Audit Report

CardFi

December 2022

cardFI_NFT	5c7f9e4126fd5429c8e045b794908c037c48ed31bc25c9876cf68cf91ca92843
cardFi	fb63b3e58e1f78494bd6380d2228ce207fc75bac72861ba4507f1da25b82fd74
Audited by	© cyberscope

Table of Contents

Contract Review

Contract Name	cardFi
Compiler Version	v0.8.17+commit.8df45f5f
Optimization	200 runs
CardFi Test Deploy	https://testnet.bscscan.com/token/0x3b8A71CCAB2D80709AEa78A084D63049f98D0DA1
CardFI_NFT Test Deploy	https://testnet.bscscan.com/token/0x0De146Cb82099BEc981b79a8343aaDA00Ca2A922
Domain	cardfi.co

Audit Updates

Initial Audit	5th December 2022
Corrected	

Source Files

Filename	**SHA256**
cardFI_NFT.sol	5c7f9e4126fd5429c8e045b794908c037c48ed31bc25c9876cf68cf91ca92843
cardFi.sol	fb63b3e58e1f78494bd6380d2228ce207fc75bac72861ba4507f1da25b82fd74
IcardFi.sol	825bbe0a96079b47fb31ac478a98b681b960599bf460cfc80cb02b99bb417472

Introduction

The project consists of two contracts, **cardFi** and **cardFi_NFT**.

The user deposits native currency to receive NFT.

The are two options:

- Pay to receive NFT.
- Pay to receive NFT and lock cardFi tokens to redeem later.

The payment amount, native currency cost, lock period and cardFi amount depends on the card type.

Roles

The projects includes two roles, Admin and User.

Admin

The Admin Role has the authority to:

- Alter NFT card type properties.
- Alter deposit and withdraw fees.

User

The User Role has the authority to:

- Register new currency.
- Register new tokens.
- Deposit native currency to receive NFT.

Contract Diagnostics

● Critical ● Medium ● Minor / Informative

Severity	Code	Description	Status
●	TO	Transactions Overflow	Unresolved
●	STC	Succeeded Transfer Check	Unresolved
●	PTC	Public Token Claim	Unresolved
●	BSM	Bypass Signating Message	Unresolved
●	CO	Code Optimization	Unresolved
●	DSM	Data Structure Misuse	Unresolved
●	RPM	Redundant Payable Methods	Unresolved
●	L04	Conformance to Solidity Naming Conventions	Unresolved
●	L05	Unused State Variable	Unresolved
●	L11	Unnecessary Boolean equality	Unresolved

TO - Transactions Overflow

Criticality	minor / informative
Location	contract/cardFi.sol#L208,238,315
Status	Unresolved

Description

The transactions will overflow if the deposit and/or Royalties[_currency].deposit are over over 100. As a result the transaction will revert.

```
uint256 royalty=(_depositAmount*Royalties[_currency].deposit)/100;
uint256 amount=_depositAmount-royalty;
...
uint256 royalty=(_depositAmount*deposit)/100;
uint256 amount=_depositAmount-royalty;
...
if(_Card.nativeCurrency==true){
    royalty=(_redeemAmount*withdraw)/100;
} else{
    royalty=(_redeemAmount*Royalties[currency].withdraw)/100;
}
uint256 amount=_redeemAmount-royalty;
```

Recommendation

The contract could embody a check for not allowing setting the deposit and Royalties[_currency].deposit more than a 100.

The team should carefully manage the private keys of the owner's account. We strongly recommend a powerful security mechanism that will prevent a single user from accessing the contract admin functions. That risk can be prevented by temporarily locking the contract or renouncing ownership.

STC - Succeeded Transfer Check

Criticality	minor / informative
Location	contract/cardFi.sol#L198,227,292 contract/cardFI_NFT.sol#L62,80,141
Status	Unresolved

Description

According to the ERC20 specification, the transfer methods should be checked if the result is successful. Otherwise, the contract may wrongly assume that the transfer has been established.

```
function deposit_ERC20(IERC721Upgradeable _contractAddress, uint256 _tokenId,
IERC20Upgradeable _currency,  uint256 _depositAmount, uint256 newTime) public
payable {
...
}
...
function deposit_native(IERC721Upgradeable _contractAddress, uint256 _tokenId,
uint256 _depositAmount, uint256 newTime) public payable {
...
}
...
function redeem(IERC721Upgradeable _contractAddress, uint256 _tokenId, uint256
_redeemAmount, uint8 v,  bytes32 r,  bytes32 s, string memory message, address
signerAddress) public payable {
...
}
```

```
function mintNft(uint256 cardTypeNumber, string memory URI) public payable {
...
}
...
function mintNftCustom(uint256 cardTypeNumber, string memory URI) public
payable {
...
}
...
function topUpBalance(uint256 amount) public onlyOwner {
...
}
```

Recommendation

The contract should check if the result of the transfer methods is successful.

PTC - Public Token Claim

Criticality	minor/informative
Location	contract/cardFi.sol#L177
Status	Unresolved

Description

The tokenToNft method is public. So any user can claim _tokenId. As a result, the deposit_native method will not be able to run for these tokenIds. Any user can exploit the public permissions to cap the potential token ids that will be claimed from the deposit_native method.

```
function tokenToNft(IERC721Upgradeable _contractAddress, uint256 _tokenId,
IERC20Upgradeable _currency) public {
...
_Card.ERC20Added=true;
...
}
...
require(_Card.ERC20Added==false, "this NFT has ERC20 attached");
```

Recommendation

The team is advised to carefully check if the implementation follows the expected business logic.

BSM - Bypass Signating Message

Criticality	critical
Location	contract/cardFi.sol#L193
Status	Unresolved

Description

The signature does not contain any unique identification like nonce. Additionally, it receives the signer address as an argument. As a result, the signature can be bypassed by the caller.

```
executeSetIfSignatureMatch(v, r, s, message, signerAddress);
```

Recommendation

The team is advised to carefully check if the implementation follows the expected business logic.

CO - Code Optimization

Criticality	minor / informative
Location	contract/cardFi.sol#L108,181
Status	Unresolved

Description

There are code segments that could be optimized. A segment may be optimized so that it becomes a smaller size, consumes less memory, executes more rapidly, or performs fewer operations. The check signer === sender is performed twice.

```
require(signer == sender, "MyFunction: invalid signature");
...
return signer == sender;
```

This check is redundant as the function tokenToNft is only called when _Card.ERC20Added is equal to false.

```
require(_Card.ERC20Added==false, "this token already has currency assigned");
```

Recommendation

The authors are advised to extract the check (signer == sender) in a variable so that the operation is performed once. The check _Card.ERC20Added==false should be removed.

DSM - Data Structure Misuse

Criticality	minor / informative
Location	contract/cardFi.sol#L151
Status	Unresolved

Description

The contract uses the valuable allowedCrypto as an array. The business logic of the contract does not require to iterate this structure sequentially. Thus, unnecessary loops are produced that increase the required gas.

```
IERC20Upgradeable[] public allowedCrypto;
...
function tokenExist(IERC20Upgradeable tokenAddress) public view returns(bool
ifExist) {
    for (uint256 i = 0; i < allowedCrypto.length; i++) {
        if (allowedCrypto[i] == tokenAddress) {
            return true;
        }
    }
    return false;
}
```

Recommendation

The contract could use a data structure that provides instant access. For instance, a Set or a Map would fit better to the business logic of the contract. This way the time complexity will be reduced from o(n) to o(1).

RPM - Redundant Payable Methods

Criticality	minor / informative
Location	contract/cardFi.sol#L198,292
Status	Unresolved

Description

The contract methods (deposit_ERC20, redeem) are payable, but they do not handle the received native currency (msg.value).

```
function deposit_ERC20(IERC721Upgradeable _contractAddress, uint256 _tokenId,
IERC20Upgradeable _currency,  uint256 _depositAmount, uint256 newTime) public
payable {}
```

```
function redeem(IERC721Upgradeable _contractAddress, uint256 _tokenId, uint256
_redeemAmount, uint8 v,  bytes32 r,  bytes32 s, string memory message, address
signerAddress) public payable {}
```

Recommendation

The authors are advised to remove payable from both methods as it is redundant.

L04 - Conformance to Solidity Naming Conventions

Criticality	minor / informative
Location	contract/IcardFi.sol#L9 contract/cardFi.sol#L66,227,266,167,130,177,28,292,198,260,67,71,13,123,138, 116,68,70,69,144,253 contract/cardFI_NFT.sol#L117,62,26,101,17,13,80,39
Status	Unresolved

Description

Solidity defines a naming convention that should be followed. Rule exceptions:

- Allow constant variable name/symbol/decimals to be lowercase.
- Allow _ at the beginning of the mixed_case match for private variables and unused parameters.

```
deposit_ERC20
newRoyalty
_contractAddress
_tokenId
_paytoken
_depositAmount
setRoyalty_ERC20
Royalties
deposit_native
...
```

Recommendation

Follow the Solidity naming convention.
https://docs.soliditylang.org/en/v0.8.17/style-guide.html#naming-conventions.

L11 - Unnecessary Boolean equality

Criticality	minor / informative
Location	contract/cardFi.sol#L292,198,227,177,167
Status	Unresolved

Description

The comparison to boolean constants is redundant. Boolean constants can be used directly and do not need to be compared to true or false.

```
_Card.nativeCurrency == true
_Card.ERC20Added == false
_Card.nativeCurrency == false
require(bool,string)(_Card.nativeCurrency == false,native currency is already assigned)
require(bool,string)(_Card.ERC20Added == false,this token already has currency assigned)
require(bool,string)(_Card.ERC20Added == false,this NFT has ERC20 attached)
tokenExist(_currency) == false
require(bool,string)(tokenExist(_paytoken) == false,THIS CURRENCY IS ALREADY ADDED)
...
```

Recommendation

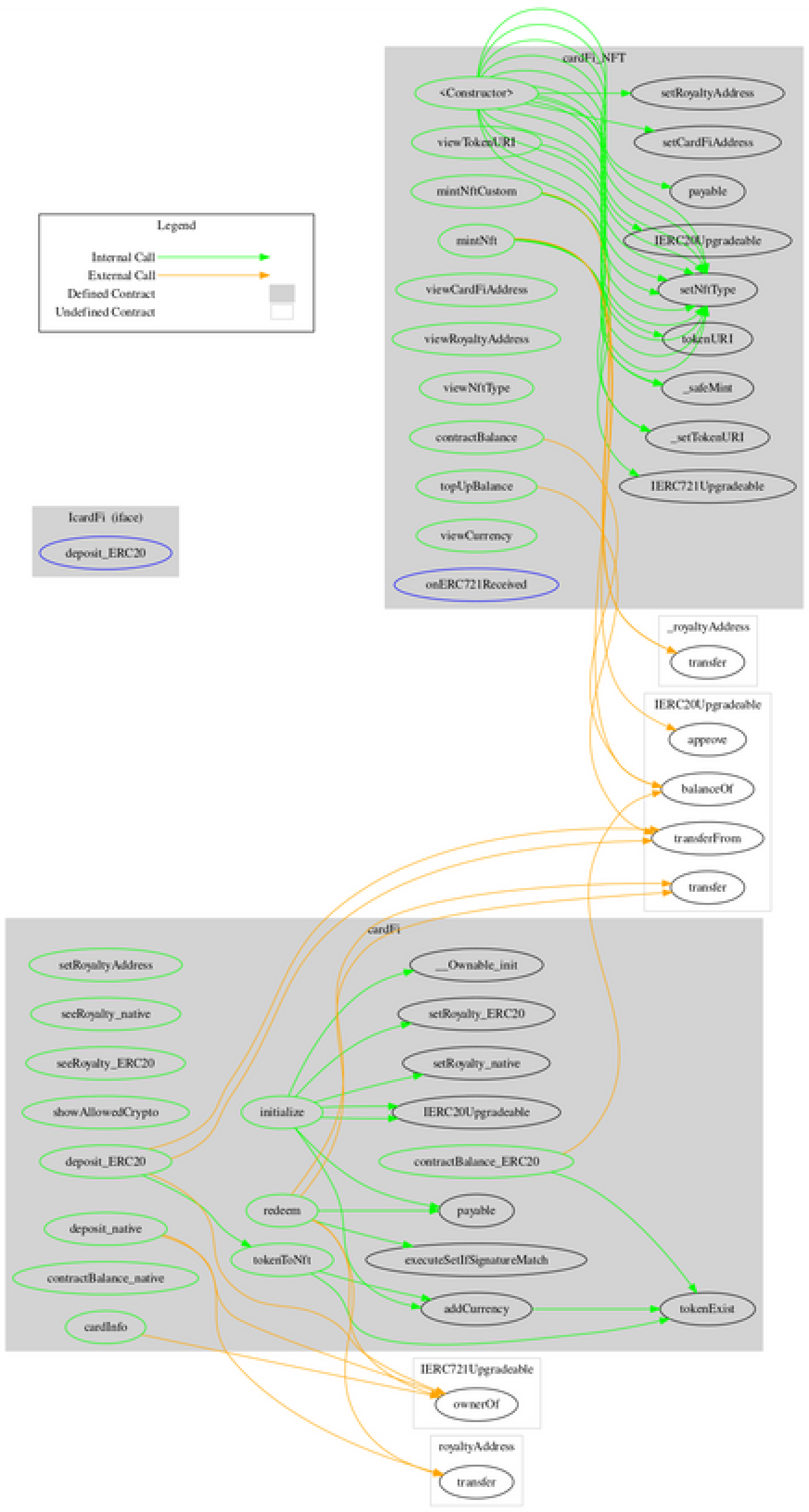
Remove the equality to the boolean constant.

Contract Functions

Contract	Type	Bases		
	Function Name	Visibility	Mutability	Modifiers
cardFi_NFT	Implementation	ERC721URIStorage, IERC721Receiver, Ownable		
	<Constructor>	Public	✓	ERC721
	mintNft	Public	Payable	-
	mintNftCustom	Public	Payable	-
	viewTokenURI	Public		-
	setCardFiAddress	Public	✓	onlyOwner
	viewCardFiAddress	Public		onlyOwner
	setRoyaltyAddress	Public	✓	onlyOwner
	viewRoyaltyAddress	Public		onlyOwner
	setNftType	Public	✓	onlyOwner
	viewNftType	Public		-
	contractBalance	Public		onlyOwner
	topUpBalance	Public	✓	onlyOwner
	viewCurrency	Public		-
	onERC721Received	External		-
cardFi	Implementation	Initializable, OwnableUpgradeable		
	initialize	Public	✓	initializer
	executeSetIfSignatureMatch	Public		-
	setRoyaltyAddress	Public	✓	onlyOwner
	setRoyalty_native	Public	✓	onlyOwner
	setRoyalty_ERC20	Public	✓	onlyOwner
	seeRoyalty_native	Public		onlyOwner
	seeRoyalty_ERC20	Public		onlyOwner
	tokenExist	Public		-

	showAllowedCrypto	Public		-
	addCurrency	Public	✓	-
	tokenToNft	Public	✓	-
	deposit_ERC20	Public	Payable	-
	deposit_native	Public	Payable	-
	contractBalance_ERC20	Public		onlyOwner
	contractBalance_native	Public		onlyOwner
	cardInfo	Public		-
	redeem	Public	Payable	-
IcardFi	Interface			
	deposit_ERC20	External	Payable	-

Contract Flow



Domain Info

Domain Name	cardfi.co
Registry Domain ID	D47284542FEC9472C9A129B2E3F85D44F-GDREG
Creation Date	2022-09-23T03:29:14Z
Updated Date	2022-09-28T03:29:14Z
Registry Expiry Date	2023-09-23T03:29:14Z
Registrar WHOIS Server	whois.godaddy.com
Registrar URL	whois.godaddy.com
Registrar	GoDaddy.com, LLC
Registrar IANA ID	146

The domain was created 2 months before the creation of the audit. It will expire in 10 months.

There is no public billing information, the creator is protected by the privacy settings.

Summary

This audit focused on investigating possible security issues and potential improvements. The audit analysis mentions some concerns regarding the message signing process. Additionally, it depicts some issues that may be produced from the methods public access.

Disclaimer

The information provided in this report does not constitute investment, financial or trading advice and you should not treat any of the document's content as such. This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes nor may copies be delivered to any other person other than the Company without Cyberscope’s prior written consent. This report is not nor should be considered an “endorsement” or “disapproval” of any particular project or team. This report is not nor should be regarded as an indication of the economics or value of any “product” or “asset” created by any team or project that contracts Cyberscope to perform a security assessment. This document does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors' business, business model or legal compliance. This report should not be used in any way to make decisions around investment or involvement with any particular project. This report represents an extensive assessment process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.

Blockchain technology and cryptographic assets present a high level of ongoing risk Cyberscope’s position is that each company and individual are responsible for their own due diligence and continuous security Cyberscope’s goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies and in no way claims any guarantee of security or functionality of the technology we agree to analyze. The assessment services provided by Cyberscope are subject to dependencies and are under continuing development. You agree that your access and/or use including but not limited to any services reports and materials will be at your sole risk on an as-is where-is and as-available basis Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty. The assessment reports could include false positives false negatives and other unpredictable results. The services may access and depend upon multiple layers of third parties.

About Cyberscope

Cyberscope is a blockchain cybersecurity company that was founded with the vision to make web3.0 a safer place for investors and developers. Since its launch, it has worked with thousands of projects and is estimated to have secured tens of millions of investors' funds.

Cyberscope is one of the leading smart contract audit firms in the crypto space and has built a high-profile network of clients and partners.

The Cyberscope team

https://www.cyberscope.io