Audit Report

Vectis

v1.0

March 14, 2024

Table of Contents

License

Disclaimer

This audit has been performed by

Oak Security GmbH

https://oaksecurity.io/
info@oaksecurity.io

Introduction

Purpose of This Report

Oak Security has been engaged by the Neutron Audit Sponsorship Program to perform a security audit of Vectis.

The objectives of the audit are as follows:

1. Determine the correct functioning of the protocol, in accordance with the project specification.
2. Determine possible vulnerabilities, which could be exploited by an attacker.
3. Determine smart contract bugs, which might lead to unexpected behavior.
4. Analyze whether best practices have been applied during development.
5. Make recommendations to improve code safety and readability.

This report represents a summary of the findings.

As with any code audit, there is a limit to which vulnerabilities can be found, and unexpected execution paths may still be possible. The author of this report does not guarantee complete coverage (see disclaimer).

Codebase Submitted for the Audit

The audit has been performed on the following targets:

Repository	https://github.com/nymlab/vectis
Commit	`1902fe863e786ac87d17f69f8ac3a0da1f54871e`
Scope	All contracts were in scope.

Repository	https://github.com/nymlab/vectis-contracts
Commit	`47e6a9113ee284264db0ca623e006806b3f35afb`
Scope	All contracts were in scope.

Repository	https://github.com/nymlab/vectis-authenticators
Commit	`0723b05601ec36309880b6c2a4beaf80c6367a88`
Scope	All contracts were in scope.

Methodology

The audit has been performed in the following steps:

1. Gaining an understanding of the code base's intended purpose by reading the available documentation.
2. Automated source code and dependency analysis.
3. Manual line-by-line analysis of the source code for security vulnerabilities and use of best practice guidelines, including but not limited to:
    a. Race condition analysis
    b. Under-/overflow issues
    c. Key management vulnerabilities
4. Report preparation

Functionality Overview

Vectis is a protocol that empowers users to generate wallets accessible via various OAUTH authenticators. These wallets operate as smart contracts and facilitate seamless integration with plugins and interchain accounts.

How to Read This Report

This report classifies the issues found into the following severity categories:

Severity	Description
Critical	A serious and exploitable vulnerability that can lead to loss of funds, unrecoverable locked funds, or catastrophic denial of service.
Major	A vulnerability or bug that can affect the correct functioning of the system, lead to incorrect states or denial of service.
Minor	A violation of common best practices or incorrect usage of primitives, which may not currently have a major impact on security, but may do so in the future or introduce inefficiencies.
Informational	Comments and recommendations of design decisions or potential optimizations, that are not relevant to security. Their application may improve aspects, such as user experience or readability, but is not strictly necessary. This category may also include opinionated recommendations that the project team might not share.

The status of an issue can be one of the following: **Pending, Acknowledged**, or **Resolved**.

Note that audits are an important step to improving the security of smart contracts and can find many issues. However, auditing complex codebases has its limits and a remaining risk is present (see disclaimer).

Users of the system should exercise caution. In order to help with the evaluation of the remaining risk, we provide a measure of the following key indicators: **code complexity**, **code readability**, **level of documentation**, and **test coverage**. We include a table with these criteria below.

Note that high complexity or low test coverage does not necessarily equate to a higher risk, although certain bugs are more easily detected in unit testing than in a security audit and vice versa.

Code Quality Criteria

The auditor team assesses the codebase's code quality criteria as follows:

Criteria	Status	Comment
Code complexity	**Medium**	-
Code readability and clarity	**Medium-High**	-
Level of documentation	**Low-Medium**	The client provided high-level documentation in README files.
Test coverage	**Low**	Some tests are failing. `cargo tarpaulin` reports a 31.02% code coverage.

Summary of Findings

No	Description	Severity	Status
1	Wallet fees can be circumvented	Major	Resolved
2	Full-featured wallet creation may run out of gas	Major	Acknowledged
3	Malfunctioning or malicious plugins could make the `proxy` contract unusable	Major	Resolved
4	It is not possible to downgrade the subscription to the `Free` tier	Major	Resolved
5	Missing deployer role segregation leads to centralization	Minor	Acknowledged
6	Wallet `nonce` can be reset	Minor	Resolved
7	Code IDs unregistered before whitelisting could allow attackers to instantiate whitelisted malicious contracts	Minor	Resolved
8	Providing multiple instances of the same `AuthenticatorInstInfo` type does not raise an error	Minor	Resolved
9	The `ensure_enough_native_funds` function fails if coins are not provided in the expected order	Minor	Resolved
10	It is possible to create wallets with not whitelisted proxies	Minor	Resolved
11	Missing address validation	Minor	Resolved
12	The update of the deployer address in the `factory` contract could disrupt the operations of the `proxy` contract	Minor	Acknowledged
13	It is possible to overwrite existing registered plugin contracts	Minor	Resolved
14	Subscription tiers are not updatable	Minor	Resolved
15	Missing `pagination` for the plugin query	Minor	Resolved
16	Missing subscriber contract validation	Minor	Resolved
17	Lack of input vector deduplication	Informational	Resolved

18	Contracts should implement a two step ownership transfer	Informational	Acknowledged
19	Missing input validation for wallet fees	Informational	Acknowledged
20	Usage of generic events in `update_code_id`	Informational	Resolved
21	Typographical error	Informational	Resolved

Detailed Findings

1. Wallet fees can be circumvented

Severity: Major

The function create_wallet calls ensure_enough_native_funds in contracts/core/factory/src/service/create.rs:26 by passing the user-provided proxy_initial_funds as an argument.

Subsequently, the ensure_enough_native_funds function iterates through these funds and adds the wallet fee to the entry in proxy_initial_funds which has the same denom as the wallet fee.

However, the code does not check that proxy_initial_funds contains such an entry. If it does not, the wallet fee is not added to the vector required_funds and no check is in place whether the user has sent the expected funds.

This logic allows users to bypass the wallet fee by defining a vector of proxy_initial_funds that does not contain an entry with the fee denom.

Recommendation

We recommend checking if the denom of the fee is contained in proxy_initial_funds. If not, an additional entry with the fee amount should be appended to the vector.

Status: Resolved

2. Full-featured wallet creation may run out of gas

Severity: Major

The create_wallet function defined in factory/src/service/contract.rs:22-30 within the factory contract has the potential to execute a substantial number of operations in case of full-featured walled creation, posing a risk of running out of gas during execution.

The following steps are executed within the create_wallet function:

1. Two messages are dispatched: One for instantiating the proxy contract and another one for sending the fee to the deployer.
2. The instantiate function of the proxy contract stores each element from the initial_data vector.
3. For each element in the relayers vector, a MsgGrantAllowance message is created and dispatched.

4. For each element in `required_chains vector`, an ICA account is created by dispatching a `MsgRegisterInterchainAccount` message.
5. For each element in `plugins`, a message is executed to the plugin registry contract and the plugin contract is created and instantiated.

As a result, if there are a sufficient number of elements in the mentioned vectors, the execution could potentially run out of gas not allowing users to create the wallet.

Recommendation

We recommend breaking down wallet creation into multiple transactions to allow users to create a basic wallet and then perform other transactions to add features.

Status: Acknowledged

3. Malfunctioning or malicious plugins could make the `proxy` contract unusable

Severity: Major

In `proxy/src/wallet/contract.rs:104-132`, the `auth_exec` function enables the user to execute transactions signed with the `authenticator`.

However, since the `auth_exec` function executes `pre_tx_check` and `exec_with_post_tx_hooks`, if a plugin is broken, intentionally fails or the plugin execution runs out of gas, the transaction will be reverted and no further operations will be allowed.

Additionally, since the only way to remove a plugin is by using the `auth_exec` function, it would not be possible for a user to remove it and recover from this issue.

Consequently, the `proxy` contract would become unusable and the funds within it would be stuck.

Recommendation

We recommend implementing a separate function to remove the plugins.

Status: Resolved

The client implemented a new function called `auth_exec_without_plugins` that enables users to execute transactions without executing plugins. While this resolves the issue, it could pose a potential risk in case some plugins should be mandatorily executed like Two-Factor Authentication (2FA) ones.

The client indicates that the `auth_exec_without_plugin` function is managed by the controller. In scenarios involving 2FA, the controller is equipped with an authenticator capable of supporting 2FA or One-Time Passwords (OTP), among other methods. It is important to note that these authenticators are distinct from the typical plugins discussed.

The client will exclusively support plugins from Vectis as the provider facilitated through a registry system. While users have the option to install their own plugins, such actions fall outside the scope of official support.

Additionally, the `auth_exec` message is primarily utilized by the wallet web application, suggesting that alternative messages for user signing are considered only under specific circumstances where users need to circumvent standard protocols.

4. It is not possible to downgrade the subscription to the `Free` tier

Severity: Major

In `plugin_registry/src/service.rs:120-165`, the `subscribe` feature in the `plugin_registry` contract allows `proxy` contracts to subscribe to the desired tier.

However, since in line `128` the execution requires at least one coin in `info.funds` and the `Free` tier has no fees, this process generates an error and consequently the revertion of the transaction not allowing users to subscribe to the `Free` tier or any other tier with zero fees.

Recommendation

We recommend allowing transactions without funds for the subscription to the `Free` tier.

Status: Resolved

5. Missing deployer role segregation leads to centralization

Severity: Minor

All the contracts in scope implement the role of the `deployer` and initialize it as the address that executes their `instantiate` function.

This account has the following roles:

- Contract administrator: it can execute all the critical messages used to configure the contract
- Fee collector: All the protocol fees are forwarded to this account
- Addresses registry: Contracts query data from this address to get the addresses of other protocol contracts such as the `factory` or the `plugin_registry`.

Consequently, no segregation of roles is enforced leading to the `deployer` being a single point of failure for the entire protocol.

Additionally, since the contract designed to be deployed to fulfill the `deployer` role is not in the audit scope no additional verification has been possible.

Recommendation

We recommend enforcing role segregation.

Status: Acknowledged

6. Wallet `nonce` can be reset

Severity: Minor

In `proxy/src/wallet/exec.rs:62-77`, the `rotation` function allows the user to substitute the `controller`.

A check is in place to not allow the user to substitute a controller with the same one in order to reset the nonce.

However, by using a third `controller`, the user can perform a double `rotation` and still reset the `nonce`, allowing message replay.

We classify this issue as minor since only the user itself can perform this operation to reset their nonce.

Recommendation

We recommend storing historical data about the `controller` and disallowing setting a previous controller to prevent a `nonce` reset.

Status: Resolved

7. Code IDs unregistered before whitelisting could allow attackers to instantiate whitelisted malicious contracts

Severity: Minor

In `factory/src/contract.rs:80-81` and `factory/src/management/contract.rs:47-53`, before storing the provided proxy contract code ID, the execution does not check whether it has already been registered.

Consequently, an attacker could attempt to upload a malicious service, included in the whitelist, using the aforementioned code ID.

We classify this issue with minor severity because this misconfiguration is only possible from the admin during the instantiation.

Recommendation

We recommend performing a query for each `code_id` specified to ensure they are already registered in `wasmd`.

Status: Resolved

8. Providing multiple instances of the same `AuthenticatorInstInfo` type does not raise an error

Severity: Minor

In `factory/src/contract.rs:100-103`, if the user supplies multiple instances of the same `AuthenticatorInstInfo` type during the execution of the `instantiate` function, the process does not produce an error, but it proceeds with a `continue` instruction.

As a result, the execution will store only the first `AuthenticatorInstInfo` encountered and no error will be raised.

It is best practice to inform the user in case the provided input is ignored or an error occurs.

Recommendation

We recommend returning an error in case multiple `AuthenticatorInstInfo` are provided.

Status: Resolved

9. The `ensure_enough_native_funds` function fails if coins are not provided in the expected order

Severity: Minor

In `factory/src/service/create.rs:105-137`, the `ensure_enough_native_funds` function verifies that the funds sent by the user cover both the `fee` and the `proxy_initial_fund`.

However, if the order of the coins in the `sent_fund` and `proxy_initial_fund` vectors is not the same, the function will fail since there is a check that uses the equality operator.

Recommendation

We recommend using the `Coins` type along with its associated methods.

Status: Resolved

10. It is possible to create wallets with not whitelisted proxies

Severity: Minor

In `factory/src/service/create.rs:35-37`, the `proxy_code_id` provided by the user in `create_wallet` should be validated to exist within the whitelist of `supported_proxies`.

Failure to enforce this check could allow the user to designate any contract as a proxy, including potentially malicious ones.

We classify this issue as minor since this misconfiguration is only possible from the admin.

Recommendation

We recommend verifying that the provided `proxy_code_id` is included in the whitelist.

Status: Resolved

11. Missing address validation

Severity: Minor

In `proxy/src/contract.rs:111`, during the execution of the `instantiate` function of the `proxy` contract, the `controller` address is stored in the contract.

However, since it can contain an address in the case of `AuthenticatorProvider::Custom`, it should be validated before being stored.

Similarly, in `plugin_registry/src/management/plugin_mgmt.rs:60`, the `metadata_data.creator` address should be validated before instantiating the `Plugin` struct.

Recommendation

We recommend validating addresses before storing them in the contract.

Status: Resolved

12. The update of the `deployer` address in the `factory` contract could disrupt the operations of the `proxy` contract

Severity: Minor

In `proxy/src/contract.rs:115-119`, during the `proxy` contract instantiation, the `deployer` address is fetched from the `factory` contract and stored.

However, since the `deployer` address can be updated in the `factory` contract, its update will not be reflected in any previously instantiated `proxy` that will continue querying the old one.

Consequently, it is advisable to not fetch the `deployer` address during the execution of the `proxy` contract's `instantiate` function, as any changes will not be reflected here.

Recommendation

We recommend querying the `deployer` address to the `factory` contract every time the `proxy` needs it.

Status: Acknowledged

13. It is possible to overwrite existing registered plugin contracts

Severity: Minor

In `plugin_registry/src/management/plugin_mgmt.rs:68-70`, the `register` function does not ensure that the provided plugin `code_id` is already registered in the `plugin_code_ids` map.

Similarly, in `plugin_registry/src/management/plugin_mgmt.rs:160-161` the `new_plugin_version` function allows the `deployer` to store existing `code_ids`.

Consequently, it is possible to overwrite an existing plugin and assign to it a new `code_id`.

We classify this issue as minor since it can be caused only by the `reviewer`, which is a privileged account.

Recommendation

We recommend verifying whether the plugin is already registered before storing a new one.

Status: Resolved

14. Subscription tiers are not updatable

Severity: Minor

In plugin_registry/src/management/config.rs:5-22, the add_subscription_tiers function does not allow the modification of existing subscription tiers.

Attempting to update an existing tier in the add_subscription_tiers function will result in an error in line 12.

Consequently, it is not possible for the deployer to modify elements such as the subscription fee.

Recommendation

We recommend enabling the deployer to update subscription tiers.

Status: Resolved

15. Missing pagination for the plugin query

Severity: Minor

In proxy/src/plugin/contract.rs, the plugins query returns a combined list of exec_plugins, pre_tx_plugins, and post_tx_hooks_plugins.

However, since there is no pagination in place, performance issues might occur as the number of plugins grows, up to the point where the query could run out of gas.

Recommendation

We recommend enhancing the efficiency of the plugins query by implementing pagination.

Status: Resolved

16. Missing subscriber contract validation

Severity: Minor

In plugin_registry/src/service.rs:120-166, the subscribe function in the plugin_registry contract enables proxy contracts to subscribe to the requested tier.

However, since there is no check in place to confirm that the sender is a proxy contract, any address can subscribe and pay the subscription fee.

Recommendation

We recommend permitting only `proxy` contracts to subscribe.

Status: Resolved

17. Lack of input vector deduplication

Severity: Informational

In the following locations, vectors provided as input to the contract are not deduplicated:

- In `factory/src/contract.rs:82-86`, during the execution of the `instantiate` function, the `supported_proxies` vector lacks deduplication. Additionally, the vector should include, at a minimum, the default one.
- In `factory/src/contract.rs:89-95`, the `supported_chains` vector is not deduplicated.
- In `plugin_registry/src/contract.rs:87-91`, the `subscription_tiers` vector lacks deduplication.
- In `factory/src/service/create.rs:105-137`, the `proxy_initial_fund` vector is not deduplicated.
- In `proxy/src/contract.rs:129`, the `relayers` vector lacks deduplication allowing the execution of duplicated `MsgGrantAllowance` messages.
- In `proxy/src/contract.rs:158-182`, the `required_chains` vector lacks deduplication allowing the execution of duplicated `MsgRegisterInterchainAccount` messages.

Consequently, the contract could store duplicated elements or in the case of mappings only the last duplicated element of the vector.

Recommendation

We recommend deduplicating vectors before storing them in the contract.

Status: Resolved

18. Contracts should implement a two-step ownership transfer

Severity: Informational

The contracts within the scope of this audit allow the current owner to execute a one-step ownership transfer. While this is common practice, it presents a risk for the ownership of the contract to become lost if the owner transfers ownership to the incorrect address. A two-step ownership transfer will allow the current owner to propose a new owner, and then the account that is proposed as the new owner may call a function that will allow them to claim ownership and actually execute the config update.

Recommendation

We recommend implementing a two-step ownership transfer. The flow can be as follows:

1. The current owner proposes a new owner address that is validated and lowercased.
2. The new owner account claims ownership, which applies the configuration changes.

Status: Acknowledged

19. Missing input validation for wallet fees

Severity: Informational

When the factory is instantiated in `contracts/core/factory/src/contract.rs:81`, the wallet fee is not validated. The specified `denom` may therefore be invalid.

In such a scenario, creating wallets will not be possible and a redeployment of the factory will be necessary.

Recommendation

We recommend validating wallet fees.

Status: Acknowledged

20. Usage of generic events in `update_code_id`

Severity: Informational

In `factory/src/management/contract.rs:23-63`, the `update_code_id` function permits the caller to perform two operations:

- Removal of a code ID by specifying the `code_id` but not the version.
- Addition of a code ID by specifying both the `code_id` and version, where the `code_id` is not yet registered.

However, the event emitted in lines `57-61` lacks clarity in indicating whether the `code_id` has been added or not, which may mislead off-chain event listeners.

Recommendation

We recommend enhancing the `update_code_id` function event emission to clearly distinguish between `code_id` removal and addition.

Status: Resolved

21. Typographical error

Severity: Informational

In all the `factory/src/contract.rs`, there is a typographical error within the `authenicators` variable.

Recommendation

We recommend renaming the variable to `authenticators`.

Status: Resolved