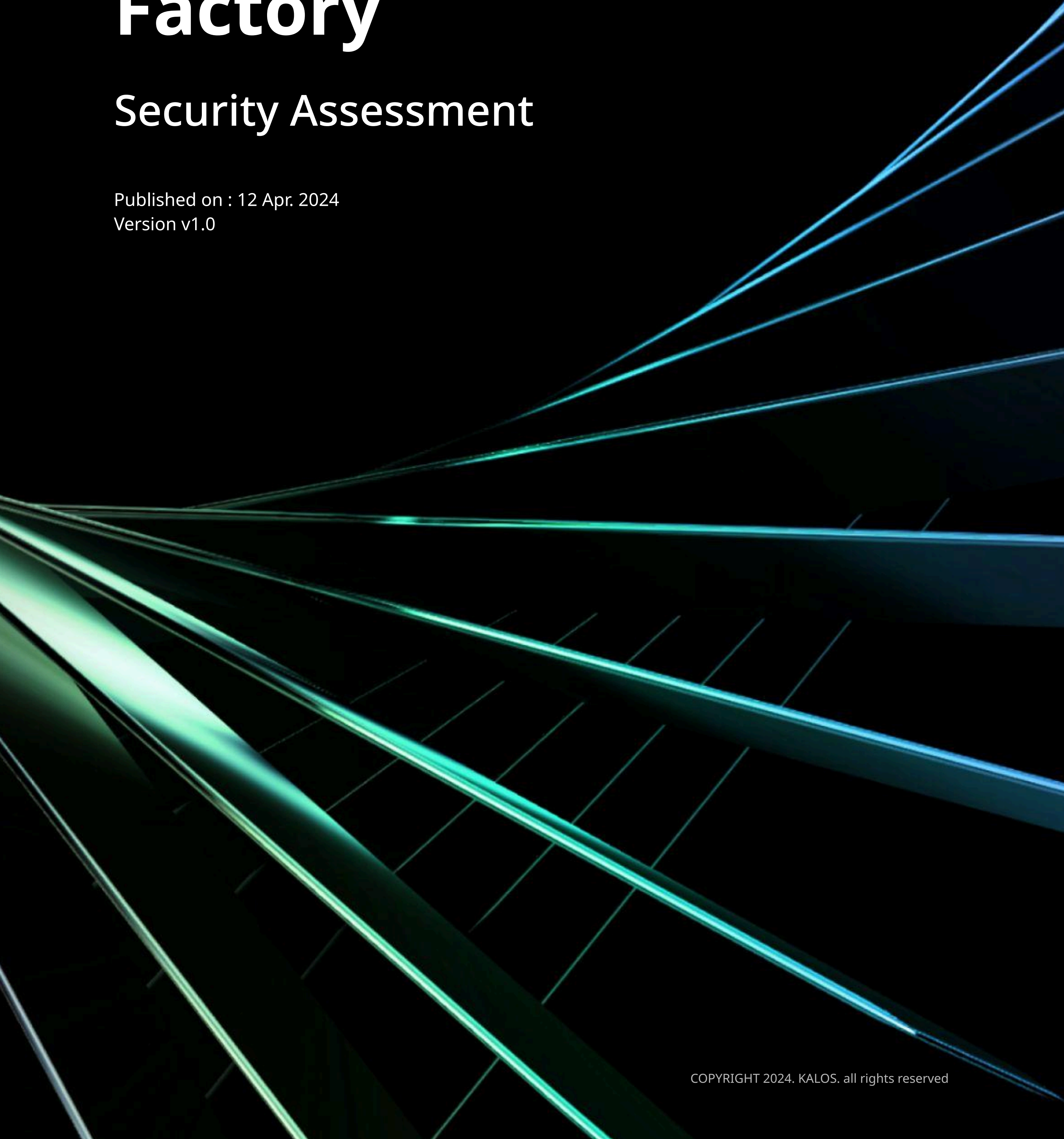
Making Web3 Space Safer for Everyone

KALOS

ZeroDev Kernel V3 Factory

Security Assessment

Published on : 12 Apr. 2024
Version v1.0



Security Report Published by KALOS

v1.0 12 Apr. 2024

Auditor : Jade Han

hojung han

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

TABLE OF CONTENTS

TABLE OF CONTENTS

ABOUT US

Executive Summary

OVERVIEW

- ERC-7579 Plugins on Kernel V3
- Scope

FINDINGS

DISCLAIMER

Appendix. A

- Severity Level
- Difficulty Level
- Vulnerability Category

ABOUT US

Making Web3 Space Safer for Everyone

Pioneering a safer Web3 space since 2018, KALOS proudly won 2nd place in the Paradigm CTF 2023. As a leader in the global blockchain industry, we unite the finest in Web3 security expertise.

Our team consists of top security researchers with expertise in blockchain/smart contracts and experience in bounty hunting. Specializing in the audit of mainnets, DeFi protocols, bridges, and the zkEVM protocol, KALOS has successfully safeguarded billions in crypto assets.

Supported by grants from the Ethereum Foundation and the Community Fund, we are dedicated to innovating and enhancing Web3 security, ensuring that our clients' digital assets are securely protected in the highly volatile and ever-evolving Web3 landscape.

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 ZeroDev team. KALOS conducted the audit focusing on whether the system created by the ZeroDev 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
- Compatibility Testing with Bundler

Codebase Submitted for the Audit (v1.0)

The codes used in this Audit can be found on GitHub (https://github.com/zerodevapp/kernel_v3/).

The commit hash of the code used for this Audit is “85617952ec4cdf8d6478a9139540968fb92a6f10”

Audit Timeline

Date	Event
2024/04/11	Audit Initiation
2024/04/12	Delivery of v1.0 report.

OVERVIEW

Kernel V3 Factory

FactoryStaker

The `FactoryStaker` contract is engineered to manage multiple `KernelFactory` instances, enhancing administrative efficiency and control over factory approvals. Leveraging the `Ownable` contract from "`solady/auth`", it restricts critical actions to the owner of the contract. The `approveFactory` method allows the contract to approve or disapprove factories, controlling which ones can be utilized to deploy accounts. The `deployWithFactory` method permits the deployment of accounts via approved factories only, reinforcing security protocols. This contract also manages staking for ERC-4337 entry points through capabilities to add, unlock, and withdraw stakes.

KernelFactory

The `KernelFactory` contract is designed to create ERC-4337 accounts using a deterministic deployment method, facilitated by the `LibClone` library from "`solady/utils`". This contract holds an immutable reference to an implementation address, which is specified during the contract's initialization. The `createAccount` method of this contract generates ERC-4337 compliant accounts by computing a unique salt from input data and another provided salt. It then uses this composite to deploy a clone of the implementation. If the account was not previously deployed, initialization code can be executed. Additionally, the `getAddress` method allows for the prediction of the address for a potential account before its actual creation, providing foresight into where the account will reside.

Scope

```
├── FactoryStaker.sol
└── KernelFactory.sol
```

*** We have verified whether the codes within the Scope are sufficiently compatible with the 4337 Specification using the Bundler (https://github.com/pimlicolabs/alto)**

FINDINGS

No security issues found

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