Competitive Security Assessment

Capsid NFR Trading

Feb 15th, 2023

Secure3

secure3.io

Summary

This report is prepared for the project to identify vulnerabilities and issues in the smart contract source code. A group of NDA covered experienced security experts have participated in the Secure3's Audit Contest to find vulnerabilities and optimizations. Secure3 team has participated in the contest process as well to provide extra auditing coverage and scrutiny of the finding submissions.

The comprehensive examination and auditing scope includes:

- Cross checking contract implementation against functionalities described in the documents and white paper disclosed by the project owner.
- Contract Privilege Role Review to provide more clarity on smart contract roles and privilege.
- Using static analysis tools to analyze smart contracts against common known vulnerabilities patterns.
- Verify the code base is compliant with the most up-to-date industry standards and security best practices.
- Comprehensive line-by-line manual code review of the entire codebase by industry experts.

The security assessment resulted in findings that are categorized in four severity levels: Critical, Medium, Low, Informational. For each of the findings, the report has included recommendations of fix or mitigation for security and best practices.

Overview

Project Detail

Project Name	Capsid NFR Trading
Platform & Language	Solidity
Codebase	• https://github.com/SolitaireNFT/capsid-nfr-trading-contracts • audit commit - 1f02a5e7a9e48c9dbb5030133c271808d53253fd • final commit - f454f5d266d4cd5753b071bbaf58308ed5389573
Audit Methodology	• Audit Contest • Business Logic and Code Review • Privileged Roles Review • Static Analysis

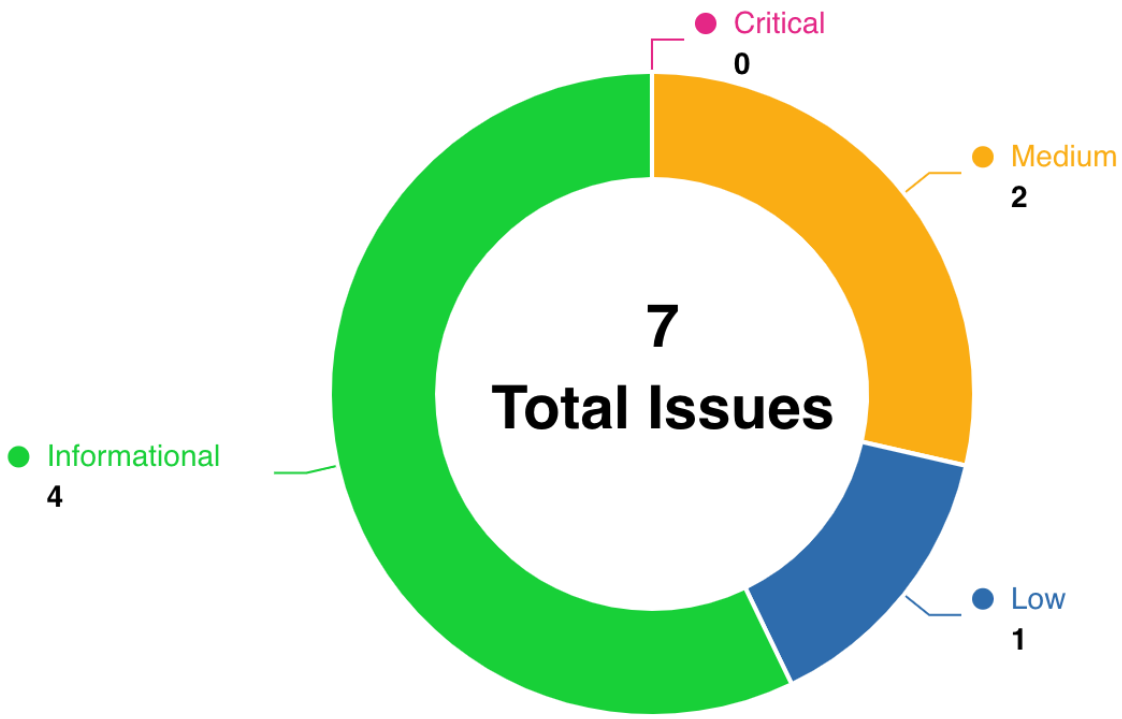
Code Vulnerability Review Summary

Vulnerability Level	Total	Reported	Acknowledged	Fixed	Mitigated	Declined
Critical	0	0	0	0	0	0
Medium	2	0	0	2	0	0
Low	1	0	1	0	0	0
Informational	4	0	1	3	0	0

Audit Scope

File	Commit Hash
contracts/FulfillEventsAndErrors.sol	1f02a5e7a9e48c9dbb5030133c271808d53253fd
contracts/I1155Mint.sol	1f02a5e7a9e48c9dbb5030133c271808d53253fd
contracts/INfrTrade.sol	1f02a5e7a9e48c9dbb5030133c271808d53253fd
contracts/MyProxyAdmin.sol	1f02a5e7a9e48c9dbb5030133c271808d53253fd
contracts/Nfr1155.sol	1f02a5e7a9e48c9dbb5030133c271808d53253fd
contracts/NfrTrade.sol	1f02a5e7a9e48c9dbb5030133c271808d53253fd
contracts/OrderEnums.sol	1f02a5e7a9e48c9dbb5030133c271808d53253fd
contracts/OrderStructs.sol	1f02a5e7a9e48c9dbb5030133c271808d53253fd
contracts/OwnableProxy.sol	1f02a5e7a9e48c9dbb5030133c271808d53253fd

Code Assessment Findings



ID	Name	Category	Severity	Status	Contributor
CNT-1	`Nfr1155` Check the length of the array	Logical	Informational	Fixed	8olidity, helookslike me
CNT-2	`NfrTrade` code does not match documentation	Logical	Informational	Acknowledged	8olidity
CNT-3	Flash loan bypasses the judgment of NFT ownership	Logical	Medium	Fixed	8olidity
CNT-4	Gas Optimizations - Cache Array Length Outside of Loop	Gas Optimization	Informational	Fixed	helookslike me

CNT-5	**No Validation/Whitelist of ERC20 token used for payment**	**Logical**	**Informational**	**Fixed**	**8olidity, porotta**
CNT-6	**Centralization Risk**	**Privilege Related**	**Low**	**Acknowledged**	**8olidity, helookslikeme**
CNT-7	**Use `safeTransferFrom()` function**	**Logical**	**Medium**	**Fixed**	**8olidity**

CNT-1: `Nfr1155` Check the length of the array

Category	Severity	Code Reference	Status	Contributor
Logical	Informational	• code/contracts/Nfr1155.sol#L75-L85	Fixed	8olidity, helookslikeme

Code

```
75:     function batchExpire(address[] memory _froms, uint256[] memory _ids)
76:         public
77:         onlyManager
78:     {
79:         for (uint256 i = 0; i < _froms.length; i++) {
80:             uint256 quantity = super.balanceOf(_froms[i], _ids[i]);
81:             if (quantity > 0) {
82:                 super._burn(_froms[i], _ids[i], quantity);
83:             }
84:         }
85:     }
```

Description

8olidity : In the batchExpire() function, two arrays of _froms and _ids are passed in, but the lengths of these two arrays may be different. If the length of _froms is greater than _ids, then there will be an array out of bounds problem.

```
function batchExpire(address[] memory _froms, uint256[] memory _ids)
    public
    onlyManager
{
    for (uint256 i = 0; i < _froms.length; i++) {
        uint256 quantity = super.balanceOf(_froms[i], _ids[i]);
        if (quantity > 0) {
            super._burn(_froms[i], _ids[i], quantity);
        }
    }
}
```

helookslikeme : When the length of the _froms array is greater than the _ids[] array, there will be an array out of bounds

Recommendation

8olidity : Determine the length of the incoming _froms and _ids arrays

```
    function batchExpire(address[] memory _froms, uint256[] memory _ids)
        public
        onlyManager
    {
        require(_froms.length == _ids.length);
        for (uint256 i = 0; i < _froms.length; i++) {
            uint256 quantity = super.balanceOf(_froms[i], _ids[i]);
            if (quantity > 0) {
                super._burn(_froms[i], _ids[i], quantity);
            }
        }
    }
```

helookslikeme : Check the length of the _froms and _ids arrays, requiring the same length

```
_froms.length == _ids.length
```

Client Response

Fixed.

CNT-2: `NfrTrade` code does not match documentation

Category	Severity	Code Reference	Status	Contributor
Logical	Informational	• code/contracts/NfrTrade.sol#L288	Acknowledged	8olidity

Code

```
288:        if (order.payPrice == 0) {
```

Description

8olidity : https://capsid.gitbook.io/capsid/products/marketplace-for-non-fungible-rights/video-tutorial-for-nfr

In the document, it is required that the price cannot be lower than 0.01ETH, but there is no check in the code

```
function _basicValidateOrder(OrderParameters calldata order) internal pure {
  if (order.signer == address(0)) {
      revert InvalidInputParameters();
  }

  if (order.token == address(0)) {
      revert InvalidInputParameters();
  }

  if (order.payPrice == 0) {              //@audit
      revert InvalidInputParameters();
  }
```

Recommendation

8olidity : In the _basicValidateOrder() function, when the payToken is ETH, it should check that the amount of payPrice is greater than 0.01ETH.

Client Response

Minimum price validation is on the platform's server-side only, as it is a platform rule and not contract-specific.

CNT-3:Flash loan bypasses the judgment of NFT ownership

Category	Severity	Code Reference	Status	Contributor
Logical	Medium	• code/contracts/NfrTrade.sol#L142-L153	Fixed	8olidity

Code

```
142:         if (order.itemType == ItemType.ERC721) {
143:             //721
144:             IERC721 con721 = IERC721(order.token);
145:             if (con721.ownerOf(order.identifier) != offerer) {
146:                 revert NotOwnNFT();
147:             }
148:         } else {
149:             IERC1155 con1155 = IERC1155(order.token);
150:             if (con1155.balanceOf(offerer, order.identifier) == 0) {
151:                 revert NotOwnNFT();
152:             }
153:         }
```

Description

8olidity : In the _fullfill function, it will be judged whether the offerer has NFT

```
if (order.itemType == ItemType.ERC721) {
    //721
    IERC721 con721 = IERC721(order.token);
    if (con721.ownerOf(order.identifier) != offerer) {//@audit
        revert NotOwnNFT();
    }
} else {
    IERC1155 con1155 = IERC1155(order.token);
    if (con1155.balanceOf(offerer, order.identifier) == 0) {
        revert NotOwnNFT();
    }
}
```

But the judgment here is just a simple ownership of NFT, you can use the flash loan to temporarily own NFT to bypass the judgment here

Recommendation

8olidity : The offerer address cannot be a contract address, it must be eoa to participate

Client Response

Fixed. Token ownership verification logic is implemented.

CNT-4:Gas Optimizations - Cache Array Length Outside of Loop

Category	Severity	Code Reference	Status	Contributor
Gas Optimization	Informational	• code/contracts/Nfr1155.sol#L79	Fixed	helookslikeme

Code

```
79:         for (uint256 i = 0; i < _froms.length; i++) {
```

Description

helookslikeme : Cache Array Length Outside of Loop

The impact is Abnormal consumption of gas

Recommendation

helookslikeme :

```
uint256 l = _froms.length;
for (uint256 i = 0; i < l; i++) {
```

Client Response

Fixed.

CNT-5:No Validation/Whitelist of ERC20 token used for payment

Category	Severity	Code Reference	Status	Contributor
Logical	Informational	• code/contracts/NfrTrade.sol#L169	Fixed	8olidity, porotta

Code

```
169:            if (!erc20.transferFrom(recipient, offerer, payAmount)) {
```

Description

8olidity : `order.payToken` should have a whitelist. There is no restriction in the code, and any token can participate. Some tokens without any value can be used to participate in trade.
porotta : There is no validation or a whitelist on which tokens can be allowed as payment
The impact here is that anyone could create a a random ERC20 token and use it as forms of payment

Recommendation

8olidity : Judge or limit the tokens in the order, and only allow specific tokens to participate
porotta : A map of allowed tokens that are accepted for payment should be created unless this is intentional

Client Response

Fixed. A whitelist of ERC20 token addresses has been implemented, restricting transactions to only those originating from addresses on the list.

CNT-6:Centralization Risk

Category	Severity	Code Reference	Status	Contributor
Privilege Related	Low	code/contracts/Nfr1155.sol#L64-L72 • code/contracts/Nfr1155.sol#L75-L85	Acknowledged	8olidity, helookslikeme

Code

```
64:    function mint(
65:        address to,
66:        uint256 amount,
67:        bytes memory data
68:    ) external onlyMintProxy returns (uint256) {
69:        uint256 id = _incrementAndGetTokenId();
70:        _mint(to, id, amount, data);
71:        return id;
72:    }

75:    function batchExpire(address[] memory _froms, uint256[] memory _ids)
76:        public
77:        onlyManager
78:    {
79:        for (uint256 i = 0; i < _froms.length; i++) {
80:            uint256 quantity = super.balanceOf(_froms[i], _ids[i]);
81:            if (quantity > 0) {
82:                super._burn(_froms[i], _ids[i], quantity);
83:            }
84:        }
85:    }
```

Description

8olidity : The administrator can call batchExpire() to destroy any asset of any user, which is a centralization risk

```
function batchExpire(address[] memory _froms, uint256[] memory _ids)
    public
    onlyManager //@audit
{
    for (uint256 i = 0; i < _froms.length; i++) {
        uint256 quantity = super.balanceOf(_froms[i], _ids[i]);
        if (quantity > 0) {
            super._burn(_froms[i], _ids[i], quantity);
        }
    }
}
```

helookslikeme : The mint function has centralized authority. If the project party is hacked, or any malicious behavior occurs, it will be an irreversible security risk Effects can be malicious mint
helookslikeme : Administrators can burn anyone's funds at will

Recommendation

8olidity : High-risk operations should not only be allowed to operate on a single address. If it is a normal function, it is recommended to use multi-signature operations
helookslikeme : The repair method is to use methods such as multi-signature or decentralized permissions to reduce permissions
helookslikeme : Use multi-signature or single user

Client Response

The method is invoked automatically by a server-side scheduler, and the private key is also configured on the server. Additional signatures do not mitigate the risk of a private key leak.

CNT-7:Use `safeTransferFrom()` function

Category	Severity	Code Reference	Status	Contributor
Logical	Medium	• code/contracts/NfrTrade.sol#L169	Fixed	8olidity

Code

```
169:            if (!erc20.transferFrom(recipient, offerer, payAmount)) {
```

Description

8olidity : The USDT's transfer and transferFromfunctions doesn't return a bool, so the call to these functions will revert although the user has enough balance and the Nfrtrade contract won't work, assuming that token is USDT.

Recommendation

8olidity : Use the OpenZepplin's safetransfer and safetransferFrom functions.

Client Response

Fixed. Called safeTransferfrom instead.

Disclaimer

This report is subject to the terms and conditions (including without limitation, description of services, confidentiality, disclaimer and limitation of liability) set forth in the Invoices, or the scope of services, and terms and conditions provided to you ("Customer" or the "Company") in connection with the Invoice. This report provided in connection with the services set forth in the Invoices shall be used by the Company only to the extent permitted under the terms and conditions set forth in the Invoice. 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 Secure3's prior written consent in each instance.

This report is not an "endorsement" or "disapproval" of any particular project or team. This report is not an indication of the economics or value of any "product" or "asset" created by any team or project that contracts Secure3 to perform a security assessment. This report does not provide any warranty or guarantee of free of bug of codes analyzed, nor do they provide any indication of the technologies, business model or legal compliancy.

This report should not be used in any way to make decisions around investment or involvement with any particular project. Instead, it represents an extensive assessing process intending to help our customers increase the quality of their code and high-level consistency of implementation and business model, while reducing the risk presented by cryptographic tokens and blockchain technology.

Secure3's position on the final decisions over blockchain technologies and corresponding associated transactions is that each company and individual are responsible for their own due diligence and continuous security.

The assessment services provided by Secure3 is subject to dependencies and under continuing development. 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.