HACKEN

SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

Customer: Aurora Labs
Date: March 27, 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 Aurora Labs
Approved By	Evgeniy Bezuglyi | SC Audits Department Head at Hacken OU
Type	Plugins
Platform	NEAR Protocol
Language	Rust
Methodology	Link
Website	https://aurora.dev/
Changelog	09.12.2022 - Initial Review 09.03.2023 - Second Review 27.03.2023 - 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 Aurora Labs (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 is review and security analysis of smart contracts in the repository:

Initial review scope

Repository	https://github.com/aurora-is-near/near-plugins/
Commit	7454bbcfcdb50addee5fefe3742879d1fac75c81
Functional Requirements	Link
Technical Requirements	Link
Contracts	File: near-plugins-derive/src/access_control_role.rs SHA3: 7f8d685883c9403f44abe2cbfebbeb17ad1ccfda144fbff9d40937da254813d8 File: near-plugins-derive/src/access_controllable.rs SHA3: 2fa47f8e6e4a380bdc9e0d49a1bad83b8899277d860534c4c5908bc794e18cf0 File: near-plugins-derive/src/full_access_key_fallback.rs SHA3: c57776273ce24a0b8b75118762cdf5c99b156b6bb99d5f66286f689c0d58ade9 File: near-plugins-derive/src/lib.rs SHA3: 85e3f6d32176be69ef72e4d68b1551f3db5630af47a559e6e28c00367966f5eb File: near-plugins-derive/src/ownable.rs SHA3: bc40929ca1d89c904b8720b6b1111be4a2e70a3abdf936835d334282a2c7750d File: near-plugins-derive/src/pausable.rs SHA3: 0a595211322f28123d54b6f4388a19ddb170930b062c902df373f98ba62febba File: near-plugins-derive/src/upgradable.rs SHA3: e7856ba91870944f4da55d00b5f9af5e6ada178b25fe32d83407ca349df832d4 File: near-plugins-derive/src/utils.rs SHA3: 9c1d9b0e4287819baca5b6c4c7620054d71559343f0b16e31a59715c10186b2a File: near-plugins/src/access_control_role.rs SHA3: cabd56ef641ec070494ea1a1160399997ae788116eeefd2efa7e499634321a19 File: near-plugins/src/access_controllable.rs SHA3: ffa84d463212879311eb5f76ed6704f4a4841b927d3d28d692643b083c399190 File: near-plugins/src/events.rs SHA3: 5824b5e2e8e710375467aa5190b5588b54e817bf012428fefd1585fefa4fc4c8 File: near-plugins/src/full_access_key_fallback.rs SHA3: fdf1775a738c8f7f52da6afb8ee81741f2a1513130f4159b716708dd04543ba8 File: near-plugins/src/lib.rs SHA3: 419458e6f74ebb55b35700af2a755805269a0e4302712528647ced4e9f8122a9

	File: near-plugins/src/ownable.rs SHA3: c51686902dfc56cd26399c8c5260c0944768ca4fb820ad629ab2807a04a265ee File: near-plugins/src/pausable.rs SHA3: 2eb1b80c69197996956e54f2a1f773f152ddaf1e81081f1cd254f860a8a7490b File: near-plugins/src/test_utils.rs SHA3: eabbad51060d51d8ff7f97f743624ab483916531d0e9ab375d6134e4ae6d66c6 File: near-plugins/src/upgradable.rs SHA3: 369bee1f5b3dc133edb28a92073973ae67f47aa7f5b928279b9707965b9f8747

Second review scope

Repository	https://github.com/aurora-is-near/near-plugins/
Commit	5545d526987d28b2b9af0dd8e76d18b8c4795a97
Functional Requirements	Link
Technical Requirements	Link
Contracts	File: ./near-plugins-derive/src/access_control_role.rs SHA3: b28cf4087139f80a75ac171b2be7aba9f3387868fc01bc8158bd2771d1981292 File: ./near-plugins-derive/src/access_controllable.rs SHA3: 71b23428175a01641362406aef31cc5cf753c47a88b197d4b5124febbc9262e2 File: ./near-plugins-derive/src/full_access_key_fallback.rs SHA3: 809e8cf130d62ca783d7e1609101e7bd8c389b10e0033d616db14ec80e454c85 File: ./near-plugins-derive/src/lib.rs SHA3: 0665755017dc43eaf2461731c26d17f896547377f1280843056991d2b394d686 File: ./near-plugins-derive/src/ownable.rs SHA3: e71f82ec0f6f215ac407a8b3ff614b049298a5e3c7e0300a8e5e410ac25823ed File: ./near-plugins-derive/src/pausable.rs SHA3: ce2e2bca54677da96ae19007907198e289751e26334eccb8e2718ed9ccd1b40d File: ./near-plugins-derive/src/upgradable.rs SHA3: 36339bc037dfc4d083f8fa9f4ddeee6e33ad17d74a688ea609d32515fbc60f6d File: ./near-plugins-derive/src/utils.rs SHA3: 372349d184b233ff85f6376a98f849cca07082a27c32c2b505208c2ac20c6edb File: ./near-plugins/src/access_control_role.rs SHA3: c520a2c4cc140e53cdf52f6e3d1a5acdf46f3580b61603f35b648fbea6d96dc8 File: ./near-plugins/src/access_controllable.rs SHA3: c35ca2cfa7de02428eb6525ba16bfe42f7a98525d4591aec71415b3e92941024 File: ./near-plugins/src/events.rs SHA3: b792393ae0eb23a607d8e560520d9bd3d147b56add485254860c90d9cc17b2ac File: ./near-plugins/src/full_access_key_fallback.rs SHA3: 5ab3af9827707b0b318378b582ebcf9c59143e51476fffd0ad7e39580dc05a12 File: ./near-plugins/src/lib.rs SHA3: 6a389ed5e8a32134974b8b98cc54bfc773fd24a31bee37c719f4960d87568a6b

	File: ./near-plugins/src/ownable.rs SHA3: 448fc7a497c6cb21a6ce001877b6f0485ef5e5957c421e6c05ce6ea2eec03487 File: ./near-plugins/src/pausable.rs SHA3: b7f327a2d470a5065a5ef173f708160ec3cb3d788e5effef26a23f4ddfdd49c0 File: ./near-plugins/src/upgradable.rs SHA3: f16837d2019644101bca8a9de5d4ca1f963082f2234f71f52925efe035933cd7

Third review scope

Repository	https://github.com/aurora-is-near/near-plugins/
Commit	93e1a30d79e72d51c4349ba71d454865c76ac690
Functional Requirements	Link
Technical Requirements	Link
Contracts	File: ./near-plugins-derive/src/access_control_role.rs SHA3: 8be164134dc10c9b0c09bac549886a615b84a000be0a002cf21d2f047113a519 File: ./near-plugins-derive/src/access_controllable.rs SHA3: 89b47a8ee90c2a3d0ccebe927dc923a06fb5731ece6405974f1537e7a8b7da9b File: ./near-plugins-derive/src/lib.rs SHA3: 02bcad5a7125f6200d7c814ab80708568b4ad4f6c20be38ed13faf5fde5a14c1 File: ./near-plugins-derive/src/ownable.rs SHA3: d49f6cf1fb8c1c614840ad86f70a31a7fc1842fc945b4d72999e8caa496ef688 File: ./near-plugins-derive/src/pausable.rs SHA3: 2a3cf41b587d9dda7a6b39b8a9f873b96992e6812581c4f86f454c7f24570d93 File: ./near-plugins-derive/src/upgradable.rs SHA3: 2363fb25aa3ac7eb096885fd0d49d95ecc24666ad76334084c0542ad6cee0407 File: ./near-plugins-derive/src/utils.rs SHA3: 372349d184b233ff85f6376a98f849cca07082a27c32c2b505208c2ac20c6edb File: ./near-plugins/src/access_control_role.rs SHA3: e7d886537c2447c35b5ad6e5d181efe073370c6820b077ef8547046e431b3c17 File: ./near-plugins/src/access_controllable.rs SHA3: 383ff16e4911cb00d166138b08c9f56448ef496bf1824a687b62bf9b1acbd617 File: ./near-plugins/src/events.rs SHA3: b792393ae0eb23a607d8e560520d9bd3d147b56add485254860c90d9cc17b2ac File: ./near-plugins/src/lib.rs SHA3: 5cd39a48ade34603fe7da07e2c6ce4dab0eeb662d64734e620a3659c95008198 File: ./near-plugins/src/ownable.rs SHA3: dfdcf2754d772133b58bd69b9e2a4d2894e269e029c581b8f8b14da9214accfd File: ./near-plugins/src/pausable.rs SHA3: d9f1c01bb1f2b649e2c051f72be9a097d2987e7640fb57aa16e4af4a704c0308 File: ./near-plugins/src/upgradable.rs SHA3: a852a6c18938a2ff62d9e762ea470587b437f56ab73733269a0819f41fc1ca1a

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 assets 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 the 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 **9** out of **10**.

- Functional requirements are provided.
- Use Cases with examples are provided.
- Technical description on how to compile code, run tests, check tests code coverage is not provided.

Code quality

The total Code Quality score is **9.5** out of **10**.

- The development environment is configured.
- Cargo aliases at *.cargo/config* as an alternative to technical description are not provided.

Test coverage

Code coverage of the project is **100%**.

Security score

As a result of the audit, the code does not contain any 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**.

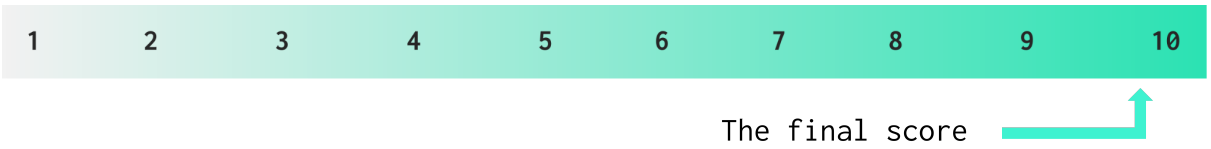

Table. The distribution of issues during the audit

Review date	Low	Medium	High	Critical
9 December 2022	0	0	0	0
9 March 2023	6	4	0	0
27 March 2023	0	0	0	0

Checked Items

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

Item	Description	Status
Default Visibility	Functions and state variables visibility should be set explicitly. Visibility levels should be specified consciously.	Passed
Integer Overflow and Underflow	If unchecked math is used, all math operations should be safe from overflows and underflows.	Passed
Outdated Compiler Version	It is recommended to use a recent version of the Rust compiler.	Passed
Access Control & Authorization	Ownership takeover should not be possible. All crucial functions should be protected. Users could not affect data that belongs to other users.	Passed
Assert Violation	Properly functioning code should never reach a failing assert statement.	Passed
Deprecated Rust Functions	Deprecated built-in functions should never be used.	Passed
DoS (Denial of Service)	Execution of the code should never be blocked by a specific contract state unless required.	Passed
Block values as a proxy for time	Block numbers should not be used for time calculations.	Not Relevant
Shadowing State Variable	State variables should not be shadowed.	Passed
Weak Sources of Randomness	Random values should never be generated from Chain Attributes or be predictable.	Not Relevant
Calls Only to Trusted Addresses	All external calls should be performed only to trusted addresses.	Passed
Presence of Unused Variables	The code should not contain unused variables if this is not justified by design.	Passed
Near Standards Violation	Near standards should not be violated.	Passed
Assets Integrity	Funds are protected and cannot be withdrawn without proper permissions or be locked on the contract.	Passed

User Balances Manipulation	Contract owners or any other third party should not be able to access funds belonging to users.	Passed
Data Consistency	Smart contract data should be consistent all over the data flow.	Not Relevant
Flashloan Attack	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	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	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.	Not Relevant
Style Guide Violation	Style guides and best practices should be followed.	Passed
Requirements Compliance	The code should be compliant with the requirements provided by the Customer.	Passed
Environment Consistency	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	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	The code should be covered with unit tests. Test coverage should be 100%, with both negative and positive cases covered. Usage of contracts by multiple users should be tested.	Passed

System Overview

Near Plugins are the implementation of common patterns used for NEAR Protocol smart contracts.

- Access Controllable – provides an ability to implement role-dependent functionality. Each role has its admin role and all admin roles are under the super admin role.
- Ownable – provides an ability to implement owner-dependent functionality.
- Pausable – provides an ability to implement an emergency stop mechanism that can be triggered by users' own authorized roles.
  Functionality could be paused partially using specific keys or all at once using the *ALL* key.
  It is possible to specify if the users who have authorized roles should be able to perform actions during the pause.
  It is possible to provide alternative functionality which would be available while the contract is paused.
- Upgradable – provides the ability to change smart contract code.
  It is possible to set up a delay before smart contract code is updated.
  It is impossible to deploy code until the delay is gone.

Privileged roles

Access Controllable:

- Super Admin:
  - grants/revokes super admin permissions
  - grants/revokes any role admin permissions
  - grants/revokes any role
- Role Admin:
  - grants/revokes the role admin permissions
  - grants/revokes the role

Ownable:

- Owner:
  - can execute special owned functionality

Pausable:

- Users owns authorized roles:
  - can pause/unpause the contract
  - can execute special functionality during the pause

Upgradable:

- Users owns authorized roles:
  - can stage and update the contract code
  - can update the delay between code staging and deployment
  - can deploy staged code

Risks

- The traits use specific storage keys to store critical data. The keys should not be used to store any other data as this may lead to data corruption. Used keys:
  - Ownable: *__OWNER__* (can be reassigned by assigning an *owner_storage_key* option).
  - Pausable: *__PAUSE__* (can be reassigned by assigning a *paused_storage_key* option).
  - Upgradable: prefix *__up__* (can be reassigned by assigning *storage_prefix* option).
  - Access Controllable: prefix *__acl* (can be reassigned by assigning a *storage_prefix* option).
- In case the permissive account id is a named account and the account was deleted from the Near Protocol by the private key owner, the name may be reused by someone and that user automatically becomes permissive on the contract.
  It is recommended to use implicit accounts for being highly permissive accounts.
- In case the ownership (Ownable) / super admin (Access Controllable) role is renounced (transferred to *None* or *null*), a new owner / super admin may be assigned by the contract's private key owner or by the contract self-call. In case an unprotected function makes a self call to an arbitrary place or the *owner_set* / *acl_init_super_admin* function, the ownership / super admin role may be taken over.
  It is recommended to avoid implementing unprotected self call functions.
- A role admin (Access Controllable) may remove/add any other role admins of/for the role and temporarily block some contract functionality.
  However, the super admin is able to cancel the actions and return the contract to a normal state.
- Unconscious contract code update (Upgradable) may lead to contract Storage Corruption, a Denial of Service situation, or Funds Lock.
  The setup call paramenters are provided on deployment and could not be validated in advance.
  It is recommended to ensure that the contract owner is trusted and would not unconsciously update the contract code.
- As the code should be firstly loaded on the contract and only then deployed (Upgradable), there is a delay during which anyone cancompare old and new implementations.
  In case a code update is needed to fix a security issue, an attacker may find the issue during code comparison and exploit the issue before the upgrade.
  It is recommended to load the code and deploy it in one batch transaction. In this case the delay is too short and the risk of an exploit is considered negligible.

- Although, there is a delay between the code staged and deployed (Upgradable), the *up_stage_update_staging_duration* allows to update the delay. The new staging duration could be applied after the current duration period is gone.
  It is recommended to ensure that the contract does not use zero delay and zero delay was not proposed last time.

Findings

Critical

No critical severity issues were found.

High

No high severity issues were found.

Medium

M01. Missing Modularity

As several functionality groups may be implemented, it may be required to have different managers which are able to pause the specific groups. The owner's permission to pause all the functionality may be unexpected.

This may lead to an inability to implement flexible pause functionality.

Path:
near-plugins-derive/src/pausable.rs: pa_unpause_feature(), pa_pause_feature()

Recommendation: separate the owner-pausing functionality in a different module, let developers use the functionality directly and provide limitations themselves.

Status: Fixed (Revised commit: 93e1a30)

M02. Missing Ownership Renounce

As the ownership-renounce functionality is not provided, wrong assumptions on how to remove the contract owner may arise.

This may lead to an unexpected contract takeover in case the owner deletes the account as the account name may be taken by another user.

Path:
near-plugins-derive/src/ownable.rs: owner_set()

Recommendation: provide a clear mechanic of ownership renouncement without any ability to set a new owner in the future.

Status: Mitigated (Ownership could be revoked using *owner_set(None)*)

M03. Possible Storage Keys Coincidence

The storage key names are not project-specific. Developers may use the same storage keys in their project.

This may lead to unexpected and hardly traceable errors during development.

It is considered providing unique storage key names which includes the package name.
For example: `__aurora:near-plugins:ownable:owner__`.

Paths:
near-plugins/src/access_controllable.rs: acl_storage_prefix()
near-plugins-derive/src/access_controllable.rs
near-plugins/src/ownable.rs: owner_storage_key()
near-plugins-derive/src/ownable.rs
near-plugins/src/pausable.rs: pa_storage_key()
near-plugins-derive/src/pausable.rs
near-plugins/src/upgradable.rs: up_storage_key()
near-plugins-derive/src/upgradable.rs

Recommendation: make the storage key names contain the crate name.

Status: Mitigated (Longer storage keys makes storage read/write using more Gas)

M04. Missing Super Admin Transfer

The Super Admin transfer functionality is not provided and by default there can be only one super admin set up.

This may lead to inability to change the super admin at critical situations.

Path:
near-plugins-derive/src/access_controllable.rs:
acl_init_super_admin()

Recommendation: provide an ability to transfer the super admin role or unblock several super admins functionality.

Status: Fixed (Revised commit: 93e1a30)

Low

L01. Redundant Clones

Checks for a redundant clone() (and its relatives) which clones an owned value that is going to be dropped without further use.
It is not always possible for the compiler to eliminate useless allocations and deallocations generated by redundant clone()s.

Path:
near-plugins/src/pausable.rs: test_pause_feature_from_owner(), test_pause_only_owner(), test_pausw_only_owner_not_self(), test_pause_with_all()

Recommendation: In nightly builds, it is possible to use cargo Clippy --fix to apply some suggestions from Clippy.

Status: Fixed (Revised commit: 7454bbc)

L02. Code duplication

The *EventMetadata* with the fixed *standard* and *version* fields is commonly used throughout the project.

As the *standard* and *version* should always be the same for each of the plugins, the functionality is considered to be implemented once.

Paths:
near-plugins/src/pausable.rs: AsEvent
near-plugins/src/upgradable.rs: AsEvent

Recommendation: move the *EventMetadata* struct generation to a separate function.

Status: Mitigated (The fields are duplicated no more than 2 times and do not affect code readability)

L03. Wrong Error Message

According to the implementation, the error occurs when methods on the specified key are paused.

The *if_paused* modifier may return a *Pausable: Method must be paused* error which is not sensible as the called method should not be paused to perform the call.

Path:
near-plugins-derive/src/pausable.rs: if_paused()

Recommendation: make the error message represent the nature of the error.

Status: Fixed (Revised commit: 93e1a30)

L04. Missing Documentation

It is mentioned in the docs for the Ownable, Pausable and Access Controllable modules how to provide custom storage keys. However, the information is missing for the Upgradable module.

Path:
near-plugins/src/upgradable.rs: up_storage_key()

Recommendation: provide corresponding documentation to the implemented features.

Status: Fixed (Revised commit: 93e1a30)

L05. Code Duplication

As the events module implements the *emit* method, the method should be used for event logging.

Paths:
near-plugins-derive/src/full_access_key_fallback.rs: attach_full_access_key()
near-plugins-derive/src/ownable.rs: owner_set()
near-plugins-derive/src/pausable.rs: pa_pause_feature(), pa_unpause_feature()

Recommendation: use the *emit* method to burn an event.

Status: Fixed (Revised commit: 93e1a30)

L06. Misleading Documentation

In the documentation for the Upgradable plugin, there is a statement that:

After the code is deployed, it should be removed from staging. This will prevent old code with a security vulnerability to be deployed.

However, it looks slippery as the code deployed and the code staged are identical and no old security issues should appear.

Path:
near-plugins/src/upgradable.rs

Recommendation: make the statement clear, provide more explanation on the possible vulnerability of the staged code.

Status: Fixed (Revised commit: 93e1a30)

L07. Different Documentation Format

The documentation for the Access Controllable trait is provided in a style that differs from other plugin descriptions.

The plugin does not contain a general overview marked with *//!* comments at the start of the file.

Path:
near-plugins/src/access_controllable.rs

Recommendation: provide documentation for the plugins in the same style.

Status: Fixed (Revised commit: 93e1a30)

Disclaimers

Hacken Disclaimer

The smart contracts given for audit have been analyzed by the best industry practices at the date 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 to 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, Consultant cannot guarantee the explicit security of the audited smart contracts.