Report

v. 1.0

Customer

Exactly

Smart Contract Audit

Escrowed EXA

26th September 2023

Report prepared by

ABDK Consulting

Contents

1 Changelog

#	Date	Author	Description
0.1	26.09.23	A. Zveryanskaya	Initial Draft
0.2	26.09.23	A. Zveryanskaya	Minor revision
1.0	26.09.23	A. Zveryanskaya	Release
1.1	26.09.23	A. Zveryanskaya	Project scope updated
1.2	26.09.23	A. Zveryanskaya	Client comments CVF-9, 12 updated
2.0	26.09.23	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.

Exactly is a decentralized, non-custodial and open-source protocol that provides an autonomous fixed and variable interest rate market enabling users to frictionlessly exchange the time value of their assets and completing the DeFi credit market.

3 Project scope

We were asked to review:

- Original Code
- Code with Fixes

Files:

periphery/ commit/1cc8f57672ad93e723acc15b4b4de7813fdfa51d

EscrowedEXA.sol Swapper.sol

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 3 major, and a few less important issues. All identified Major issues have been fixed.

	Info	Fixed
Major	0	3
Moderate	0	1
Minor	9	4

Fixed 8 out of 17 issues

6 Major Issues

CVF-1. FIXED

- **Category** Unclear behavior
- **Source** EscrowedEXA.sol

Description Granting a role to a zero address looks weird.

Recommendation Consider either removing this line or clearly explaining in a comment why this line is here.

Client Comment *The zero address needs the role so we can mint and burn the token. We added natspec for this.*

```
41 _grantRole(TRANSFERRER_ROLE, address(0));
```

CVF-2. FIXED

- **Category** Suboptimal
- **Source** EscrowedEXA.sol

Description It would be more efficient to check this inside the "_beforeTokenTransfer" hook.

Client Comment *We moved the logic to "_beforeTokenTransfer" hook.*

```
49 if (!hasRole(TRANSFERRER_ROLE, from) && !hasRole(TRANSFERRER_ROLE,
   ↪ to)) revert Untransferable();
```

CVF-3. FIXED

- **Category** Unclear behavior
- **Source** EscrowedEXA.sol

Description Unchained initializers should be used for all inherited contracts except for the first one.

Client Comment *We updated it to use unchained initializers.*

```
36 __ERC20Permit_init("escrowed EXA");
__ERC20Votes_init();
__AccessControl_init();
```

7 Moderate Issues

CVF-4. FIXED

- **Category** Unclear behavior
- **Source** Swapper.sol

Description In case keepETH > msg.value, this will send to ”account” less ether than the ”keepETH” value.

Recommendation Consider reverting in such a case.

Client Comment *We added natspec. This could come from a bridge, so we don’t revert on invalid value.*

```
49 if (keepETH >= msg.value) return account.safeTransferETH(msg.value);
```

8 Minor Issues

CVF-5. INFO

- **Category** Procedural
- **Source** EscrowedEXA.sol

Description Declaring a top-level error in a file named after a contract makes it harder to navigate through code.

Recommendation Consider either moving the error declaration into the contract or moving it into a separate file.

Client Comment *This is a style choice and we are being consistent with other Exactly contracts in which we follow the same code structure.*

145
```
error Untransferable();
```

CVF-6. INFO

- **Category** Procedural
- **Source** EscrowedEXA.sol

Description Defining top-level structs in a file named after a contract makes it harder to navigate through code.

Recommendation Consider either moving the structs definitions into the contract or moving them into a separate file.

Client Comment *This is a style choice and we are being consistent with other Exactly contracts in which we follow the same code structure.*

159
```
struct Durations {
```

164
```
struct Broker {
```

169
```
struct CreateWithDurations {
```

179
```
struct Permit {
```

CVF-7. INFO

- **Category** Procedural
- **Source** Swapper.sol

Description Defining top-level structures in a file named after a contract makes it harder to navigate through code.

Recommendation Consider either moving the struct definitions into the contract or moving them into a separate file.

Client Comment *This is a style choice and we are being consistent with other Exactly contracts in which we follow the same code structure.*

```
118 struct Permit {
```

```
126 struct Permit2 {
```

CVF-8. INFO

- **Category** Bad naming
- **Source** EscrowedEXA.sol

Description Events are usually named via nouns, such as ”ReserveRatio” or ”VestingPeriod”.

Client Comment *This is a style choice and we are being consistent with other Exactly contracts in which we follow the same code structure.*

```
139 event ReserveRatioSet(uint256 reserveRatio);
140 event VestingPeriodSet(uint256 vestingPeriod);
```

CVF-9. FIXED

- **Category** Documentation
- **Source** Swapper.sol

Description It is a good practice to put a comment into an empty block to explain why the block is empty.

Client Comment *We added natspec.*

```
115 receive() external payable {}
```

CVF-10. FIXED

- **Category** Procedural
- **Source** Swapper.sol

Description Specifying a particular compiler version makes it harder to migrate to newer versions.

Recommendation Consider specifying as "^0.8.0".

Client Comment *We updated it to "^0.8.17".*

```
2 pragma solidity 0.8.17;
```

CVF-11. FIXED

- **Category** Procedural
- **Source** EscrowedEXA.sol

Description Specifying a particular compiler version makes it harder to migrate to newer versions.

Recommendation Consider specifying as "^0.8.0".

Client Comment *We updated it to "^0.8.17".*

```
2 pragma solidity 0.8.17;
```

CVF-12. FIXED

- **Category** Documentation
- **Source** EscrowedEXA.sol

Description The number format of this variable is unclear.

Recommendation Consider documenting.

Client Comment *We added natspec.*

```
22 uint256 public reserveRatio;
```

CVF-13. INFO

- **Category** Procedural
- **Source** EscrowedEXA.sol

Description This interface should be defined in a separate file named ”ISablierV2LockupLinear.sol”.

Client Comment *This is a style choice and we are being consistent with other Exactly contracts in which we follow the same code structure.*

```
147 interface ISablierV2LockupLinear {
```

CVF-14. INFO

- **Category** Procedural
- **Source** Swapper.sol

Description This interface should be moved into a separate file named ”IPermit2.sol”.

Client Comment *This is a style choice and we are being consistent with other Exactly contracts in which we follow the same code structure.*

```
132 interface IPermit2 {
```

CVF-15. INFO

- **Category** Procedural
- **Source** Swapper.sol

Description This interface should be moved into a separate file named ”IPool.sol”.

Client Comment *This is a style choice and we are being consistent with other Exactly contracts in which we follow the same code structure.*

```
160 interface IPool {
```

CVF-16. INFO

- **Category** Procedural
- **Source** Swapper.sol

Description We didn't review these files.

```
4 import { WETH, ERC20 } from "solmate/src/tokens/WETH.sol";
  import { SafeTransferLib } from "solmate/src/utils/SafeTransferLib.
     ↪ sol";
```

CVF-17. INFO

- **Category** Procedural
- **Source** EscrowedEXA.sol

Description We didn't review this file.

```
9 import { FixedPointMathLib } from "solmate/src/utils/
     ↪ FixedPointMathLib.sol";
```

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