DeHacker

Code Security Assessment

FortKnoxster - Audit

July 2 1 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	FortKnoxster - Audit
Platform	FortKnoxster
Website	https://fortknoxster.com/
Type	Bridge
Language	Solidity

Vulnerability Summary

Vulnerability Level	**Total**	Mitigated	**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	2	0	0	1	0	1
Informational	0	0	0	1	0	0
Discussion	0	0	0	0	0	0

Audit scope

ID	File	SHA256 Checksum
DFP	contracts/policy/DieFiPolicy.sol	89c958cef4c0ad35aaa4c44b2cf348aca92 fea7d7d8ba66f862a433aa55c69a2
DOD	contracts/metatx/ DieFiForwarder.sol	898de496654f2fe40315622b83d49f9e4c 192d469b5ced701d86627c0c10365e

Findings

ID	Category	Severity	Status
DFP-01	Volatile Code	Minor	Acknowledged
FKB-01	Centralization /Privilege	Major	Resolved
FKB-02	Volatile Code	Minor	Resolved

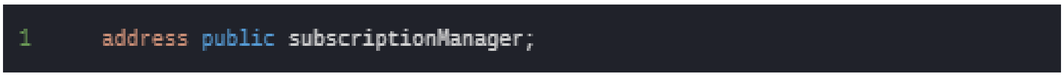
MINOR

DFP-01|THIRD PARTY DEPENDENCY

Category	Severity	Location	Status
Volatile Code	Minor	contracts/policy/ DieFiPolicy.sol: 29, 66~72, 89, 93, 97	Acknowledged

Description

The contract is serving as the underlying entity to interact with one or more third party protocols. The scope of the audit treatsthird party entities as black boxes and assumes their functional correctness. However, in the real world, third parties can becompromised and this may lead to lost or stolen assets. In addition, upgrades of third parties can create severe impacts,such as increasing fees of third parties, migrating to new LP pools, etc.



```
address public subscriptionManager;
```

The contract DieFiPolicy interacts with third party contract with ISubscriptionManager interface viasubscriptionManager .

Recommendation

We understand that the business logic requires interaction with the third parties. We encourage the team to constantlymonitor the statuses of third parties to mitigate the side effects when unexpected activities are observed.

MAJOR

FKB-01|CENTRALIZATION RISKS IN DieFiForwarder.sol ANDDieFiPolicy.sol

Category	Severity	Location	Status
Centralization /Privilege	Major	contracts/metatx/ DieFiForwarder.sol: 52, 86, 93; contracts/ policy/DieFiPolicy.sol: 37	Resolved

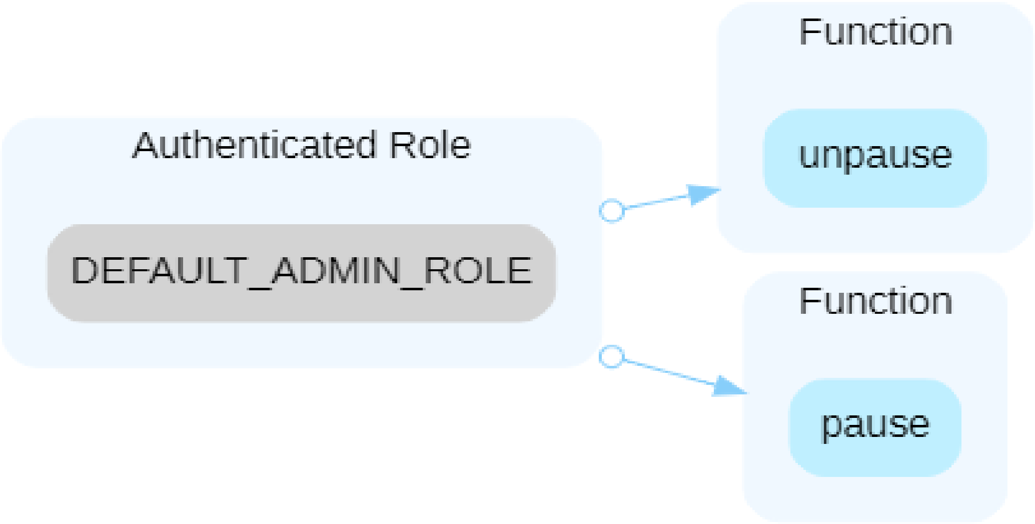
Description

In the contract DieFiForwarder the role DEFAULT_ADMIN_ROLE has authority over the functions shown
in the diagram below and the following functions:
grantRole()
revokeRole()

Any compromise to the DEFAULT_ADMIN_ROLE account may allow the hacker to take advantage of this authority and:

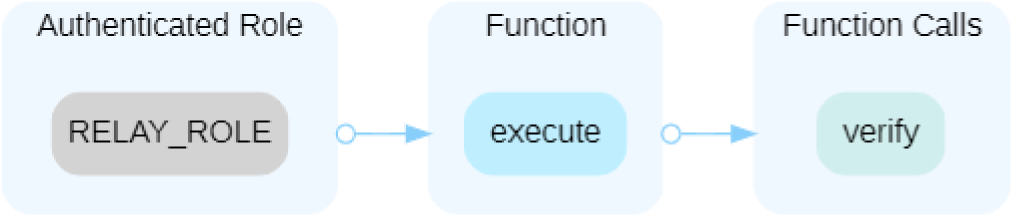
Pause or unpause the function execute() from executing meta transactions by the RELAY_ROLE address, withfunctions pause() and unpause() .
Grant or revoke the RELAY_ROLE to/from any address with functions grantRole() and revokeRole() . A

n any



In the contract DieFiForwarder the role RELAY_ROLE has authority over the functions shown in the diagram below. Anycompromise to the RELAY_ROLE account may allow the hacker to take advantage of this authority and:

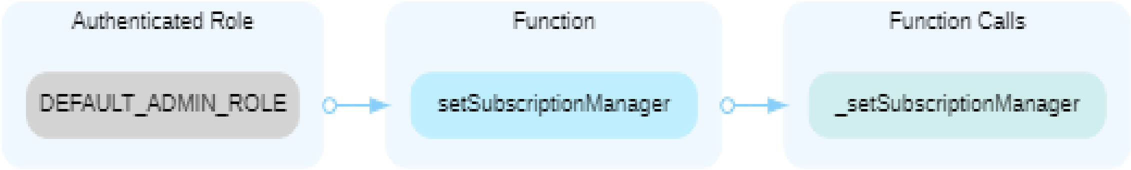
Verify a signature against the given ForwardRequest variables and execute the meta transaction in place of thesignature signer.



In the contract DieFiPolicy the role DEFAULT_ADMIN_ROLE has authority over the functions shown in the diagram belowand the following functions:
grantRole()
revokeRole()

Any compromise to the DEFAULT_ADMIN_ROLE account may allow the hacker to take advantage of this authority and:

Grant or revoke the DEFAULT_ADMIN_ROLE to or from any address. The DEFAULT_ADMIN_ ROLE has the ability to callthe function described below.
Change the subscriptionManager address to any address with function setSubscriptionManager() . The subscriptionManager address is used throughout the contract to call external policy functions.



Recommendation

The risk describes the current project design and potentially makes iterations to improve in the security operation and level ofdecentralization, which in most cases cannot be resolved entirely at the present stage. We advise the client to carefullymanage the privileged account's private key to avoid any potential risks of being hacked. In general, we strongly recommendcentralized privileges
or roles in the protocol be improved via a decentralized mechanism or smart-contract-based accountswith enhanced security practices, e.g., multisignature wallets. Indicatively, here are some

feasible suggestions that wouldalso mitigate the potential risk at a different level in terms of short-term, long-term and permanent:

Short Term:
Timelock and Multi sign (2/5, 3/5) combination mitigate by delaying the sensitive operation and avoiding a single point of keymanagement

Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
AND
Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private keycompromised;
AND
A medium/blog link for sharing the timelock contract and multi-signers addresses information with the publicaudience.

Long Term:
Timelock and DAO, the combination, mitigate by applying decentralization and transparency.
Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
AND
Introduction of a DAO/governance/voting module to increase transparency and user involvement.
AND
A medium/blog link for sharing the timelock contract, multi-signers addresses, and DAO information with the publicaudience.

Permanent:
Renouncing the ownership or removing the function can be considered fully resolved.
Renounce the ownership and never claim back the privileged roles.ORRemove the risky functionality.

INFORMATIONAL

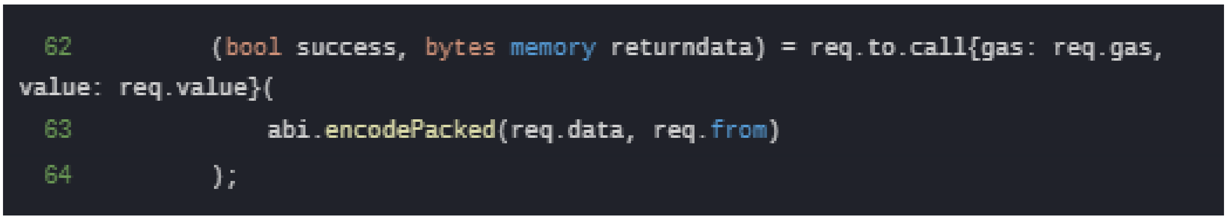
FKB-02|CHECKS EFFECTS INTERACTION PATTERN VIOLATED

Category	Severity	Location	Status
Volatile Code	Minor	contracts/metatx/ DieFiForwarder. sol: 62~64, 78; contracts/policy/DieFiPolicy. sol: 66~72, 74~81	Resolved

Description

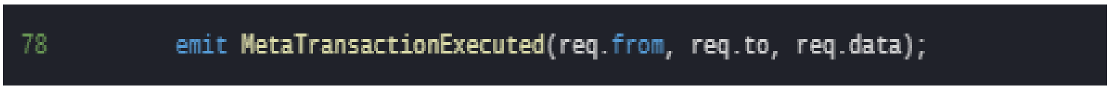
A reentrancy attack can occur when the contract creates a function that makes an external call to another untrusted contractbefore resolving any effects. If the attacker can control the untrusted contract, they can make a recursive call back to theoriginal function, repeating interactions that would have otherwise not run after the external call resolved the effects.If a recursive call is preformed on either function execute() or createPolicy() , the ordering of events emitted will beinaccurate compared to the actual execution timeline of the external calls.

External call



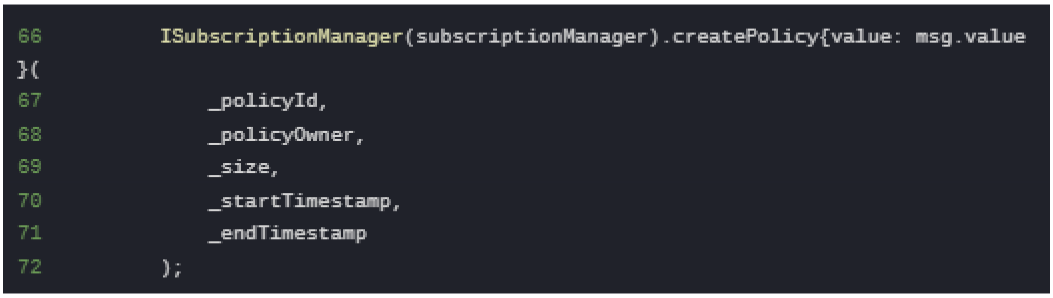
```
 62          (bool success, bytes memory returndata) = req.to.call{gas: req.gas,
value: req.value}(
 63              abi.encodePacked(req.data, req.from)
 64          );
```

Events emitted after the call



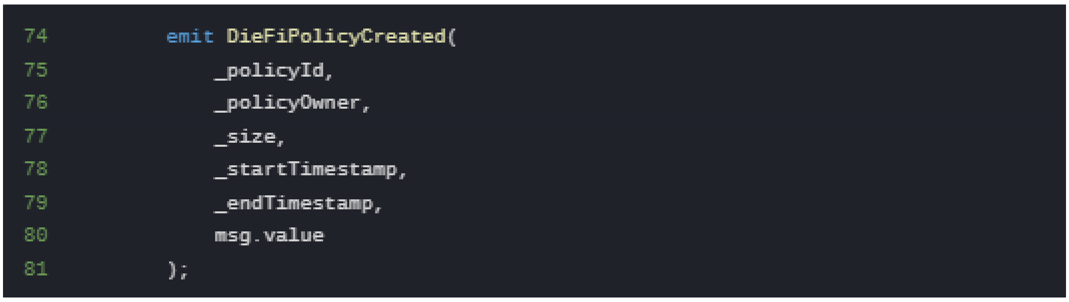
```
78          emit MetaTransactionExecuted(req.from, req.to, req.data);
```

External call



```
66          ISubscriptionManager(subscriptionManager).createPolicy{value: msg.value
}(
67              _policyId,
68              _policyOwner,
69              _size,
70              _startTimestamp,
71              _endTimestamp
72          );
```

Events emitted after the call



```
74          emit DieFiPolicyCreated(
75              _policyId,
76              _policyOwner,
77              _size,
78              _startTimestamp,
79              _endTimestamp,
80              msg.value
81          );
```

Recommendation

We recommend using the Checks-Effects-Interactions Pattern to avoid the risk of calling unknown contracts or applyingOpenZeppelin ReentrancyGuard library - nonReentrant modifier for the aforementioned functions to prevent reentrancyattack.

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

Eos

Cosmos

Substrate

TECH STACK

Python

Rust

Solidity

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