



Smart contracts security assessment

Final report

[Tariff: Standard](#)

Anyrand

October 2024



0xguard.com



hello@0xguard.com

Contents

1. Introduction	3
2. Contracts checked	3
3. Procedure	4
4. Known vulnerabilities checked	4
5. Classification of issue severity	6
6. Issues	6
7. Conclusion	9
8. Disclaimer	10

Introduction

Anyrand is a random number coordinator contract similar to Chainlink VRF (Verifiable Random Function). It allows to request and fulfill requests of verifiable random numbers derived from off-chain entropy provided by [drand](#). Execution of request fulfillment is sponsored by the request creator, execution is also permissionless, but the sponsored amount remains for the contract owner. Sponsoring amount is calculated by using a gas station contract, that is individual and specific for each chain.

Anyrand is an upgradable contract with a single owner. We recommend the owner to secure his account, and users should check the current implementation before interacting with the contract.

The code is available at the GitHub repository [frogworksio/anyrand](#) and was audited after the commit [a48f47c84c945d0729515324d51780ab3a9c6c56](#).

The audit scope excludes external libraries, including the [@kevincharm/bls-bn254](#) cryptographic library.

Report update. The contract's code was updated according to this report and rechecked after the commit [60b8143b568e16d11417c3b47137d4dee30d8550](#). The Anyrand contract was deployed to the Scroll network at address [0x7ED45287f817842d72753FE02617629c4c7c2FBE](#).

Name	Anyrand
Audit date	2024-10-10 - 2024-10-12
Language	Solidity
Platform	Scroll zkEVM

Contracts checked

Name	Address
Anyrand.sol	0x7ED45287f817842d72753FE02617629c4c7c2FBE

AnyrandStorage.sol
Gas.sol
GasStationEthereum.sol
GasStationOptimism.sol
GasStationScroll.sol
DrandBeacon.sol

Procedure

We perform our audit according to the following procedure:

Automated analysis

- Scanning the project's smart contracts with several publicly available automated Solidity analysis tools
- Manual verification (reject or confirm) all the issues found by the tools

Manual audit

- Manually analyze smart contracts for security vulnerabilities
- Smart contracts' logic check

Known vulnerabilities checked

Title	Check result
<u>Unencrypted Private Data On-Chain</u>	passed
<u>Code With No Effects</u>	passed
<u>Message call with hardcoded gas amount</u>	passed
<u>Typographical Error</u>	passed
<u>DoS With Block Gas Limit</u>	passed

<u>Presence of unused variables</u>	passed
<u>Incorrect Inheritance Order</u>	passed
<u>Requirement Violation</u>	passed
<u>Weak Sources of Randomness from Chain Attributes</u>	passed
<u>Shadowing State Variables</u>	passed
<u>Incorrect Constructor Name</u>	passed
<u>Block values as a proxy for time</u>	passed
<u>Authorization through tx.origin</u>	passed
<u>DoS with Failed Call</u>	passed
<u>Delegatecall to Untrusted Callee</u>	passed
<u>Use of Deprecated Solidity Functions</u>	passed
<u>Assert Violation</u>	passed
<u>State Variable Default Visibility</u>	passed
<u>Reentrancy</u>	passed
<u>Unprotected SELFDESTRUCT Instruction</u>	passed
<u>Unprotected Ether Withdrawal</u>	passed
<u>Unchecked Call Return Value</u>	passed
<u>Floating Pragma</u>	passed
<u>Outdated Compiler Version</u>	passed
<u>Integer Overflow and Underflow</u>	passed
<u>Function Default Visibility</u>	passed

🛡️ Classification of issue severity

High severity	High severity issues can cause a significant or full loss of funds, change of contract ownership, major interference with contract logic. Such issues require immediate attention.
Medium severity	Medium severity issues do not pose an immediate risk, but can be detrimental to the client's reputation if exploited. Medium severity issues may lead to a contract failure and can be fixed by modifying the contract state or redeployment. Such issues require attention.
Low severity	Low severity issues do not cause significant destruction to the contract's functionality. Such issues are recommended to be taken into consideration.

🛡️ Issues

High severity issues

1. Incorrect deadline to round calculations (Anyrand.sol)

Status: Fixed

The function `requestRandomness` transforms the deadline input parameter into the beacon's round parameter. The rounding up uses `delta % period`, that leads to non-continuous result and undocumented features, as `round(deadline1) = round(deadline1 + period - 1)`.

```
function requestRandomness(
    uint256 deadline,
    uint256 callbackGasLimit
) external payable override nonReentrant returns (uint256) {
    . . .
    uint256 genesis = drandBeacon.genesisTimestamp();
    uint256 period = drandBeacon.period();
    uint256 delta = deadline - genesis;
    round = uint64((delta / period) + (delta % period));
    . . .
}
```

Recommendation: Add `round = ((delta % period) > 0) ? (round + 1) : (round);` line or document the feature.

2. Owner privileges (Anyrand.sol)

Status: Partially fixed

The contract owner can fulfill any request by changing request's beacon address. He can also set a malicious beacon address to prevent requests with that beacon public key from fulfillment (including permissionless fulfillment).

```
/// @notice Add a new beacon and set the current beacon to it (privileged)
/// @notice This is intended to be used only in the case that the evmnet
///         beacon is deprecated in favour of the BLS12-381 beacon.
/// @notice NB: This can replace/fix a beacon that is known to this
///         contract by its public key hash.
/// @param newBeacon The new beacon
function setBeacon(address newBeacon) external onlyOwner {
    _setBeacon(newBeacon);
}

/// @notice Add a new beacon and set the current beacon to it
/// @param newBeacon The new beacon
function _setBeacon(address newBeacon) internal {
    // Sanity check
    try IDrandBeacon(newBeacon).publicKeyHash() returns (
        bytes32 pubKeyHash
    ) {
        if (pubKeyHash == bytes32(0) || pubKeyHash == keccak256(hex"")) {
            revert InvalidBeacon(newBeacon);
        }

        // Looks good - add the beacon and update it
        MainStorage storage $ = _getMainStorage();
        $.beacons[pubKeyHash] = newBeacon;
        $.currentBeaconPubKeyHash = pubKeyHash;
        emit BeaconUpdated(newBeacon);
    } catch {
        revert InvalidBeacon(newBeacon);
    }
}
```

```
}
```

Recommendation: Secure the owner's account.

Anyrand comment: Following production deployment, the owner will immediately be transferred to a 3/4 multisig, initially only the Anyrand team, with the aim of increasing the threshold and finding more reputable signers from the community.

Medium severity issues

No issues were found

Low severity issues

1. `getRequestPrice` function may return incorrect data in explorers (Anyrand.sol)

Status: Fixed

Return values of the function `getRequestPrice` are calculated via external call to an `IGasStation` implementation, but all existed implementations (`GasStationEthereum`, `GasStationOptimism`, `GasStationScroll`) rely on `tx.gasprice` value, which can equal to zero when viewed via the network explorer.

Recommendation: Cover this case in the documentation and in the NatSpec descriptions.

Conclusion

Anyrand Anyrand.sol, AnyrandStorage.sol, Gas.sol, GasStationEthereum.sol, GasStationOptimism.sol, GasStationScroll.sol, DrandBeacon.sol contracts were audited. 2 high, 1 low severity issues were found.

1 high, 1 low severity issues have been fixed in the update.

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 Services Agreement, or the scope of services, and terms and conditions provided to the Company in connection with the Agreement. This report provided in connection with the Services set forth in the Agreement shall be used by the Company only to the extent permitted under the terms and conditions set forth in the Agreement. This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes without 0xGuard 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 considered, an indication of the economics or value of any “product” or “asset” created by any team or project that contracts 0xGuard to perform a security assessment. This report 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 in no way provides investment advice, nor should be leveraged as investment advice of any sort. This report represents an extensive assessing 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.



 Guard