

and *_msgData()*:

```
if (isTrustedForwarder(msg.sender)) {
  return msg.data[:msg.data.length - 20];
}
```

Paths: ./contracts/MultiERC2771Context.sol: _msgData(), _msgSender(), ./contracts/MultiERC2771ContextUpgradeable.sol: _msgData(), _msgSender(),

Recommendation: It is recommended add *msg.data.length >= 20* check to prevent underflow.

Found in: 4b7bc57e40497b44e694b75b76603f046de7bbaa

Status: Fixed (Revised commit: d62d94)

Remediation: The length of *msg.data* check was introduced in aforementioned functions.

Informational

I01. Floating pragma

The project uses floating pragma (*>=0.8.19*).

This may result in the contracts being deployed using the wrong pragma version, which is different from the one they were tested with. For example, they might be deployed using an outdated pragma version, which may include bugs that affect the system negatively.

Path: ./contracts: *

Recommendation: Consider locking the pragma version whenever possible and avoid using a floating pragma in the final deployment. Consider known bugs (https://github.com/ethereum/solidity/releases) for the compiler version that is chosen.

Found in: 4b7bc57e40497b44e694b75b76603f046de7bbaa

Status: Fixed (Revised commit: d62d94)

Remediation: The floating pragma is now locked on 0.8.19 version.

I02. Redundant check of user allowance

The *_handleERC20Charge()* function is designed to handle *ERC20* transfers with *safeERC20*. This function checks the transfer allowance twice, which increases Gas consumption.

The first check is performed in *_handleERC20Charge()*:

```
uint256 allowance = token.allowance(charge.tokenSender, address(this));
```

```
if (allowance < charge.value) {
    revert Charge__IncorrectAllowance(allowance, charge.value);
}
```

And a second in the *transferFrom()* function of *ERC20* contract.

Path: ./contracts/ChargeHandler.sol: _handleERC20Charge()

Recommendation: It is recommended to remove the first allowance check since an equivalent check is already performed internally in the *ERC20* contract with every *safeTransfer()*.

Found in: 4b7bc57e40497b44e694b75b76603f046de7bbaa

Status: Fixed (Revised commit: d62d94)

Remediation: Redundant allowance check was removed.

I03. Redundant imports

Following redundant imports were observed:

- The contract *OwnableUpgradeable* is imported in the *FlagsStorage* contract but never used.
- The enum *ChargeType* is imported in the *GatewayToken* contract but never used.

This redundancy in import operations has the potential to result in unnecessary Gas consumption during deployment and impacts the code quality.

Paths: ./contracts/FlagsStorage.sol: OwnableUpgradeable,

./contracts/GatewayToken.sol: ChargeType,

Recommendation: It is recommended to remove redundant imports, and ensure that the contract is imported only in the required locations, avoiding unnecessary duplications.

Found in: 4b7bc57e40497b44e694b75b76603f046de7bbaa

Status: Fixed (Revised commit: d62d94)

Remediation: Redundant imports were removed.

I04. State variables can be declared immutable

Compared to regular state variables, the Gas costs of constant and immutable variables are much lower. Immutable variables are evaluated once at construction time and their value is copied to all the places in the code where they are accessed.

Following variables can be declared as immutable:

- *_gatewayTokenContract* and *_gatekeeperNetwork* values are set in the constructor of *Gated* and *GatedERC2771* contracts and cannot be changed.

Paths: ./contracts/Gated.sol: _gatewayTokenContract, _gatekeeperNetwork,

./contracts/GatedERC2771.sol: _gatewayTokenContract, _gatekeeperNetwork

Recommendation: It is recommended to declare mentioned variables as immutable.

Found in: 4b7bc57e40497b44e694b75b76603f046de7bbaa

Status: Fixed (Revised commit: 71a6ad5)

Remediation: Aforementioned variables were declared as immutable.

I05. Missing validation if a DaoManager is a contract account

The *createNetwork()* function, the *daoManger* address is validated to ensure it is a contract account. This address is subsequently granted the *DAO_MANAGER_ROLE* and *NETWORK_AUTHORITY_ROLE* roles.

Regrettably, this check is omitted in the *transferDAOManager()* function. Consequently, externally owned addresses (EOA) can be designated as the new DAO manager and receive both the *DAO_MANAGER_ROLE* and *NETWORK_AUTHORITY_ROLE* roles.

Paths: ./contracts/GatewayToken.sol: transferDAOManager(),

Recommendation: It is recommended to include validation of address to ensure that the provided address is a contract address.

Found in: 4b7bc57e40497b44e694b75b76603f046de7bbaa

Status: Fixed (Revised commit: d62d94)

Remediation: Check in the *transferDAOManager()* was introduced to verify that new *daoManger* is a contract account.

I06. Missing event for critical value updation

Events for critical state changes should be emitted for tracking things off-chain. It was observed that event emission is missing for following functions: *initialize(), removeForwarder(),* and *addForwarder()* within the *GatewayToken* contract.

Path: ./contracts/GatewayToken.sol: initialize(), addForwarder(), removeForwarder()

Recommendation: Consider emitting events for tracking changes in aforementioned functions.

Found in: 4b7bc57e40497b44e694b75b76603f046de7bbaa

Status: Fixed (Revised commit: d62d94)

Remediation: Events were added to aforementioned functions.

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.

Appendix 1. Severity Definitions

When auditing smart contracts Hacken is using a risk-based approach that considers the potential impact of any vulnerabilities and the likelihood of them being exploited. The matrix of impact and likelihood is a commonly used tool in risk management to help assess and prioritize risks.

The impact of a vulnerability refers to the potential harm that could result if it were to be exploited. For smart contracts, this could include the loss of funds or assets, unauthorized access or control, or reputational damage.

The likelihood of a vulnerability being exploited is determined by considering the likelihood of an attack occurring, the level of skill or resources required to exploit the vulnerability, and the presence of any mitigating controls that could reduce the likelihood of exploitation.

Risk Level	High Impact	Medium Impact	Low Impact
High Likelihood	Critical	High	Medium
Medium Likelihood	High	Medium	Low
Low Likelihood	Medium	Low	Low

Risk Levels

Critical: Critical vulnerabilities are usually straightforward to exploit and can lead to the loss of user funds or contract state manipulation.

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.

Medium: Medium vulnerabilities are usually limited to state manipulations and, in most cases, cannot lead to asset loss. Contradictions and requirements violations. Major deviations from best practices are also in this category.

Low: Major deviations from best practices or major Gas inefficiency. These issues won't have a significant impact on code execution, and do not affect security score but can affect code quality score.

Impact Levels

High Impact: Risks that have a high impact are associated with financial losses, reputational damage, or major alterations to contract state. High impact issues typically involve invalid calculations, denial of service, token supply manipulation, and data consistency, but are not limited to those categories.

Medium Impact: Risks that have a medium impact could result in financial losses, reputational damage, or minor contract state manipulation. These risks can also be associated with undocumented behavior or violations of requirements.

Low Impact: Risks that have a low impact cannot lead to financial losses or state manipulation. These risks are typically related to unscalable functionality, contradictions, inconsistent data, or major violations of best practices.

Likelihood Levels

High Likelihood: Risks that have a high likelihood are those that are expected to occur frequently or are very likely to occur. These risks could be the result of known vulnerabilities or weaknesses in the contract, or could be the result of external factors such as attacks or exploits targeting similar contracts.

Medium Likelihood: Risks that have a medium likelihood are those that are possible but not as likely to occur as those in the high likelihood category. These risks could be the result of less severe vulnerabilities or weaknesses in the contract, or could be the result of less targeted attacks or exploits.

Low Likelihood: Risks that have a low likelihood are those that are unlikely to occur, but still possible. These risks could be the result of very specific or complex vulnerabilities or weaknesses in the contract, or could be the result of highly targeted attacks or exploits.

Informational

Informational issues are mostly connected to violations of best practices, typos in code, violations of code style, and dead or redundant code.

Informational issues are not affecting the score, but addressing them will be beneficial for the project.

Appendix 2. 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/release/v2-upgrade-full/ethereum
Commit	b7bc57e40497b44e694b75b76603f046de7bbaa
Whitepaper	https://github.com/identity-com/gateway-whitepaper/blob/main/gateway-whitepaper.pdf
Requirements	http://docs.civic.com/
Technical Requirements	https://github.com/identity-com/gateway-whitepaper
Contracts	File: smart-contract/contracts/ChargeHandler.sol SHA3:fc5191aaee8f451bed3123cb3b1f48011ac0d1fe3f1318d4888f6c5eb019fd70 File: smart-contract/contracts/FlagsStorage.sol SHA3 72443df34280f7ff54ec0f53d2ee59bb8a756f02c1e77f28983d367112555e2a File: smart-contract/contracts/FlexibleNonceForwarder.sol SHA3:05ba68d8b40b49f7d2831fb3e72c59598a9d803429cea263d5c1229efae5562a File: smart-contract/contracts/Gated.sol SHA3: 57d8a35a37069411b863330eff0bbcbd311967e0765102c7e57df5293478d1bf File: smart-contract/contracts/GatewayToken.sol SHA: a03a4be310c32a5346e6b5b4ea836fe8746947b58bf694aa016b45a42dd62452 File: smart-contract/contracts/MultiERC2771Context.sol SHA3: 751901e7c58155786521983efc20f81120ca968a5993b0d1082815fc264bd945 File: smart-contract/contracts/ParameterizedAccessControl.sol SHA3: dd9f82270f0e479e9c4cb5a183603546dfdc5d862a31644d116b5d289a0cbca5 File: smart-contract/contracts/TokenBitMask.sol SHA3: 832196e4631f0edb903ed0a3551b695e04b2fb83f5f96727fd7c4f5c7163b010 File: smart-contract/contracts/interfaces/IChargeHandler.sol SHA3: c1bd92f1b0b5c3c6277056633126ec1e0c7b80389c05fdb436f32291914d541b File: smart-contract/contracts/interfaces/IERC721Expirable.sol SHA3: 0e1345c59c398c72d195d3e6c1b29a0524a0673689ab643f77209cf7fc2e99bf FIle: smart-contract/contracts/interfaces/IERC721Freezable.sol SHA3: 29a6884240951985868d6fde73ffe0b721612e8b2ac1bd63b77b88ac599b0107 File: smart-contract/contracts/interfaces/IERC721Revokable.sol SHA3: a69af47b12c2885793bbe66628b4831f0149e29eee399470462e2909cc385c7d File: smart-contract/contracts/interfaces/IFlagsStorage.sol SHA3: 482405b2b3daf4025fa89b0505c43000600c1c4c755fc7e164873b1d16325be1 FIle: smart-contract/contracts/interfaces/IForwarder.sol SHA3: 42388bd2df4cd8659cc9af88afce02dc1a86119319553ed35f80821c38d98cc4

	File: smart-contract/contracts/interfaces/IGatewayToken.sol SHA3: 199d9e156d2e0c0eb6dbe48155c9d804bfd0b08b8e91b02a97f0a50e65eb0d6a File: smart-contract/contracts/interfaces/IGatewayTokenVerifier.sol SHA3: cf713f871fe109af1c39aab965f516ce42aebf4afb9db15fa528b041a1cff342 FIle: smart-contract/contracts/interfaces/IParameterizedAccessControl.sol SHA3: 22a068aabe3632eef2f5d1d47b13a0963eaaca4780ea3dc653eb0e18713c62a6 File: smart-contract/contracts/library/BitMask.sol SHA3: 82f41bc52ab3025e394ba70c01ee5dc3a314d359ba950f30f2a58f7b714bc8ba FIle: smart-contract/contracts/library/Charge.sol SHA3: 232f513aa1e18d430524c98738d6f21b0e11dfcf41121faccca93557bf5885f9 File: smart-contract/contracts/library/CommonErrors.sol SHA3: 250d5785d419029be741c8c88f38d279c9f42b056b7c1e1ee68da55f6e97a666 File: smart-contract/contracts/library/InternalTokenApproval.sol SHA3: b38255ad91e5d67f2f1dae6144db36d2063cadd845ea3d11709365e72fc88f22

Second review scope

Repository	https://github.com/identity-com/on-chain-identity-gateway/pull/468/commits/d62d943e7cf8d84c4dc2371a072e0d7e594a6805
Commit	d62d943e7cf8d84c4dc2371a072e0d7e594a6805
Whitepaper	https://github.com/identity-com/gateway-whitepaper/blob/main/gateway-whitepaper.pdf
Requirements	http://docs.civic.com/
Technical Requirements	https://github.com/identity-com/gateway-whitepaper
Contracts	File: ChargeHandler.sol SHA3: 479524f6868e113a23ec3e56cbfc35554f1612467988a78ed1f6ccbd6ff1310f File: FlagsStorage.sol SHA3: 112df6a6a47da3614cb1ba4f526fb2147681934b3c52114e0ec9d2ae370bf725 File: FlexibleNonceForwarder.sol SHA3: 880c8c00496dfb9ce84df803bd139be141c3a50c6f3c4f303d00912f16e21d09 File: Gated.sol SHA3: b03217e8bbdb097603b9c00e9dd26c6ab1d61b522c92af57511bc93d404dfd40 File: GatewayToken.sol SHA3: edca63743ac1c74d98597638bafcae0eb9710a810746da324175e6c25c434078 File: MultiERC2771Context.sol SHA3: c7e4c7c2935e6ed402a988424b473c7d0f183446217874168fca0e23dcc9ed8f File: ParameterizedAccessControl.sol SHA3: d75d9c004a53a5575500a7ca3387281a0a2944c215718e28e48c1f8e428d1cbb File: TokenBitMask.sol

	SHA3: 2b7e7ef26f7465913e20baf144a8803fa6369debeecdadd374d6c02d95ee755d File: interfaces/IChargeHandler.sol SHA3: c1bd92f1b0b5c3c6277056633126ec1e0c7b80389c05fdb436f32291914d541b File: interfaces/IERC721Expirable.sol SHA3: 0e1345c59c398c72d195d3e6c1b29a0524a0673689ab643f77209cf7fc2e99bf File: interfaces/IERC721Freezable.sol SHA3: 29a6884240951985868d6fde73ffe0b721612e8b2ac1bd63b77b88ac599b0107 File: interfaces/IERC721Revokable.sol SHA3: a69af47b12c2885793bbe66628b4831f0149e29eee399470462e2909cc385c7d File: interfaces/IFlagsStorage.sol SHA3: 482405b2b3daf4025fa89b0505c43000600c1c4c755fc7e164873b1d16325be1 File: interfaces/IForwarder.sol SHA3: 42388bd2df4cd8659cc9af88afce02dc1a86119319553ed35f80821c38d98cc4 File: interfaces/IGatewayToken.sol SHA3: ac38e219b225782b10904c61c677b1434a4dad79ba24ed187dfb1f47ef9243f9 File: interfaces/IGatewayTokenVerifier.sol SHA3: cf713f871fe109af1c39aab965f516ce42aebf4afb9db15fa528b041a1cff342 File: interfaces/IParameterizedAccessControl.sol SHA3: c4cba2e2c00dabc7b17eabcaad7e24658c91e78ab349dcf405be63e07736aed0 File: library/BitMask.sol SHA3: 82f41bc52ab3025e394ba70c01ee5dc3a314d359ba950f30f2a58f7b714bc8ba File: library/Charge.sol SHA3: 232f513aa1e18d430524c98738d6f21b0e11dfcf41121faccca93557bf5885f9 File: library/CommonErrors.sol SHA3: 250d5785d419029be741c8c88f38d279c9f42b056b7c1e1ee68da55f6e97a666 File: library/InternalTokenApproval.sol SHA3: cf7c54f5629f4210a66438380c4aeef82669876cc0296dc773ff6e769a6a4039