Fireverse
Audit Report

Thu Jun 26 2025

contact@bitslab.xyz https://twitter.com/scalebit_

ScaleBit

Fireverse Audit Report

1 Executive Summary

1.1 Project Information

Description	Fireverse AI enables the transformation of AI-generated music into on-chain music NFTs. Users can freely trade their music NFT creations within the platform
Type	NFT Marketplace
Auditors	ScaleBit
Timeline	Thu May 22 2025 - Thu Jun 26 2025
Languages	Solidity
Platform	EVM Chains
Methods	Architecture Review, Unit Testing, Manual Review
Source Code	https://github.com/FireverseAI/contract
Commits	2639fbdb1a6e3017072db7437c71bd2d8a389732 61f6f9a6db6801330b87f528bccde21b53189cdf

1.2 Files in Scope

The following are the SHA1 hashes of the original reviewed files.

ID	File	SHA-1 Hash
FVNFTM1	contracts/FireVerseNFTMarketplace.sol	27de9140c7592f029ba116ac67fcbe04cd2ba527
FVNFT1	contracts/FireVerseNFT.sol	0ceff9311e3fc94ab34116bdf2358d2f0c13f503

1.3 Issue Statistic

Item	Count	Fixed	Acknowledged
Total	4	2	2
Informational	0	0	0
Minor	2	1	1
Medium	2	1	1
Major	0	0	0
Critical	0	0	0

1.4 ScaleBit Audit Breakdown

ScaleBit aims to assess repositories for security-related issues, code quality, and compliance with specifications and best practices. Possible issues our team looked for included (but are not limited to):

- Transaction-ordering dependence
- Timestamp dependence
- Integer overflow/underflow
- Number of rounding errors
- Unchecked External Call
- Unchecked CALL Return Values
- Functionality Checks
- Reentrancy
- Denial of service / logical oversights
- Access control
- Centralization of power
- Business logic issues
- Gas usage
- Fallback function usage
- tx.origin authentication
- Replay attacks
- Coding style issues

1.5 Methodology

The security team adopted the **"Testing and Automated Analysis"**, **"Code Review"** and **"Formal Verification"** strategy to perform a complete security test on the code in a way that is closest to the real attack. The main entrance and scope of security testing are stated in the conventions in the "Audit Objective", which can expand to contexts beyond the scope according to the actual testing needs. The main types of this security audit include:

(1) Testing and Automated Analysis

Items to check: state consistency / failure rollback / unit testing / value overflows / parameter verification / unhandled errors / boundary checking / coding specifications.

(2) Code Review

The code scope is illustrated in section 1.2.

(3) Audit Process

- Carry out relevant security tests on the testnet or the mainnet;
- If there are any questions during the audit process, communicate with the code owner in time. The code owners should actively cooperate (this might include providing the latest stable source code, relevant deployment scripts or methods, transaction signature scripts, exchange docking schemes, etc.);
- The necessary information during the audit process will be well documented for both the audit team and the code owner in a timely manner.

2 Summary

This report has been commissioned by Fireverse to identify any potential issues and vulnerabilities in the source code of the Fireverse smart contract, as well as any contract dependencies that were not part of an officially recognized library. In this audit, we have utilized various techniques, including manual code review and static analysis, to identify potential vulnerabilities and security issues.

During the audit, we identified 4 issues of varying severity, listed below.

ID	Title	Severity	Status
FVN-1	Single-step Ownership Transfer Can be Dangerous	Medium	Fixed
FVN-2	`call()` Should be Used Instead of `transfer()` on An Address Payable	Medium	Acknowledged
FVN-3	The Signature does not Include the Chain ID	Minor	Acknowledged
FVN-4	`setPlatformFee` Parameter Range Of FeeBps Is Incorrect	Minor	Fixed

3 Participant Process

Here are the relevant actors with their respective abilities within the Fireverse Smart Contract :

Admin

- `setTokenRoyalty` : Specified NFT sets the royalty recipient and royalty ratio
- `setDefaultFeeNumerator` Set the default royalty ratio of the contract
- `allowPaymentToken` Add or remove supported payment tokens
- `allowNFT` Manage the NFT contract addresses that are allowed to be traded in the market
- `setPlatformFee` : Set the platform's handling fee receiving address and handling fee ratio

User

- `mint/batchMint` : Mint individual or batch NFTS
- `burn` Owner destroys the NFT
- `buy` Users purchase NFTS based on orders
- `cancelOrder/batchCancelOrder` : Users cancel individual orders or batch orders

4 Findings

FVN-1 Single-step Ownership Transfer Can be Dangerous

Severity: Medium

Status: Fixed

Code Location:

contracts/FireVerseNFTMarketplace.sol#12

Descriptions:

Single-step ownership transfer means that if a wrong address was passed when transferring ownership or admin rights it can mean that role is lost forever. If the admin permissions are given to the wrong address within this function, it will cause irreparable damage to the contract. Below is the official documentation explanation from OpenZeppelin https://docs.openzeppelin.com/contracts/4.x/api/access

`Ownable is a simpler mechanism with a single owner "role" that can be assigned to a single account. This simpler mechanism can be useful for quick tests but projects with production concerns are likely to outgrow it.`

The `FireVerseNFTMarketplace` contract inherits from the `Ownable` contract.

```
contract FireVerseNFTMarketplace is EIP712, Ownable, ReentrancyGuard {
    using ECDSA for bytes32;
    using SafeERC20 for IERC20;
    using Address for address payable;
```

In these contracts, transferring ownership is a single-step process, which poses the aforementioned risk. https://github.com/OpenZeppelin/openzeppelin-contracts-upgradeable/blob/master/contracts/access/OwnableUpgradeable.sol#L102-L118

Suggestion:

It is recommended to use the `Ownable2StepUpgradeable` contract.

Resolution:

This issue has been fixed. The client has adopted our suggestions.

FVN-2 `call()` Should be Used Instead of `transfer()` on An Address Payable

Severity: Medium

Status: Acknowledged

Code Location:

contracts/FireVerseNFTMarketplace.sol#119-121

Descriptions:

The `transfer()` and `send()` functions forward a fixed amount of 2300 gas. Historically, it has often been recommended to use these functions for value transfers to guard against reentrancy attacks. However, the gas cost of EVM instructions may change significantly during hard forks which may break already deployed contract systems that make fixed assumptions about gas costs. For example. EIP 1884 broke several existing smart contracts due to a cost increase of the SLOAD instruction. https://swcregistry.io/docs/SWC-134

In the `buy()` function, the protocol utilizes the `sendValue()` method to transfer the ETH to the target address.

```
    if (order.paymentToken == address(0)) {
        // Native Token
        require(msg.value == order.price, "Incorrect Native Token amount");
        if (royaltyAmount > 0) payable(royaltyRecipient).sendValue(royaltyAmount);
        if (platformFee > 0) payable(platformFeeRecipient).sendValue(platformFee);
        payable(order.seller).sendValue(sellerAmount);
      } else {
        // ERC20
        IERC20 token = IERC20(order.paymentToken);
        if (royaltyAmount > 0) token.safeTransferFrom(msg.sender, royaltyRecipient,
royaltyAmount);
        if (platformFee > 0) token.safeTransferFrom(msg.sender, platformFeeRecipient,
platformFee);
```

```
        token.safeTransferFrom(msg.sender, order.seller, sellerAmount);
    }
```

The use of the deprecated `sendValue()` function for an address will inevitably make the transaction fail when:

The claimer smart contract does not implement a payable function.

The claimer smart contract does implement a payable fallback which uses more than 2300 gas unit.

The claimer smart contract implements a payable fallback function that needs less than 2300 gas units but is called through proxy, raising the call's gas usage above 2300.

Additionally, using higher than 2300 gas might be mandatory for some multisig wallets.

Suggestion:

It is recommended to use `call()` instead of `sendValue()`, but be sure to respect the CEI pattern and/or add re-entrancy guards, as several hacks already happened in the past due to this recommendation not being fully understood.

FVN-3 The Signature does not Include the Chain ID

Severity: Minor

Status: Acknowledged

Code Location:

contracts/FireVerseNFTMarketplace.sol#79-94

Descriptions:

The `verify()` function is used to verify signatures. However, the signature does not include the chain ID, which means the signed message could potentially be replayed on a different blockchain

```
function verify(Order calldata order, bytes calldata signature) public view returns (bool) {
    bytes32 structHash = keccak256(
      abi.encode(
        ORDER_TYPEHASH,
        order.seller,
        order.nft,
        order.tokenId,
        order.price,
        order.paymentToken,
        order.nonce,
        order.expiry
      )
    );
    bytes32 digest = _hashTypedDataV4(structHash);
    return digest.recover(signature) == order.seller;
  }
```

Suggestion:

It is recommended to incorporate the chain ID into the `ORDER_TYPEHASH`

FVN-4 `setPlatformFee` Parameter Range Of FeeBps Is Incorrect

Severity: Minor

Status: Fixed

Code Location:

contracts/FireVerseNFTMarketplace.sol#72-77

Descriptions:

In the function setPlatformFee, the maximum value of the feeBps parameter is 10,000

```
function setPlatformFee(address recipient, uint96 feeBps) external onlyOwner {
    require(feeBps <= 10000, "Over 100%");
    platformFeeRecipient = recipient;
    platformFeeBps = feeBps;
    emit PlatformFeeUpdated(recipient, feeBps);
}
```

When calculating the transaction fee and other charges of the function buy, if platformFee is set to 10,000, sellerAmount parameter will be less than 0, it will cause the function to fail to run

```
(address royaltyRecipient, uint256 royaltyAmount) = _getRoyalty(order.nft,
order.tokenId, order.price);
uint256 platformFee = (order.price * platformFeeBps) / 10_000;
uint256 sellerAmount = order.price - royaltyAmount - platformFee;
```

Suggestion:

Modify the parameter range and detect the sellerAmount value

```
require(feeBps < 10000, "Over 100%");
```

Resolution:

This issue has been fixed. The client has adopted our suggestions.

Appendix 1

Issue Level

- **Informational** issues are often recommendations to improve the style of the code or to optimize code that does not affect the overall functionality.
- **Minor** issues are general suggestions relevant to best practices and readability. They don't post any direct risk. Developers are encouraged to fix them.
- **Medium** issues are non-exploitable problems and not security vulnerabilities. They should be fixed unless there is a specific reason not to.
- **Major** issues are security vulnerabilities. They put a portion of users' sensitive information at risk, and often are not directly exploitable. All major issues should be fixed.
- **Critical** issues are directly exploitable security vulnerabilities. They put users' sensitive information at risk. All critical issues should be fixed.

Issue Status

- **Fixed:** The issue has been resolved.
- **Partially Fixed:** The issue has been partially resolved.
- **Acknowledged:** The issue has been acknowledged by the code owner, and the code owner confirms it's as designed, and decides to keep it.

Appendix 2

Disclaimer

This report is based on the scope of materials and documents provided, with a limited review at the time provided. Results may not be complete and do not include 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 own risk. A report does not imply an endorsement of any particular project or team, nor does it guarantee its security. These reports should not be relied upon in any way by any third party, including for the purpose of making any decision to buy or sell products, services, or any other assets. TO THE FULLEST EXTENT PERMITTED BY LAW, WE DISCLAIM ALL WARRANTIES, EXPRESS OR IMPLIED, IN CONNECTION WITH THIS REPORT, ITS CONTENT, RELATED SERVICES AND PRODUCTS, AND YOUR USE, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, NOT INFRINGEMENT.