Report

v. 1.0

Customer

StarkWare

Cairo code audit

Perpetual. V 3.1

7th November 2023

Report prepared by

ABDK Consulting

Contents

1 Changelog

#	Date	Author	Description
0.1	07.11.23	A. Zveryanskaya	Initial Draft
0.2	07.11.23	A. Zveryanskaya	Minor revision
1.0	07.11.23	A. Zveryanskaya	Release

2 Introduction

The following document provides the result of the audit performed by ABDK Consulting (Mikhail Vladimirov and Dmitry Khovratovich) at the customer request. The audit goal is a general review of the smart contracts structure, critical/major bugs detection and issuing the general recommendations.

StarkWare Industries is an Israeli software company that specializes in cryptography. It develops zero-knowledge proof technology that compresses information to address the scalability problem of the blockchain, and works on the Ethereum platform.

3 Project scope

We have audited the version 3.1 of Perpetual project, concretely the diff between commit d1edb22 and commit eaa2846 in the starkware-libs/stark-perpetual repository. The fixes were provided via commit e6189a.

Files:

exchange/definitions/		
constants.cairo		
perpetual/definitions/		
general _config_hash.cairo	general_config.cairo	perpetual_error_code.cairo
perpetual/oracle/		
oracle_price.cairo		
perpetual/order/		
order.cairo		
perpetual/output/		
data_availability.cairo	program_output.cairo	
perpetual/position/		
add_asset.cairo	update_position.cairo	
perpetual/transactions/		
deleverage.cairo	funding_tick.cairo	liquidate.cairo
withdrawal.cairo		
perpetual/		
main.cairo		

4 Methodology

The methodology is not a strict formal procedure, but rather a selection of methods and tactics combined differently and tuned for each particular project, depending on the project structure and technologies used, as well as on client expectations from the audit.

- **General Code Assessment**. The code is reviewed for clarity, consistency, style, and for whether it follows best code practices applicable to the particular programming language used. We check indentation, naming convention, commented code blocks, code duplication, confusing names, confusing, irrelevant, or missing comments etc. At this phase we also understand overall code structure.
- **Entity Usage Analysis**. Usages of various entities defined in the code are analysed. This includes both: internal usages from other parts of the code as well as potential external usages. We check that entities are defined in proper places as well as their visibility scopes and access levels are relevant. At this phase, we understand overall system architecture and how different parts of the code are related to each other.
- **Access Control Analysis**. For those entities, that could be accessed externally, access control measures are analysed. We check that access control is relevant and done properly. At this phase, we understand user roles and permissions, as well as what assets the system ought to protect.
- **Code Logic Analysis**. The code logic of particular functions is analysed for correctness and efficiency. We check if code actually does what it is supposed to do, if that algorithms are optimal and correct, and if proper data types are used. We also make sure that external libraries used in the code are up to date and relevant to the tasks they solve in the code. At this phase we also understand data structures used and the purposes they are used for.

We classify issues by the following severity levels:

- **Critical issue** directly affects the smart contract functionality and may cause a significant loss.
- **Major issue** is either a solid performance problem or a sign of misuse: a slight code modification or environment change may lead to loss of funds or data. Sometimes it is an abuse of unclear code behaviour which should be double checked.
- **Moderate issue** is not an immediate problem, but rather suboptimal performance in edge cases, an obviously bad code practice, or a situation where the code is correct only in certain business flows.
- **Minor issues** contain code style, best practices and other recommendations.

5 Our findings

We found 1 major, and a few less important issues. All identified Major issues have been fixed.

	Info	Fixed
Major	0	1
Minor	6	3

Fixed 4 out of 10 issues

6 Major Issues

CVF-1. FIXED

- **Category** Readability
- **Source** order.cairo

Description Here the value of “fulfilled_amount” is used before being calculated. This makes code harder to read.

Recommendation Consider refactoring.

```
92 +let remaining_capacity = full_amount - fulfilled_amount;
```

```
83 94      ids.fulfilled_amount = __dict_manager.get_dict(ids.orders_dict)
              ↪ [ids.order_id]
```

7 Minor Issues

CVF-2. INFO

- **Category** Bad naming
- **Source** oracle_price.cairo

Description The term “upper bound” is ambiguous, as it is unclear, whether this bound is inclusive or exclusive.

Recommendation Consider defining a constant named “MAX_EXTERNAL_PRICE” instead for inclusive upper bound. This would make code simpler, easier to read and would allow using the full felt range if necessary.

Client Comment *Upper bound is always exclusive. I think changing it will be more confusing.*

```
91 +sig.external_price, EXTERNAL_PRICE_UPPER_BOUND - 1
```

CVF-3. FIXED

- **Category** Unclear behavior
- **Source** withdrawal.cairo

Recommendation Consider checking the key equality explicitly.

```
54 +local has_address = withdrawal.owner_key - withdrawal.base.
   ↪ public_key;
```

```
59 +if (has_address == 0) {
```

CVF-4. INFO

- **Category** Procedural
- **Source** funding_tick.cairo

Recommendation Consider extracting this repeating pattern as a function.

```
33 +assert_250_bit{range_check_ptr=range_check_ptr}(
34 +    max_funding_rate * price * timestamp_diff - funding_index_diff
   ↪ * FXP_32_ONE
35 +);
```

CVF-5. INFO

- **Category** Procedural
- **Source** deleverage.cairo

Recommendation Consider extracting this repeating pattern as a function.

```
163 +assert_250_bit{range_check_ptr=range_check_ptr}(
164 +    (initial_tv * updated_tr) - ((updated_tv - FXP_32_ONE) *
    ↪ initial_tr + 1)
165 +);
```

CVF-6. INFO

- **Category** Procedural
- **Source** liquidate.cairo

Recommendation Consider extracting this repeating pattern as a function.

```
99 +assert_250_bit{range_check_ptr=range_check_ptr}(updated_tr - (
    ↪ updated_tv * FXP_32_ONE + 1));
```

CVF-7. INFO

- **Category** Suboptimal
- **Source** add_asset.cairo

Recommendation Consider merging these branches together.

Client Comment *Changing the code will be less readable or less efficient.*

```
82 +right_start_ptr = left_end_ptr;
```

```
87 +    right_start_ptr = left_end_ptr;
```

CVF-8. INFO

- **Category** Suboptimal
- **Source** update_position.cairo

Recommendation Consider merging these two branches together.

Client Comment *Changing the code will be less efficient.*

```
125 +return (
126 +    range_check_ptr=range_check_ptr,
127 +    updated_position=funded_position,
128 +    funded_position=funded_position,
129 +    return_code=PerpetualErrorCode.INVALID_PUBLIC_KEY,
130 +);
```

```
133 +return (
134 +    range_check_ptr=range_check_ptr,
135 +    updated_position=funded_position,
136 +    funded_position=funded_position,
137 +    return_code=PerpetualErrorCode.INVALID_PUBLIC_KEY,
138 +);
```

CVF-9. FIXED

- **Category** Documentation
- **Source** data_availability.cairo

Description These constants are not used in the file.

Recommendation Consider documenting.

```
12 +const VALIDIUM_MODE = 0;
13 +const ROLLUP_MODE = 1;
```

CVF-10. FIXED

- **Category** Documentation
- **Source** constants.cairo

Recommendation Consider adding a comment explaining how this value was calculated.

```
5 +const MINT_TREE_INDEX_SALT = 0x6d696e74;
```

ABDK
Consulting

About us

Established in 2016, is a leading service provider in the space of blockchain development and audit. It has contributed to numerous blockchain projects, and co-authored some widely known blockchain primitives like Poseidon hash function.

The ABDK Audit Team, led by Mikhail Vladimirov and Dmitry Khovratovich, has conducted over 40 audits of blockchain projects in Solidity, Rust, Circom, C++, JavaScript, and other languages.

Contact

Email

dmitry@abdkconsulting.com

Website

abdk.consulting

Twitter

twitter.com/ABDKconsulting

LinkedIn

linkedin.com/company/abdk-consulting