Making Web3 Space Safer for Everyone

KALOS

Intella X-gasless V1.0

Security Assessment

Published on : 1 Sep. 2023
Version v1.0

Security Report Published by KALOS

v1.0 1 Sep. 2023

Auditor : Jade Han

hojung han

Found issues

Severity of Issues	Findings	Resolved	Acknowledged	Comment
Critical	-	-	-	-
High	-	-	-	-
Medium	-	-	-	-
Low	-	-	-	-
Tips	1	-	1	-

TABLE OF CONTENTS

ABOUT US

Making Web3 Space Safer for Everyone

KALOS is a flagship service of HAECHI LABS, the leader of the global blockchain industry. We bring together the best Web2 and Web3 experts. Security Researchers with expertise in cryptography, leaders of the global best hacker team, and blockchain/smart contract experts are responsible for securing your Web3 service.

Having secured $60B crypto assets on over 400 main-nets, Defi protocols, NFT services, P2E, and Bridges, KALOS is the only blockchain technology company selected for the Samsung Electronics Startup Incubation Program in recognition of our expertise. We have also received technology grants from the Ethereum Foundation and Ethereum Community Fund.

Inquiries: audit@kalos.xyz
Website: https://kalos.xyz

Executive Summary

Purpose of this report

This report was prepared to audit the security of the project developed by the Intella X team. KALOS conducted the audit focusing on whether the system created by the Intella X team is soundly implemented and designed as specified in the published materials, in addition to the safety and security of the project.

In detail, we have focused on the following

- Denial of Service
- Access Control of Various Storage Variables
- Access Control of Important Functions
- Freezing of User Assets
- Theft of User Assets
- Unhandled Exceptions

Codebase Submitted for the Audit

The codes used in this Audit can be found on GitHub (https://github.com/intella-x/intellax-gasless).

The commit hash of the code used for this Audit is "e48b9ef28afa0b569b7c907e76a52c5579511d91",

Audit Timeline

Date	Event
2023/08/07	Audit Initiation
2023/09/01	Delivery of v1.0 report.

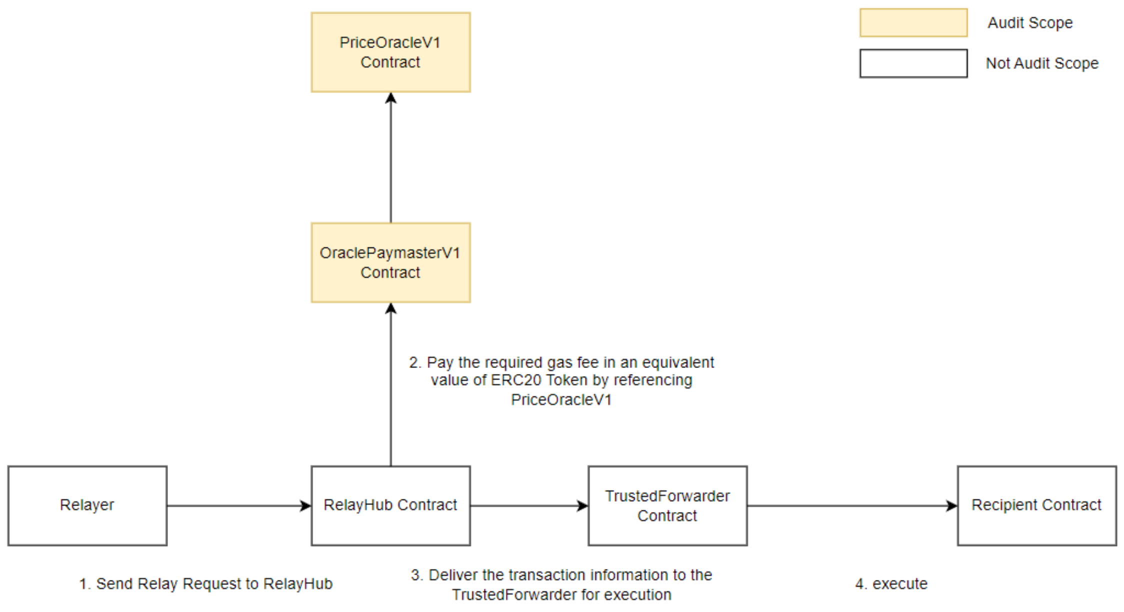
Findings

KALOS found - 0 Critical, 0 High, 0 medium, 0 Low and 1 Tip severity issues.

Severity	Issue	Status
Tip	Potential Vulnerability in Forwarder Contract's execute Function	(Acknowledged - v1.0)

OVERVIEW

Contract Overview


PriceOracleV1 Contract
OraclePaymasterV1 Contract
Audit Scope
Not Audit Scope
2. Pay the required gas fee in an equivalent value of ERC20 Token by referencing PriceOracleV1
Relayer
RelayHub Contract
TrustedForwarder Contract
Recipient Contract
1. Send Relay Request to RelayHub
3. Deliver the transaction information to the TrustedForwarder for execution
4. execute

OraclePaymasterV1

The contract addresses compensating Relayers for the gas fees they've expended and enables the transaction initiator (constructor) to provide ERC20 Tokens to the Paymaster. Transaction initiator sends ERC20 Tokens to the Paymaster.

Also, The Paymaster reimburses the Relayer for the gas fees they've used as Native Tokens. This approach allows the Paymaster to facilitate gas fee payments in ERC20 Tokens for users who possess ERC20 Tokens but lack Native Tokens.

PriceOracleV1

This contract acts as a trusted source of token price information for the blockchain ecosystem. Authorized updaters can submit accurate token price updates stored securely in the contract.

OraclePaymasterV1 contract can retrieve these prices, ensuring reliable and meaningful data for gas fee calculations and ERC20 Token payments. The contract also integrates with the FeeWorkerV1 mechanism to maintain operational efficiency by recharging gas fees for authorized updaters.

Scope

```
├── OraclePaymasterV1.sol
├── PriceOracleV1.sol
├── interfaces
│   ├── IFeeManager.sol
│   ├── IPriceOracle.sol
│   └── IWETH.sol
├── rechargable
│   └── FeeWorkerV1.sol
├── registrable
│   └── AddressRegistry.sol
```

Access Controls

Access control in contracts is achieved using modifiers, and inline require statements. The access control of the audit scope refers to the following actors.

- ❖ RelayHub
- ❖ Owner
- ❖ Updater

RelayHub : In the OpenGSN Architecture, some functions should only be invoked on the RelayHub contract for security reasons.

- OraclePaymasterV1.sol#preRelayedCall
- OraclePaymasterV1.sol#postRelayedCall

Owner : It is an entity that sets variables with a significant impact on the contract's execution flow.

- OraclePaymasterV1.sol#setRelayHub
- OraclePaymasterV1.sol#setTrustedForwarder
- OraclePaymasterV1.sol#setWeth
- OraclePaymasterV1.sol#setPriceOracle
- OraclePaymasterV1.sol#setFeeCollector
- OraclePaymasterV1.sol#setFeePayer
- OraclePaymasterV1.sol#setGasUsedByPost
- OraclePaymasterV1.sol#setGasAndDataLimits
- OraclePaymasterV1.sol#withdrawRelayHubDepositTo
- PriceOracleV1.sol#setFeeManager
- PriceOracleV1.sol#registerAddress
- PriceOracleV1.sol#unregisterAddress

Updater : Only addresses authorized as updaters can update prices within PriceOracleV1.

- PriceOracleV1.sol#update

FINDINGS

1. Potential Vulnerability in Forwarder Contract's execute Function

ID: IntellaX-gasless-1	Severity: Tips
Type: DoS	Difficulty: -

Issue

Intella X Team said plans to support a range of Gasless Transactions, including DEX swaps, DEX liquidity provides, NFT launchpads, and NFT marketplaces via OpenGSN. They said they will utilize the existing Forwarder Contract deployed on the Polygon Mainnet by OpenGSN. Upon analysis of the execute function of the Forwarder Contract, which runs through the RelayHub Contract, a potentially vulnerable scenario was identified.

The execute function is open to everyone, making it prone to unwanted public invocations, and doesn't revert if the forward transaction fails. This behavior presents a significant risk.

Given the nature of the Polygon Network, it is susceptible to front-running. Malicious actors can catch pending transactions that call the relayCall function in the mempool, extract the ForwardRequest, and run this part in advance.

Assuming the ForwardRequest is used to swap owned assets to a specific asset via QuickSwap (a fork of UniswapV2), front-runners could trigger the flashloan handler in the QuickSwap pair and then run the ForwardRequest through the forwarder. Due to the reentrancy guard in the QuickSwap pair, this transaction would fail, and the nonce would increase. Consequently, the original forward request would face a nonce mismatch, causing its call to fall.

While this scenario may not immediately impact the products of the Intella X Team, we highlight the potential that someone could deliberately manipulate the ForwardRequest to fail, preventing certain transactions from being executed.

Recommendation

To prevent possible exploits, we strongly recommend restricting access to the execute function, ensuring that only the RelayHub can call it. Consider redeploying the Forwarder Contract with this modification for enhanced security.

DISCLAIMER

This report does not guarantee investment advice, the suitability of the business models, and codes that are secure without bugs. This report shall only be used to discuss known technical issues. Other than the issues described in this report, undiscovered issues may exist such as defects on the main network. In order to write secure codes, correction of discovered problems and sufficient testing thereof are required.

Appendix. A

Severity Level

CRITICAL	Must be addressed as a vulnerability that has the potential to seize or freeze substantial sums of money.
HIGH	Has to be fixed since it has the potential to deny users compensation or momentarily freeze assets.
MEDIUM	Vulnerabilities that could halt services, such as DoS and Out-of-Gas, need to be addressed.
LOW	Issues that do not comply with standards or return incorrect values
TIPS	Tips that makes the code more usable or efficient when modified

Difficulty Level

	Low	**Medium**	**High**
Privilege	anyone	Miner/Block Proposer	Admin/Owner
Capital needed	Small or none	Gas fee or volatile as price change	More than exploited amount
Probability	100%	Depend on environment	Hard as mining difficulty

Vulnerability Category

Arithmetic	• Integer under/overflow vulnerability • floating point and rounding accuracy
Access & Privilege Control	• Manager functions for emergency handle • Crucial function and data access • Count of calling important task, contract state change, intentional task delay
Denial of Service	• Unexpected revert handling • Gas limit excess due to unpredictable implementation
Miner Manipulation	• Dependency on the block number or timestamp. • Frontrunning
Reentrancy	•Proper use of Check-Effect-Interact pattern. •Prevention of state change after external call • Error handling and logging.
Low-level Call	• Code injection using delegatecall • Inappropriate use of assembly code
Off-standard	• Deviate from standards that can be an obstacle of interoperability.
Input Validation	• Lack of validation on inputs.
Logic Error/Bug	• Unintended execution leads to error.
Documentation	•Coherency between the documented spec and implementation
Visibility	• Variable and function visibility setting
Incorrect Interface	• Contract interface is properly implemented on code.

End of Document