DeHacker

Code Security Assessment

Fabwelt

July 2 7 th, 2023

Contents

Summary

DeHacker's objective was to evaluate the repository for security-related issues, code quality, and adherence to specification and best practices.
Possible issues we looked for included (but are not limited to):

- Transaction-ordering dependence
- Timestamp dependence
- Mishandled exceptions and call stack limits
- Unsafe external calls
- Integer overflow/underflow
- Number rounding errors
- Reentrancy and cross-function vulnerabilities
- Denial of service/logical oversights
- Access control
- Centralization of power
- Business logic contradicting the specification
- Code clones, functionality duplication
- Gas usage
- Arbitrary token minting

Issue Categories

Every issue in this report was assigned a severity level from the following:

Critical severity issues

A vulnerability that can disrupt the contract functioning in a number of scenarios or creates a risk that the contract may be broken.

Major severity issues

A vulnerability that affects the desired outcome when using a contract or provides the opportunity to use a contract in an unintended way.

Medium severity issues

A vulnerability that could affect the desired outcome of executing the contract in a specific scenario.

Minor severity issues

A vulnerability that does not have a significant impact on possible scenarios for the use of the contract and is probably subjective.

Informational

A vulnerability that has informational character but is not affecting any of the code.

Overview

Project Summary

Project Name	Fabwelt
Platform	Polygon (MATIC)
Website	https://www.fabwelt.com/
Type	Dex
Language	Solidity

Vulnerability Summary

Vulnerability Level	Total	Pending	Declined	Acknowledged	Partially Resolved	Resolved
Critical	0	0	0	0	0	0
Major	1	0	0	0	0	1
Medium	0	0	0	0	0	0
Minor	0	0	0	0	0	0
Informational	3	0	0	3	0	0
Discussion	0	0	0	0	0	0

Audit scope

ID	File	SHA256 Checksum
TCK	fabweitToken.sol	3137dcd9f3a3bf7da610f952b2c3b3dada336c3d2814e98b9301591fcd422273
CON	contracts	

Findings

ID	Category	Severity	Status
TCK-01	Centralization /Privileg	Major	Resolved
TCK-02	Language Specific	Informational	Acknowledged
TCK-04	Coding Style	Informational	Acknowledged
TCK-08	Coding Style	Informational	Acknowledged

MAJOR

TCK-01|CENTRALIZATION RISK

Category	Severity	Location	Status
Centralization / Privilege	Major	fabweitToken.sol: 432, 441, 598, 607, 620, 624	Resolved

Description

n the contracts Ownable and FabweltToken , the role _owner has the authority over the following functions:
Ownable.renounceOwnership() , which renounces the owner role and disables all functions with the onlyOwnermodifier;
Ownable.transferOwnership() , which transfers the owner role to another address;
FabweltToken.excludeAccount() , which excludes an address's tokens for rate calculations;
FabweltToken.includeAccount() , which includes an excluded address's tokens for rate calculations;
FabweltToken.setAsCharityAccount() , which sets the address for FeeAddress ;
FabweltToken.updateFee() , which decides the fees for transfers.

For the contract deployed at 0x23e8b6a3f6891254988b84da3738d2bfe5e703b9 on Polygon, the _owner is 0x63401aac2469bfe676d134571defe64839c35a61, which is an EOA (externally owned account).

Any compromise to the _owner account may allow the hacker to take advantage of this and disrupt how the token shouldoperate .

Recommendation

We advise the client to carefully manage the _owner account's private key to avoid any potential risks of being hacked. Ingeneral, we strongly recommend centralized privileges or roles in the protocol to be improved via a decentralized mechanismor smart-contract-based accounts with enhanced security practices, e.g., Multisignature wallets.

Here are some feasible suggestions that would also mitigate this risk in the short-term and long-term:

A time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key;
Introduction of a DAO/governance/voting module to increase transparency and user involvement.

INFORMATIONAL

TCK-02|UNLOCKED COMPILER VERSION

Category	Severity	Location	Status
Language Specific	Informational	fabweitToken. sol:11	Acknowledged

Description

The contract has an unlocked compiler version. An unlocked compiler version in the source code of the contract permits theuser to compile it at or above a particular version. This, in turn, leads to differences in the generated bytecode betweencompilations due to differing compiler version numbers. This can lead to ambiguity when debugging as compiler-specificbugs may occur in the codebase that would be hard to identify over a span of multiple compiler versions rather than aspecific one .

Recommendation

We advise that the compiler version is instead locked at the lowest version possible that the contract can be compiled at. Forexample, for version v0.8.2 the contract should contain the following line:

```
pragma solidity 0.8.2;
```

INFORMATIONAL

TCK-04|LACK OF EVENT EMISSIONS FOR SIGNIFICANTTRANSACTIONS

Category	Severity	Location	Status
Coding Style	Informational	fabweitToken.sol: 572, 598, 607, 620, 624	Acknowledged

Description

The following functions update crucial state variables. Events should be emitted to log these updates.

FabweltToken.deliver()
FabweltToken.excludeAccount()
FabweltToken.includeAccount()
FabweltToken.setAsCharityAccount()
FabweltToken.updateFee()

Recommendation

We advise adding events for sensitive actions in the aforementioned functions and emitting them in the correspondingfunctions.

INFORMATIONAL

TCK-08 MISSING ERROR MESSAGES

Category	Severity	Location	Status
Coding Style	Informational	fabweitToken.sol: 625	Acknowledged

Description

The require statement can be used to check for conditions and throw an exception if the condition is not met. It is better toprovide a string message containing details about the error that will be passed back to the caller.

Recommendation

We advise refactoring the linked codes as below:

```
625         require(_txFee < 100 && _stakeFee < 100 && _charityFee < 100, "Invalid
fee rates");
```

Disclaimer

This report is based on the scope of materials and documentation provided for a limited review at the time provided. Results may not be complete nor inclusive of all vulnerabilities. The review and this report are provided on an as-is, where-is, and as-available basis. You agree that your access and/or use, including but not limited to any associated services, products, protocols, platforms, content, and materials, will be at your sole risk. Blockchain technology remains under development and is subject to unknown risks and flaws. The review does not extend to the compiler layer, or any other areas beyond the programming language, or other programming aspects that could present security risks. A report does not indicate the endorsement of any particular project or team, nor guarantee its security. No third party should rely on the reports in any way, including for the purpose of making any decisions to buy or sell a product, service or any other asset. To the fullest extent permitted by law, we disclaim all warranties, expressed or implied, in connection with this report, its content, and the related services and products and your use thereof, including, without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement. We do not warrant, endorse, guarantee, or assume responsibility for any product or service advertised or offered by a third party through the product, any open source or third-party software, code, libraries, materials, or information linked to, called by, referenced by or accessible through the report, its content, and the related services and products, any hyperlinked websites, any websites or mobile applications appearing on any advertising, and we will not be a party to or in any way be responsible for monitoring any transaction between you and any third-party providers of products or services. As with the purchase or use of a product or service through any medium or in any environment, you should use your best judgment and exercise caution where appropriate.

FOR AVOIDANCE OF DOUBT, THE REPORT, ITS CONTENT, ACCESS, AND/OR USAGE THEREOF, INCLUDING ANY ASSOCIATED SERVICES OR MATERIALS, SHALL NOT BE CONSIDERED OR RELIED UPON AS ANY FORM OF FINANCIAL, INVESTMENT, TAX, LEGAL, REGULATORY, OR OTHER ADVICE.

Appendix

Finding Categories

Centralization / Privilege

Centralization / Privilege findings refer to either feature logic or implementation of components that act against the nature of decentralization, such as explicit ownership or specialized access roles in combination with a mechanism to relocate funds.

Coding Style

Coding Style findings usually do not affect the generated bytecode but rather comment on how to make the codebase more legible and, as a result, easily maintainable.

Volatile Code

Volatile Code findings refer to segments of code that behave unexpectedly on certain edge cases that may result in a vulnerability.

Logical Issue

Logical Issue findings detail a fault in the logic of the linked code, such as an incorrect notion on how block. timestamp works.

Checksum Calculation Method

The "Checksum" field in the "Audit Scope" section is calculated as the SHA-256 (Secure Hash Algorithm 2 with digest size of 256 bits) digest of the content of each file hosted in the listed source repository under the specified commit.

The result is hexadecimal encoded and is the same as the output of the Linux “sha256sum" command against the target file.

About

DeHacker is a team of auditors and white hat hackers who perform security audits and assessments. With decades of experience in security and distributed systems, our experts focus on the ins and outs of system security. Our services follow clear and prudent industry standards. Whether it's reviewing the smallest modifications or a new platform, we'll provide an in-depth security survey at every stage of your company's project. We provide comprehensive vulnerability reports and identify structural inefficiencies in smart contract code, combining high-end security research with a real-world attacker mindset to reduce risk and harden code.

BLOCKCHAIINS

- Ethereum
- Cosmos
- Eos
- Substrate

TECH STACK

- Python
- Solidity
- Rust
- c++

CONTACTS

- https://dehacker.io
- https://twitter.com/dehackerio
- https://github.com/dehacker/audits_public
- https://t.me/dehackerio
- https://blog.dehacker.io/

DeHacker

July 2023