SALUS SECURITY

MAY 2023

CODE SECURITY ASSESSMENT

ORIGIN

Overview

Project Summary

- Name: Origin
- Platform: Ethereum
- Language: Solidity
- Audit Range: See Appendix - 1

Project Dashboard

Application Summary

Name	Origin
Version	v2
Type	Solidity
Dates	May 17 2023
Logs	May 04 2023; May 17 2023

Vulnerability Summary

Total High-Severity issues	0
Total Medium-Severity issues	4
Total Low-Severity issues	2
Total informational issues	5
Total	11

Contact

E-mail: support@salusec.io

Risk Level Description

High Risk	The issue puts a large number of users’ sensitive information at risk, or is reasonably likely to lead to catastrophic impact for clients’ reputations or serious financial implications for clients and users.
Medium Risk	The issue puts a subset of users’ sensitive information at risk, would be detrimental to the client’s reputation if exploited, or is reasonably likely to lead to a moderate financial impact.
Low Risk	The risk is relatively small and could not be exploited on a recurring basis, or is a risk that the client has indicated is low impact in view of the client’s business circumstances.
Informational	The issue does not pose an immediate risk, but is relevant to security best practices or defense in depth.

Content

Introduction

1.1 About SALUS

At Salus Security, we are in the business of trust.

We are dedicated to tackling the toughest security challenges facing the industry today. By building foundational trust in technology and infrastructure through security, we help clients to lead their respective industries and unlock their full Web3 potential.

Our team of security experts employ industry-leading proof-of-concept (PoC) methodology for demonstrating smart contract vulnerabilities, coupled with advanced red teaming capabilities and a stereoscopic vulnerability detection service, to deliver comprehensive security assessments that allow clients to stay ahead of the curve.

In addition to smart contract audits and red teaming, our Rapid Detection Service for smart contracts aims to make security accessible to all. This high calibre, yet cost-efficient, security tool has been designed to support a wide range of business needs including investment due diligence, security and code quality assessments, and code optimisation.

We are reachable on Telegram (https://t.me/salusec), Twitter (https://twitter.com/salus_sec), or Email (support@salusec.io).

1.2 Audit Breakdown

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

- Risky external calls
- Integer overflow/underflow
- Transaction-ordering dependence
- Timestamp dependence
- Access control
- Call stack limits and mishandled exceptions
- Number rounding errors
- Centralization of power
- Logical oversights and denial of service
- Business logic specification
- Code clones, functionality duplication

1.3 Disclaimer

Note that this security audit is not designed to replace functional tests required before any software release and does not give any warranties on finding all possible security issues with the given smart contract(s) or blockchain software, i.e., the evaluation result does not guarantee the nonexistence of any further findings of security issues.

Findings

2.1 Summary of Findings

ID	Title	Severity	Category	Status
1	User able to free mint NFT	Medium	Business Logic	Resolved
2	Wrong token received in makeOffer() function	Medium	Business Logic	Resolved
3	The NFT will be locked if the auction ends without a bidder	Medium	Business Logic	Resolved
4	Missing checks for user-provided address	Medium	Business Logic	Partially Resolved
5	Auction.winner should be auction.lastBidder	Low	Business Logic	Resolved
6	Mismatch between error message and implementation	Low	Business Logic	Resolved
7	Spelling mistakes	Informational	Code Quality	Resolved
8	Missing zero address checks	Informational	Data Validation	Acknowledged
9	Redundant code	Informational	Redundancy	Resolved
10	Gas optimization suggestions	Informational	Gas optimization	Acknowledged
11	Mismatch between error message and check	Informational	Business Logic	Resolved

2.2 Notable Findings

Significant flaws that impact system confidentiality, integrity, or availability are listed below.

1. User able to free mint NFT	
Severity: Medium	Category: Business Logic
Target: - Origindev-marketplace\contracts\contracts\OriginCreate.sol	

Description

Origindev-marketplace\contracts\contracts\OriginCreate.sol:L20-L37

```
function createNFT(string memory tokenURI) public returns (uint) {
    _tokenIds.increment();
    uint256 newItemId = _tokenIds.current();
    _mint(msg.sender, newItemId);
    _setTokenURI(newItemId, tokenURI);
    setApprovalForAll(contractAddress, true);
    return newItemId;
}

function mintNFT(string memory tokenURI) public payable returns (uint) {
    require(msg.value == cost, "Need to send 0.075 ether!");
    _tokenIds.increment();
    uint256 newItemId = _tokenIds.current();
    _mint(msg.sender, newItemId);
    _setTokenURI(newItemId, tokenURI);
    setApprovalForAll(contractAddress, true);
    return newItemId;
}
```

In the OriginCreate contract, there are two NFT minting functions, createNFT and mintNFT, where the mintNFT function requires the user to pay a cost amount of ether to mint NFT, while createNFT allows for free minting.

Users can use the createNFT function to mint NFT for themselves instead of the mintNFT function, bypassing the need to pay ether for the contract.

Recommendation

Consider adding access control to the createNFT function.

Status

This issue has been resolved by the team. The team has removed the createNFT function.

2. Wrong token received in makeOffer() function	
Severity: Medium	Category: Business Logic
Target: - Origindev-marketplace\contracts\contracts\OriginMarketplace.sol - Origindev-marketplace\contracts\contracts\Unik.sol	

Description

Origindev-marketplace\contracts\contracts\OriginMarketplace.sol:L268-L294

```
function makeOffer(address _nft, uint256 _tokenId, address _payToken, uint256
_offerPrice) external isListedNFT(_nft, _tokenId) {
       require(_offerPrice > 0, "price can not 0");

       ListNFT memory nft = listNfts[_nft][_tokenId];
       IERC20(nft.payToken).transferFrom(
           msg.sender,
           address(this),
           _offerPrice
       );

       offerNfts[_nft][_tokenId][msg.sender] = OfferNFT({
           nft: nft.nft,
           tokenId: nft.tokenId,
           offerer: msg.sender,
           payToken: _payToken,
           offerPrice: _offerPrice,
           accepted: false
       });

       ...
}
```

Origindev-marketplace\contracts\contracts\Unik.sol:L1733-L1764

```
function makeOffer(
       address _nft,
       uint256 _tokenId,
       address _payToken,
       uint256 _offerPrice
) external isListedNFT(_nft, _tokenId) {
       require(_offerPrice > 0, "price can not 0");

       ListNFT memory nft = listNfts[_nft][_tokenId];
       IERC20(nft.payToken).transferFrom(
           msg.sender,
           address(this),
           _offerPrice
       );

       offerNfts[_nft][_tokenId][msg.sender] = OfferNFT({
           nft: nft.nft,
            tokenId: nft.tokenId,
            offerer: msg.sender,
            payToken: _payToken,
            offerPrice: _offerPrice,
            accepted: false
        });

        ...
}
```
Users can make an offer on an NFT with the specified _payToken and _offerPrice. If an offer is accepted, the amount of payToken provided by the offeror, with fees deducted, is transferred to the seller. However, in the makeOffer function, the contract receives the token as nft.paytoken instead of _paytoken.

Recommendation

Consider changing `IERC20(nft.payToken).transferFrom` to `IERC20(_payToken).transferFrom`.

Status

This issue has been resolved by the team. The payToken has been fixed by the team. All Bids (previously Offer) can only use the contract-specified payToken.

3. The NFT will be locked if the auction ends without a bidder	
Severity: Medium	Category: Business Logic
Target: - Origindev-marketplace\contracts\contracts\OriginMarketplace.sol - Origindev-marketplace\contracts\contracts\Unik.sol	

Description

The NFT will be transferred from the caller to the contract when a user creates an auction. The completeBid function and cancelAuction function can be called to send NFT.

Origindev-marketplace\contracts\contracts\OriginMarketplace.sol:L504
Origindev-marketplace\contracts\contracts\Unik.sol:L2021

```
nft.transferFrom(address(this), auction.lastBidder, auction.tokenId);
```

If an auction ends without a bidder, the value of auction.lastBidder will be zero and the NFT transfer will be reverted. Thus, this auction can not be completed.

Origindev-marketplace\contracts\contracts\OriginMarketplace.sol:L425
Origindev-marketplace\contracts\contracts\Unik.sol:L1920

```
require(block.timestamp < auction.startTime, "auction already started");
```

Meanwhile, an auction can not be cancelled since it has already started.

Therefore, the NFT for auction is locked.

Recommendation

Consider transferring the NFT to the creator if there is no bidder when the auction ends.

Status

This issue has been resolved by the team. The team has updated the business logic. A seller does not have to transfer the NFT to the contract when listing an NFT. When a user purchases or the bid is accepted by the seller, the NFT is transferred from seller to buyer.

4. Missing checks for user-provided address	
Severity: Medium	Category: Business Logic
Target: - Origindev-marketplace\contracts\contracts\OriginMarketplace.sol - Origindev-marketplace\contracts\contracts\Unik.sol	

Description

Origindev-marketplace\contracts\contracts\OriginMarketplace.sol:L268-L294

```
function makeOffer(address _nft, uint256 _tokenId, address _payToken, uint256
_offerPrice) external isListedNFT(_nft, _tokenId) {
    ...
}
```

Origindev-marketplace\contracts\contracts\Unik.sol:L1733-L1764

```
function makeOffer(
    address _nft,
    uint256 _tokenId,
    address _payToken,
    uint256 _offerPrice
) external isListedNFT(_nft, _tokenId) {
    ...
}
```

Since the _payToken parameter is not checked, the user can provide any address that is not actually allowed.

Origindev-marketplace\contracts\contracts\OriginMarketplace.sol:L383-L415

```
function createAuction(address _nft, uint256 _tokenId, address _payToken, uint256
_price, uint256 _minBid, uint256 _startTime, uint256 _endTime) external
isPayableToken(_payToken) isNotAuction(_nft, _tokenId) {
    ...
}
```

Origindev-marketplace\contracts\contracts\Unik.sol:L1867-L1907

```
function createAuction(
    address _nft,
    uint256 _tokenId,
    address _payToken,
    uint256 _price,
    uint256 _minBid,
    uint256 _startTime,
    uint256 _endTime
) external isPayableToken(_payToken) isNotAuction(_nft, _tokenId) {
    ...
}
```

The isNotAuction modifier can only guarantee that the auction has not been created or the auction has succeeded. Since the _nft parameter is not checked, the user can provide any address that is not actually allowed.

Recommendation

Consider adding the isPayableToken modifier to the makeOffer function.

Consider adding the isUnikNFT modifier to the createAuction function of the UnikNFTMarketplace contract and adding checks for the _nft parameter to the createAuction function of the OriginMarketplace contract.

Status

This issue has been partially resolved by the team. The _payToken parameter has been removed but the erc721Address parameter (previously the _nft parameter) is not checked.

5. Auction.winner should be auction.lastBidder	
Severity: Low	Category: Business Logic
Target: - Origindev-marketplace\contracts\contracts\Unik.sol - Origindev-marketplace\contracts\contracts\OriginMarketplace.sol	

Description

Origindev-marketplace\contracts\contracts\OriginMarketplace.sol:L479
Origindev-marketplace\contracts\contracts\Unik.sol:L1996

```
auction.winner = auction.creator;
```

auction.winner records the final winner of the auction process, it should be recorded as lastBidder and not as the creator of the auction.

Recommendation

Consider replacing auction.creator with auction.lastBidder.

Status

This issue has been resolved by the team. The team has updated the business logic.

6. Mismatch between error message and implementation	
Severity: Low	Category: Business Logic
Target: - Origindev-marketplace\contracts\contracts\Unik.sol - Origindev-marketplace\contracts\contracts\OriginMarketplace.sol	

Description

Origindev-marketplace\contracts\contracts\OriginMarketplace.sol:L520-522
Origindev-marketplace\contracts\contracts\Unik.sol:L2041-L2047

```
function calculateRoyalty(uint256 _royalty, uint256 _price)
    public
    pure
    returns (uint256)
{
    return (_price * _royalty) / 10000;
}
```

Origindev-marketplace\contracts\contracts\OriginMarketplace.sol:L126,L544
Origindev-marketplace\contracts\contracts\Unik.sol:L1320,L1361,L1548,L2077

```
require(_royaltyFee <= 10000, "can't more than 10 percent");
```

The error message shows the _royaltyFee can't be more than 10 percent, but the check for _royaltyFee is less than 10000, which actually cannot be more than 100%.

Recommendation

Consider replacing <= 10000 with <= 1000 to fix the mismatch between error message and implementation.

Status

This issue has been resolved by the team.

2.3 Informational Findings

7. Spelling mistakes	
Severity: Informational	Category: Code Quality
Target: - Origindev-marketplace\contracts\contracts\Unik.sol	

Description

Origindev-marketplace\contracts\contracts\Unik.sol:L1471

```
// nft => tokenId => acuton struct
```

Acuton should be auction.

Origindev-marketplace\contracts\contracts\Unik.sol:L1474

```
// auciton index => bidding counts => bidder address => bid price
```

Auciton should be auction.

Origindev-marketplace\contracts\contracts\Unik.sol:L1603

```
"not offerred nft"
```

Offerred should be offered.

Recommendation

It is recommended to change acuton to auction and auciton to auction, offerred to offered.

Status

This issue has been resolved by the team.

8. Missing zero address checks	
Severity: Informational	Category: Data Validation
Target: - Origindev-marketplace\contracts\contracts\Unik.sol	

Description

Origindev-marketplace\contracts\contracts\Unik.sol:L1543-L1552

```
constructor(
    uint256 _platformFee,
    address _feeRecipient,
    IUnikNFTFactory _unikNFTFactory
) {
    require(_platformFee <= 10000, "can't more than 10 percent");
    platformFee = _platformFee;
    feeRecipient = _feeRecipient;
    unikNFTFactory = _unikNFTFactory;
}
```

It is considered a security best practice to verify addresses against the zero-address during constructor or setting. However, this precautionary step is absent for address variables in the constructors.

Recommendation

Consider adding zero-address checks for address variables in the constructors.

Status

This issue has been acknowledged by the team.

9. Redundant code	
Severity: Informational	Category: Redundancy
Target: - Origindev-marketplace\contracts\contracts\Unik.sol	

Description

Origindev-marketplace\contracts\contracts\Unik.sol:L1422

```
contract UnikNFTMarketplace is Ownable, ReentrancyGuard {
```

The inheritance of ReentrancyGuard is redundant, it is not used in the UnikNFTMarketplace contract.

Origindev-marketplace\contracts\contracts\Unik.sol:L1609-L1612

```
require(
     _payToken != address(0) && payableToken[_payToken],
     "invalid pay token"
);
```

Checking if _payToken is not 0 is redundant, since `payableToken[0]` is always false.

Origindev-marketplace\contracts\contracts\Unik.sol:L1674-L1677

```
require(
    _payToken != address(0) && _payToken == listedNft.payToken,
    "invalid pay token"
);
```

The paytoken check can be removed because the user provide _payToken is not used.

Origindev-marketplace\contracts\contracts\Unik.sol:L1308-L1310

```
struct Artwork{
    address payable creator;
    uint8 royalty;
}
```

The struct Artwork is not used in the UnikNFT contract.

Recommendation

Consider removing the redundant code.

Status

This issue has been resolved by the team.

10. Gas optimization suggestions	
Severity: Informational	Category: Gas Optimization
Target: - Origindev-marketplace\contracts\contracts\OriginCreate.sol	

Description

Origindev-marketplace\contracts\contracts\OriginCreate.sol:L13

```
address contractAddress;
```

To reduce gas costs, the state variable contractAddress could be declared as immutable since its value is fixed after the contract has been deployed.

Origindev-marketplace\contracts\contracts\OriginCreate.sol:L14

```
uint256 public cost = 0.0075 ether;
```

The state variable cost could be declared as constant since its value is fixed before the contract compilation.

Recommendation

Consider adding the immutable modifier to the contractAddress state variable and adding the constant modifier to the cost state variable.

Status

This issue has been acknowledged by the team.

11. Mismatch between error message and check	
Severity: Informational	Category: Business Logic
Target: - Origindev-marketplace\contracts\contracts\Unik.sol	

Description

Origindev-marketplace\contracts\contracts\Unik.sol:L1577-L1584

```
modifier isAuction(address _nft, uint256 _tokenId) {
    AuctionNFT memory auction = auctionNfts[_nft][_tokenId];
    require(
        auction.nft != address(0) && !auction.success,
        "auction already created"
    );
    _;
}
```

isAuction checks if the auction has been started, so the error message should be “auctions have not been created”.

Recommendation

Consider fixing the mismatch between error message and check.

Status

This issue has been resolved by the team.

Appendix

Appendix 1 - Files in Scope

This audit covered the following file:

File	SHA-1 hash
Origindev-marketplace\contracts\contracts\lib\IOriginNFT.sol	e1235256bdadfedb8bb8f66c4c6ef1789eeee3d7
Origindev-marketplace\contracts\contracts\OriginCreate.sol	438d0bc083904255a6b3aff4aa2535b61b3618ec
Origindev-marketplace\contracts\contracts\OriginMarketplace.sol	5c6c5461e70fe47c4db75f357c77c03b07322ad3
Origindev-marketplace\contracts\contracts\PayToken.sol	671d12d49a2da6d74fcb8bd4cb73ae688faa9633
Origindev-marketplace\contracts\contracts\Unik.sol	66274528c7b44080eb089da0e2b9c44f8ddf20bd