VIBRANIUM AUDITS

Security Assessment

Pulse Markets

Verified by Vibranium Audits on 1 June 2023

Vibranium Audits Verified on Jne 1st, 2023

Pulse Markets

The security assessment was prepared by Vibranium Audits.

Executive Summary

TYPES	ECOSYSTEM	METHODS
DEFI	Near	Manual Review & Static Analysis

LANGUAGE	TIMELINE	KEY COMPONENTS
Rust	Delivered on 07/06/2023	N/A

CODEBASE	COMMITS
https://github.com/aufacicenta/pulsemarkets-v2-contracts	ddce0b7abc1f66f33f611e20e4d2b1039f95747f

Vulnerability Summary

29	0	0	0	29	0	29
Total Findings	Resolved	Mitigated	Partially Resolved	Acknowledged	Declined	Unresolved

0 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.
14 High	0 Resolved		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.
1 Medium	0 Resolved		Medium vulnerabilities are usually limited to state manipulations, but cannot lead to assets loss. Major deviations from best practices are also in this category.
4 Low	0 Resolved		Low vulnerabilities are related to outdated and unused code or minor gas optimization.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.
9 Informational	0 Resolved		Informational errors are often recommendations to improve the style of the code or certain operations to fall within industry best practices. They usually do not affect the overall functioning of the code.

TABLE OF CONTENTS | PULSEMARKETS

CODEBASE | PULSE MARKETS

Repository

https://github.com/aufacicenta/pulsemarkets-v2-contracts

Commit

ddce0b7abc1f66f33f611e20e4d2b1039f95747f

AUDIT SCOPE | PULSE MARKETS

28 files audited ● 28 file with Acknowledged findings ● 0 files with Resolved findings

ID	Files	Commit Hash
● AVA	All files under pulsemarkets-v2-contracts/amm/src	2ec655ff6ef9d1ef0cce68777c983e04652a15ec
● FVA	All files under pulsemarkets-v2-contracts/feed-parser/src	e50e2d6946cd513c19c0f8dfcd9eb27813cbb5b5
● MVA	All files under pulsemarkets-v2-contracts/market-factory/src	24d4fd571ebc127270285bec825cdeda28c4ba5c
● SVA	All files under pulsemarkets-v2-contracts/shared/src	e50e2d6946cd513c19c0f8dfcd9eb27813cbb5b5
● GVA	Encompassing all code within the Pulse Markets scope repository	ddce0b7abc1f66f33f611e20e4d2b1039f95747f

APPROACH & METHODS | PULSE MARKETS

This report has been prepared for Pulse Markets(2023) to discover issues and vulnerabilities in the source code of the Pulse Markets project as well as any contract dependencies that were not part of an officially recognized library. A comprehensive examination has been performed, utilizing Manual Review and Static Analysis techniques.

The auditing process pays special attention to the following considerations:

- Testing the smart contracts against both common and uncommon attack vectors.
- Assessing the code base to ensure compliance with current best practices and industry standards
- Ensuring contract logic meets the specifications and intentions of the client.
- Cross referencing contract structure and implementation against similar smart contracts produced by industry leaders.
- Thorough line-by-line manual review of the entire code base by industry experts.

The security assessment resulted in findings that ranged from critical to informational We recommend addressing these findings to ensure a high level of security standards and industry practices.
We suggest recommendations that could better serve the project from the security perspective:

- Testing the smart contracts against both common and uncommon attack vectors.
- Enhance general coding practices for better structures of source codes.
- Review unit tests to cover the possible use cases.
- Review functions for readability, especially for future development work.

FINDINGS | PULSE MARKETS

29	0	12	5	5	4
Total Findings	Critical	High	Medium	Low	Informational

This report has been prepared to discover issues and vulnerabilities for Pulse Markets. Through this audit, we have uncovered 29 issues ranging from different severity levels. Utilizing the techniques of Manual Review & Static Analysis to complement rigorous manual code reviews, we discovered the following findings:

ID	Title	Category	Severity	Status
MVA-01	Unhandled Promise Rejection	Logical Issue	High	Acknowledged
MVA-02 AVA-01	Incorrect Json Type	Logical Issue	High	Acknowledged
MVA-03	Assert in callback function may lock contract	Logical Issue	Minor	Acknowledged
MVA-04	Lack of gas check for storage expansion	Logical Issue	Medium	Acknowledged
FVA-01	Unhandled Promise	Logical Issue	High	Acknowledged
GVA-01	No Upgrade Function Found	Logical Issue	Minor	Acknowledged
GVA-02	Lack of Documentation	Coding Style	Informational	Acknowledged
GVA-03	Crate Update Needed	Dependency Issue	Medium	Acknowledged

MVA-01 | Unhandled Promise Rejection

Category	Severity	Location	Status
Logical Issue	High	market-factory/src/contract.rs	Acknowledged

Description

Promise results should always be handled by a callback function or another promise. It is not recommended to leave promises unhandled in contracts. Otherwise, the changed state cannot be rolled back if the promise failed.

This is present in pub fn create_market(&mut self, name: AccountId, args: Base64VecU8) -> Promise {...}, line 67.

Potential loss of user funds in case of failed promise.

```
55        // @TODO if this promise fails, the funds (attached_deposit) are not returned to the signer
56        let create_market_promise = Promise::new(market_account_id.clone())
57            .create_account()
58            .deploy_contract(MARKET_CODE.to_vec())
59            .transfer(env::attached_deposit() - STORAGE_DEPOSIT_BOND)
60            .function_call(
61                "new".to_string(),
62                init_args.to_string().into_bytes(),
63                0,
64                GAS_FOR_CREATE_MARKET,
65            );
66
67        let create_market_callback = ext_self::ext(env::current_account_id())
68            .with_attached_deposit(0)
69            .with_static_gas(GAS_FOR_CREATE_MARKET_CALLBACK)
70            .on_create_market_callback(market_account_id, collateral_token_account_id);
71
72        create_market_promise.then(create_market_callback)
73    }
```

Recommendation

Implement a function call of the result of the promise if it fails.

MVA-02 | Incorrect Json Type

Category	Severity	Location	Status
Logical Issue	High	market-factory/src/views.rs	Acknowledged

Description (Finding is present in 2 functions in views.rs)

Don't use type i64, i128, u64, or u128 as the parameters or return values of public interfaces (public functions without #[private] macro in a #[near_bindgen] struct). This is because the largest integer that json can support is 2^53-1.
Check https://2ality.com/2012/04/number-encoding.html for more details.

```
11      pub fn get_markets_count(&self) -> u64 {
12          self.markets.len()
13      }
14
15      pub fn get_markets(&self, from_index: u64, limit: u64) -> Vec<AccountId> {
16          let elements = self.markets.as_vector();
17
18          (from_index..std::cmp::min(from_index + limit, elements.len()))
19              .filter_map(|index| elements.get(index))
20              .collect()
21      }
```

Recommendation

We advise using near_sdk::json_types instead.
Types I64, I128, U64, and U128 (instead of i64, i128, u64 and u128) in Near SDK are recommended.

AVA-01 | Incorrect Json Type

Category	Severity	Location	Status
Logical Issue	High	amm/src/views.rs amm/src/fees.rs amm/src/contract.rs	Acknowledged

Description

Don't use type i64, i128, u64, or u128 as the parameters or return values of public interfaces (public functions without #[private] macro in a #[near_bindgen] struct). This is because the largest integer that json can support is 2^53-1.
Check https://2ality.com/2012/04/number-encoding.html for more details.

Finding is present across multiple functions:

- views.rs:
  - pub fn get_market_data(&self) -> MarketData {...}
  - pub fn get_resolution_data(&self) -> Resolution {...}
  - pub fn get_outcome_token(&self, outcome_id: OutcomeId) -> OutcomeToken {...}
  - pub fn get_block_timestamp(&self) -> Timestamp {...}
  - pub fn resolution_window(&self) -> Timestamp {...}
  - pub fn resolved_at(&self) -> Timestamp {...}
- fees.rs:
  - pub fn claiming_window(&self) -> Timestamp {...}
- contract.rs
  - pub fn new(market: MarketData, ...) -> Self {..}

Recommendation

We advise using near_sdk::json_types instead.
Types I64, I128, U64, and U128 (instead of i64, i128, u64 and u128) in Near SDK are recommended.

FVA-01 | Unhandled Promise

Categor y	Severity	Location	Status
Logical Issue	High	feed-parser/src/callback.rs	Acknowledged

Description

Promise results should always be handled by a callback function or another promise. It is not recommended to leave promises unhandled in contracts. Otherwise, the changed state cannot be rolled back if the promise failed.

```
45          // @TODO add a callback for this promise in case it errors
46          ext_market::ext(predecessor_account_id).resolve(winning_outcome_id, payload.ix);
47
48          return winning_outcome_id;
49      }
```

Potential state lock if promise is not handled.

Recommendation

Implement a callback function or another promise to handle current existing promise.

MVA-03 | Assert in callback function may lock contract

Category	Severity	Location	Status
Logical Issue	Medium	marketfactory/src/callback.rs	Acknowledged

Description

It's not recommended to use **assert** or **require** or anything that will result in panic in a callback function.

In Near, the callback function needs to recover some state changes made by a failed promise. If the callback function panics, the state may not be fully recovered, which results in unexpected results.

```
57    #[private]
58    pub fn on_create_outcome_tokens_ft_storage_deposit_callback(
59        &mut self,
60        market_account_id: AccountId,
61    ) -> bool {
62        require!(env::promise_results_count() == 2);
63
64        let are_outcome_tokens_created = match env::promise_result(0) {
65            PromiseResult::Successful(_result) => true,
66            _ => env::panic_str("ERR_ON_CREATE_OUTCOME_TOKENS_CALLBACK_0"),
67        };
68
69        let is_storage_deposit_success = match env::promise_result(1) {
70            PromiseResult::Successful(_result) => true,
71            _ => env::panic_str("ERR_ON_FT_STORAGE_DEPOSIT_CALLBACK_1"),
72        };
73
74        if !are_outcome_tokens_created || !is_storage_deposit_success {
75            return false;
76        }
77
78        self.markets.insert(&market_account_id);
79
80        true
81    }
```

Recommendation

Avert from using the **require** statement within the callback functions.

MVA-04 | Lack of gas check for storage expansion

Category	Severity	Location	Status
Logical Issue	Low	marketfactory/src/callback.rs	Acknowledged

Description

Each time the state grows, it should be ensured that there is enough Balance to cover the expansion.

```
57    #[private]
58    pub fn on_create_outcome_tokens_ft_storage_deposit_callback(
59        &mut self,
60        market_account_id: AccountId,
61    ) -> bool {
62        require!(env::promise_results_count() == 2);
63
64        let are_outcome_tokens_created = match env::promise_result(0) {
65            PromiseResult::Successful(_result) => true,
66            _ => env::panic_str("ERR_ON_CREATE_OUTCOME_TOKENS_CALLBACK_0"),
67        };
68
69        let is_storage_deposit_success = match env::promise_result(1) {
70            PromiseResult::Successful(_result) => true,
71            _ => env::panic_str("ERR_ON_FT_STORAGE_DEPOSIT_CALLBACK_1"),
72        };
73
74        if !are_outcome_tokens_created || !is_storage_deposit_success {
75            return false;
76        }
77
78        self.markets.insert(&market_account_id);
79
80        true
81    }
```

Recommendation

storage usage expands after the insertion to self.markets.insert(&market_account_id), the difference should be checked to ensure the storage fee attached by the caller is enough.

GVA-01 | No upgrade function found

Category	Severity	Location	Status
Logical Issue	Low	Global	Acknowledged

Description

Contracts **may** need the upgrade function.
Without an upgrade function, the contract cannot be upgraded and contract states cannot be migrated.

Recommendation

It is recommended to study the need of implementing upgradability in the future and implementing an upgrade function globally if deemed possibly necessary to update contract states in the future.

GVA-02 | Lack of documentation

Categor y	Severity	Location	Status
Logical Issue	Informational	Global	Acknowledged

Description

Althought the Pulse Markets contract set features good documentation on the main contracts (specifically amm/src/contract.rs), it is recommended to fully document all the written code for better future communication between developers.

GVA-03 | Crate Update Needed

Category	Severity	Location	Status
Dependency Issue	Medium	Global	Acknowledged

Description

Potential segfault in the time crate due to version "0.1.44", you can learn more at: https://rustsec.org/advisories/RUSTSEC-2020-0071

Recommendation

Upgrade to >=0.2.23.

DISCLAIMER | VIBRANIUM AUDITS

This report is subject to the terms and conditions (including without limitation description of services confidentiality disclaimer and limitation of liability) set forth in the Services Agreement or the scope of services and terms and conditions provided to you (“Customer” or the “Company”) in connection with the Agreement This report provided in connection with the Services set forth in the Agreement shall be used by the Company only to the extent permitted under the terms and conditions set forth in the Agreement This report may not be transmitted disclosed referred to or relied upon by any person for any purposes nor may copies be delivered to any other person other than the Company without Vibranium Audits prior written consent in each instance

This report is not nor should be considered an “endorsement” or “disapproval” of any particular project or team This report is not nor should be considered an indication of the economics or value of any “product” or “asset”created by any team or project that contracts Vibranium Audits to perform a security assessment This report does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed nor do they provide any indication of the technologies proprietors business business model or legal compliance

This report should not be used in any way to make decisions around investment or involvement with any particular project This report in no way provides investment advice nor should be leveraged as investment advice of any sort This report represents an extensive assessing process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology

Blockchain technology and cryptographic assets present a high level of ongoing risk Vibranium Audits position is that each company and individual are responsible for their own due diligence and continuous security Vibranium Audits goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies and in no way claims any guarantee of security or functionality of the technology we agree to analyze

The assessment services provided by Vibranium Audits is subject to dependencies and under continuing development You Agree that your access and/or use including but not limited to any services reports and materials will be at your sole risk on an as-is where-is and as-available basis Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty The assessment reports could include false positives false negatives and other unpredictable results The services may access and depend upon multiple layers of third-parties

ALL SERVICES THE LABELS THE ASSESSMENT REPORT WORK PRODUCT OR OTHER MATERIALS OR ANY PRODUCTS OR RESULTS OF THE USE THEREOF ARE PROVIDED “AS IS” AND “AS AVAILABLE” AND WITH ALL FAULTS AND DEFECTS WITHOUT WARRANTY OF ANY KIND TO THE MAXIMUM EXTENT PERMITTED UNDER APPLICABLE LAW VIBRANIUM AUDITS HEREBY DISCLAIMS ALL WARRANTIES WHETHER EXPRESS IMPLIED STATUTORY OR OTHERWISE WITH RESPECT TO THE SERVICES ASSESSMENT REPORT OR OTHER MATERIALS WITHOUT LIMITING THE FOREGOING VIBRANIUM AUDITS SPECIFICALLY DISCLAIMS ALL IMPLIED WARRANTIES OF MERCHANTABILITY FITNESS FOR A PARTICULAR PURPOSE TITLE AND NON-INFRINGEMENT AND ALL WARRANTIES ARISING FROM COURSE OF DEALING USAGE OR TRADE PRACTICE WITHOUT LIMITING THE FOREGOING VIBRANIUM AUDITS MAKES NO WARRANTY OF ANY KIND THAT THE SERVICES THE LABELS THE ASSESSMENT REPORT WORK PRODUCT OR OTHER MATERIALS OR ANY PRODUCTS OR RESULTS OF THE USE THEREOF WILL MEET CUSTOMER’S OR ANOTHER PERSON’S REQUIREMENTS ACHIEVE ANY INTENDED RESULT BE COMPATIBLE OR WORK WITH ANY SOFTWARE SYSTEM OR OTHER SERVICES OR BE SECURE ACCURATE COMPLETE FREE OF HARMFUL CODE

OR ERROR-FREE WITHOUT LIMITATION TO THE FORGOING, VIBRANIUM AUDITS PROVIDES NO WARRANTY OR UNDERTAKING, AND MAKES NO REPRESENTATION OF ANY KINDTHAT WILL MEET CUSTOMER' S REQUIREMENTS ACHIEVE ANY INTENDED RESULTS BE COMPATIBLE OR WORK WITH ANY OTHER SOFTWARE APPLICATIONS SYSTEMS OR SERVICES OPERATE WITHOUT INTERRUPTION MEET ANY PERFORMANCE OR RELIABILITY STANDARDS OR BE ERROR FREE OR THAT ANY ERRORS OR DEFECTS CAN OR WILL BE CORRECTED

WITHOUT LIMITING THE FOREGOING NEITHER VIBRANIUM AUDITS NOR ANY OF VIBRANIUM AUDITS AGENTS MAKES ANY REPRESENTATION OR WARRANTY OF ANY KIND EXPRESS OR IMPLIED AS TO THE ACCURACY RELIABILITY ORCURRENCY OF ANY INFORMATION OR CONTENT PROVIDED THROUGH THE SERVICE VIBRANIUM AUDITS WILL ASSUME NO LIABILITY OR RESPONSIBILITY FOR (I) ANY ERRORS MISTAKES OR INACCURACIES OF CONTENT AND MATERIALS OR FOR ANY LOSS OR DAMAGE OF ANY KIND INCURRED AS A RESULT OF THE USE OF ANY CONTENT OR (II) ANY PERSONAL INJURY OR PROPERTY DAMAGE OF ANY NATURE WHATSOEVER RESULTING FROM CUSTOMER'S ACCESS TO OR USE OF THE SERVICES ASSESSMENT REPORT OR OTHER MATERIALS

ALL THIRD-PARTY MATERIALS ARE PROVIDED "AS IS" AND ANY REPRESENTATION OR WARRANTY OF OR CONCERNING ANY THIRD-PARTY MATERIALS IS STRICTLY BETWEEN CUSTOMER AND THE THIRD-PARTY OWNEROR DISTRIBUTOR OF THE THIRD-PARTY MATERIALS

THE SERVICES ASSESSMENT REPORT AND ANY OTHER MATERIALS HERE UNDER ARE SOLELY PROVIDED TO CUSTOMER AND MAY NOT BE RELIED ON BY ANY OTHER PERSON OR FOR ANY PURPOSE NOT SPECIFICALLY IDENTIFIED IN THIS AGREEMENT NOR MAY COPIES BE DELIVERED TO ANY OTHER PERSON WITHOUT VIBRANIUM AUDITS PRIOR WRITTEN CONSENT IN EACH INSTANCE

NO THIRD PARTY OR ANYONE ACTING ON BEHALF OF ANY THEREOF SHALL BE A THIRD PARTY OR OTHER BENEFICIARY OF SUCH SERVICES ASSESSMENT REPORT AND ANY ACCOMPANYING MATERIALS AND NO SUCH THIRD PARTY SHALL HAVE ANY RIGHTS OF CONTRIBUTION AGAINST VIBRANIUM AUDITS WITH RESPECT TO SUCH SERVICES ASSESSMENT REPORT AND ANY ACCOMPANYING MATERIALS

THE REPRESENTATIONS AND WARRANTIES OF VIBRANIUM AUDITS CONTAINED IN THIS AGREEMENT ARE SOLELY FOR THE BENEFIT OF CUSTOMER ACCORDINGLY NO THIRD PARTY OR ANYONE ACTING ON BEHALF OF ANY THEREOF SHALL BE A THIRD PARTY OR OTHER BENEFICIARY OF SUCH REPRESENTATIONS AND WARRANTIES AND NO SUCH THIRD PARTY SHALL HAVE ANY RIGHTS OF CONTRIBUTION AGAINST VIBRANIUM AUDITS WITH RESPECT TO SUCH REPRESENTATIONS OR WARRANTIES OR ANY MATTER SUBJECT TO OR RESULTING IN INDEMNIFICATION UNDER THIS AGREEMENT OR OTHERWISE

FOR AVOIDANCE OF DOUBT THE SERVICES INCLUDING ANY ASSOCIATED ASSESSMENT REPORTS ORMATERIALS SHALL NOT BE CONSIDERED OR RELIED UPON AS ANY FORM OF FINANCIAL, TAX, LEGAL, REGULATORY, OR OTHER ADVICE.

Vibranium | Securing the Web3 World

Vibranium Audits is a blockchain security company that was founded in 2021 by professors from the University of Greenwich and cyber-security engineers from ITI Capital. As pioneers in the field, Vibranium Audits utilizes best-in-class Formal Verification and AI technology to secure and monitor blockchains, smart contracts, and Web3 apps.