Real Estate Executive SmarDex

Security Assessment (Summary Report)

April 28, 2023

Prepared for:
Stéphane Ballmer
Real Estate Executive SA

Prepared by: **Gustavo Grieco and Elvis Skoždopolj**

About Trail of Bits

Founded in 2012 and headquartered in New York, Trail of Bits provides technical security assessment and advisory services to some of the world's most targeted organizations. We combine high-end security research with a real-world attacker mentality to reduce risk and fortify code. With 100+ employees around the globe, we've helped secure critical software elements that support billions of end users, including Kubernetes and the Linux kernel.

We maintain an exhaustive list of publications at https://github.com/trailofbits/publications, with links to papers, presentations, public audit reports, and podcast appearances.

In recent years, Trail of Bits consultants have showcased cutting-edge research through presentations at CanSecWest, HCSS, Devcon, Empire Hacking, GrrCon, LangSec, NorthSec, the O'Reilly Security Conference, PyCon, REcon, Security BSides, and SummerCon.

We specialize in software testing and code review projects, supporting client organizations in the technology, defense, and finance industries, as well as government entities. Notable clients include HashiCorp, Google, Microsoft, Western Digital, and Zoom.

Trail of Bits also operates a center of excellence with regard to blockchain security. Notable projects include audits of Algorand, Bitcoin SV, Chainlink, Compound, Ethereum 2.0, MakerDAO, Matic, Uniswap, Web3, and Zcash.

To keep up to date with our latest news and announcements, please follow @trailofbits on Twitter and explore our public repositories at https://github.com/trailofbits. To engage us directly, visit our "Contact" page at https://www.trailofbits.com/contact, or email us at info@trailofbits.com.

Trail of Bits, Inc.
228 Park Ave S #80688
New York, NY 10003
https://www.trailofbits.com
info@trailofbits.com

Notices and Remarks

Copyright and Distribution

Test Coverage Disclaimer

All activities undertaken by Trail of Bits in association with this project were performed in accordance with a statement of work and agreed upon project plan.

Security assessment projects are time-boxed and often reliant on information that may be provided by a client, its affiliates, or its partners. As a result, the findings documented in this report should not be considered a comprehensive list of security issues, flaws, or defects in the target system or codebase.

Trail of Bits uses automated testing techniques to rapidly test the controls and security properties of software. These techniques augment our manual security review work, but each has its limitations: for example, a tool may not generate a random edge case that violates a property or may not fully complete its analysis during the allotted time. Their use is also limited by the time and resource constraints of a project.

Table of Contents

Executive Summary

Engagement Overview

Real Estate Executive SA engaged Trail of Bits to review the security of its SmarDex automated market maker (AMM). From April 10 to April 14, 2023, a team of two consultants conducted a security review of the client-provided source code, with two person-weeks of effort. Details of the project's timeline, test targets, and coverage are provided in subsequent sections of this report.

Project Scope

Our testing efforts were focused on the identification of flaws that could result in a compromise of integrity or availability of the target system. We conducted this audit with full knowledge of the target system, including access to the source code and white paper. We performed manual code review and dynamic testing of the target system.

Summary of Findings

The audit uncovered significant flaws or defects that could impact system confidentiality, integrity, or availability.

EXPOSURE ANALYSIS

Severity	*Count*
High	2
Medium	3
Low	0
Informational	2
Undetermined	0

CATEGORY BREAKDOWN

Category	*Count*
Auditing and Logging	1
Data Validation	4
Timing	1
Undefined Behavior	1

Project Summary

Contact Information

The following managers were associated with this project:

Dan Guido, Account Manager
dan@trailofbits.com

Brooke Langhorne, Project Manager
brooke.langhorne@trailofbits.com

The following engineers were associated with this project:

Gustavo Grieco, Consultant
gustavo.grieco@trailofbits.com

Elvis Skoždopolj, Consultant
elvis.skozdopolj@trailofbits.com

Project Timeline

The significant events and milestones of the project are listed below.

Date	Event
April 10, 2023	Pre-project kickoff call
April 19, 2023	Delivery of report draft
April 19, 2023	Report readout meeting
April 28, 2023	Delivery of final report

Project Goals

The engagement was scoped to provide a security assessment of Real Estate Executive's SmarDex AMM. Specifically, we sought to answer the following non-exhaustive list of questions:

- Are proper access controls used in the contracts that users interact with?
- Could one user drain the pair reserves?
- Are any of the components vulnerable to price manipulation?
- Could an attacker realize a profit through illicit arbitrage?
- Could a user's funds be frozen or stolen?
- Are events used appropriately?

Project Targets

The engagement involved a review and testing of the following target.

SmarDex smart contracts

Repository	https://github.com/SmarDex-Dev/smart-contracts
Version	`3ee81dfe361fded4487f18203576c2c36872e193`
Type	Solidity
Platform	Ethereum

Project Coverage

This section provides an overview of the analysis coverage of the review, as determined by our high-level engagement goals. Our approaches include the following:

- SmarDex: This is an automated market maker (AMM) with a similar interface to that of Uniswap v2. It allows users to make instantaneous swaps between tokens, adjusting the price using a new algorithm, as described in the SmarDex white paper. We manually reviewed the SmarDex contracts to ensure that all the bookkeeping is consistent and that the calculations are sound, including potential over- and underflow issues, rounding issues, and corner cases. We reviewed how third-party contracts can use the SmarDex API, either through the router or by directly interfacing with the swapping pairs. We also used smart contract fuzzing to ensure that important invariants hold—in particular, ensuring that important operations will not revert and verifying that the price computation cannot be misused.

- SmarDex ERC-20 token and reward contracts: These contracts allow users to stake their SmarDex tokens to profit from the interactions of other users with the swapping pairs. We manually reviewed these contracts to make sure the tokens cannot be stolen or frozen and examined the expected interactions regarding the use of shares. We also reviewed how the reward contracts interact with the SmarDex pair to swap other tokens into the SmarDex ERC-20 token.

Coverage Limitations

Because of the time-boxed nature of testing work, it is common to encounter coverage limitations. During this project, we were unable to perform comprehensive testing of the following system elements, which may warrant further review:

- The reward contracts received less coverage due to our focus on the SmarDex core/peripheral contracts.

- We did not review the coherence, consistency, or clarity of the white paper.

- Math, SafeCast, Strings, and Bytes libraries were not reviewed, as they were assumed to be correct.

- We did not review the high-level economic incentives and disincentives imposed by the system.

- We did not report previously known centralization issues.

Codebase Maturity Evaluation

Trail of Bits uses a traffic-light protocol to provide each client with a clear understanding of the areas in which its codebase is mature, immature, or underdeveloped. Deficiencies identified here often stem from root causes within the software development life cycle that should be addressed through standardization measures (e.g., the use of common libraries, functions, or frameworks) or training and awareness programs.

Category	Summary	Result
Arithmetic	Overall, the codebase relies on complex arithmetic. Most of the arithmetic-related complexity is in the computation of the swapping values. While this aspect has been documented, the code does not properly state how invariants are enforced. Additionally, while there are many tests to check the arithmetic, the use of a fuzzer—or a similar automated testing method—is missing. Using a fuzzer would increase confidence in the arithmetic operations. For example, a fuzzer could stress test the tokens swapped over time and ensure that there are no unexpected profits. Additionally, for operations that lead to a loss of precision, we recommend including additional documentation to indicate that the rounding direction does not benefit users.	**Further Investigation Required**
Auditing	While there are events for critical components to monitor the system off-chain, there is a lack of documentation on how to do so. Additionally, there is no incident response plan describing how the protocol's actors must react in case of failure.	**Moderate**
Authentication / Access Controls	The system defines components that are either tightly secured with an owner account or decentralized. The roles of liquidity provider and user performing swaps cannot interfere with each other, except during the expected arbitrage opportunities.	**Strong**
Complexity Management	The code is well structured with the use of core and peripheral contracts and internal libraries. The functions are short and well identified. The code follows the	**Strong**

	Uniswap v2 structure and adheres to the same principles in terms of expected interactions and interfaces.	
Decentralization	While SmarDex contracts are intended to be decentralized, the implementation of the swap pair factories and reward contracts still allows the owner to control the fees. This is known to the Real Estate Executive SA team, and they are committed to relinquishing ownership once they are confident the code works as expected and does not contain any security vulnerabilities.	Satisfactory
Documentation	SmarDex AMM features a new algorithm to enforce fair swapping and mitigate impermanent loss. This is described in detail in the white paper, which includes examples on how it works. The code also features some inline documentation; however, it was not enough for a full review of the code. This documentation must be expanded to ensure future audits can effectively review the code. Finally, documentation for users to safely integrate their smart contracts with SmarDex is missing and could be the cause of security issues if the users are not careful.	Moderate
Transaction Ordering Risks	SmarDex implements all the checks and mitigations for known transaction ordering risks (e.g., sandwich attacks) in the router contract code. However, documentation for integrating the swapping pair directly into third-party smart contracts should be created to ensure users are aware of the risks.	Moderate
Low-Level Manipulation	While the code includes a small number of optimizations using low-level operands (e.g., bit shifting) and assembly snippets, their use is limited and can be understood without extra documentation.	Satisfactory
Testing and Verification	The codebase features an extensive number of unit tests, especially for the math-related code. Although unit tests have good source coverage, they do not cover certain	Moderate

	corner cases; as a result, we encountered code that reverts unexpectedly. In addition, there are no fuzz tests to ensure users cannot cheat or misuse the price computation during mints, burns, or swaps. The fuzz tests should be carefully defined since there are expected arbitrages and interactions that are difficult to formalize. These must be carefully considered to avoid introducing false positives.	

Summary of Findings

The table below summarizes the findings of the review, including type and severity details.

ID	Title	Type	Severity
1	First depositor can steal tokens by manipulating the share price	Timing	Medium
2	_transferFromWithAllowance can fail silently	Data Validation	Informational
3	Owner can multiply accrued campaign rewards	Data Validation	High
4	Insufficient event generation	Auditing and Logging	Informational
5	Sending tokens to the pair contract can block swapping or withdrawing liquidity	Data Validation	High
6	Lack of documentation on smart contract integration with swapping pairs	Undefined Behavior	Medium
7	Very large, small, or unbalanced mints or swaps can trigger unexpected reverts	Data Validation	Medium

A. Vulnerability Categories

The following tables describe the vulnerability categories, severity levels, and difficulty levels used in this document.

Vulnerability Categories	
Category	Description
Access Controls	Insufficient authorization or assessment of rights
Auditing and Logging	Insufficient auditing of actions or logging of problems
Authentication	Improper identification of users
Configuration	Misconfigured servers, devices, or software components
Cryptography	A breach of system confidentiality or integrity
Data Exposure	Exposure of sensitive information
Data Validation	Improper reliance on the structure or values of data
Denial of Service	A system failure with an availability impact
Error Reporting	Insecure or insufficient reporting of error conditions
Patching	Use of an outdated software package or library
Session Management	Improper identification of authenticated users
Testing	Insufficient test methodology or test coverage
Timing	Race conditions or other order-of-operations flaws
Undefined Behavior	Undefined behavior triggered within the system

Severity Levels

Severity	Description
Informational	The issue does not pose an immediate risk but is relevant to security best practices.
Undetermined	The extent of the risk was not determined during this engagement.
Low	The risk is small or is not one the client has indicated is important.
Medium	User information is at risk; exploitation could pose reputational, legal, or moderate financial risks.
High	The flaw could affect numerous users and have serious reputational, legal, or financial implications.

Difficulty Levels

Difficulty	Description
Undetermined	The difficulty of exploitation was not determined during this engagement.
Low	The flaw is well known; public tools for its exploitation exist or can be scripted.
Medium	An attacker must write an exploit or will need in-depth knowledge of the system.
High	An attacker must have privileged access to the system, may need to know complex technical details, or must discover other weaknesses to exploit this issue.

B. Code Maturity Categories

The following tables describe the code maturity categories and rating criteria used in this document.

Code Maturity Categories

Category	Description
Arithmetic	The proper use of mathematical operations and semantics
Auditing	The use of event auditing and logging to support monitoring
Authentication / Access Controls	The use of robust access controls to handle identification and authorization and to ensure safe interactions with the system
Complexity Management	The presence of clear structures designed to manage system complexity, including the separation of system logic into clearly defined functions
Cryptography and Key Management	The safe use of cryptographic primitives and functions, along with the presence of robust mechanisms for key generation and distribution
Decentralization	The presence of a decentralized governance structure for mitigating insider threats and managing risks posed by contract upgrades
Documentation	The presence of comprehensive and readable codebase documentation
Front-Running Resistance	The system's resistance to front-running attacks
Low-Level Manipulation	The justified use of inline assembly and low-level calls
Testing and Verification	The presence of robust testing procedures (e.g., unit tests, integration tests, and verification methods) and sufficient test coverage

Rating Criteria

Rating	Description
Strong	No issues were found, and the system exceeds industry standards.
Satisfactory	Minor issues were found, but the system is compliant with best practices.
Moderate	Some issues that may affect system safety were found.
Weak	Many issues that affect system safety were found.
Missing	A required component is missing, significantly affecting system safety.
Not Applicable	The category is not applicable to this review.
Not Considered	The category was not considered in this review.
Further Investigation Required	Further investigation is required to reach a meaningful conclusion.

C. Fix Review Results

On April 24 2023, Trail of Bits reviewed the fixes and mitigations implemented by the Real Estate Executive SA team for the issues identified in this report.

We reviewed each of the fixes to determine their effectiveness in resolving the associated issues. For additional information, see the Detailed Fix Log.

ID	Title	Severity	Status
1	First depositor can steal tokens by manipulating the share price	Medium	Resolved
2	_transferFromWithAllowance can fail silently	Informational	Unresolved
3	Owner can multiply accrued campaign rewards	High	Resolved
4	Insufficient event generation	Informational	Unresolved
5	Sending tokens to the pair contract can block swapping or withdrawing liquidity	High	Unresolved
6	Lack of documentation on smart contract integration with swapping pairs	Medium	Unresolved
7	Very large, small, or unbalanced mints or swaps can trigger unexpected reverts	Medium	Unresolved

Detailed Fix Log

TOB-SDEX-1: First depositor can steal tokens by manipulating the share price
Resolved in PR #5. The Real Estate Executive SA team has added a minimum number of shares that are burned during the first deposit to mitigate share price manipulation attacks. The Real Estate Executive team has provided the following context:

> *This bug cannot be exploited on existing contracts deployed on Ethereum. However, it could be possible on future chains. We resolved this bug and it has been reviewed by ToB. The result will be in a dedicated PR in the public code.*

TOB-SDEX-2: _transferFromWithAllowance can fail silently
Unresolved. The Real Estate Executive team has provided the following context:

> *No resolution planned as there is nothing to resolve. If someone reuses our contract, they should be cautious.*

TOB-SDEX-3: Owner can multiply accrued campaign rewards
Resolved in PR #4. The Real Estate Executive team has removed the `upgradePrecision` function that contained the issue. The Real Estate Executive team has provided the following context:

> *We decided to resolve this bug and it has been reviewed by ToB. The result will be in a dedicated PR in the public code.*

TOB-SDEX-4: Insufficient event generation
Unresolved. The Real Estate Executive team has provided the following context:

> *No resolution planned as there is nothing to resolve.*

TOB-SDEX-5: Sending tokens to the pair contract can block swapping or withdrawing liquidity
Unresolved. The Real Estate Executive team has provided the following context:

> *No resolution planned since we don't consider this a serious problem, nor does any DEX for that matter.*

TOB-SDEX-6: Lack of documentation on smart contract integration with swapping pairs
Unresolved. The Real Estate Executive team has provided the following context:

> *No resolution planned since we don't consider this a serious problem, but we will add some documentation to encourage people to build on our protocol in which we will explain to developers how to use our contracts correctly to avoid them making a mistake.*

TOB-SDEX-7: Very large, small, or unbalanced mints or swaps can trigger unexpected reverts
Unresolved. The Real Estate Executive team has provided the following context:

> *No resolution planned as there is nothing to resolve, but we will add some documentation to encourage people to build on our protocol. The decision to not include too many error messages makes the contract use less gas and allows user to save on fees. We consider this, therefore, as feature and not a bug. We will explain to developers how to use our contracts correctly, what type of error could correspond to a specific issue, and most importantly, how to verify the range of numbers that can be used in the contract beforehand.*