Vulnerability Disclosure: Shadowcorn contract randomness exploit

yhtyyar, Moonstream DAO

Date of disclosure: July 10, 2022
Status: Vulnerability was fixed by Laguna Games in transaction
0xbcb3395f55b74428694d2caca894162b9ba1faed5091a5e8116db35f75a737bd

Vulnerability description:

This report discloses a vulnerability in the Crypto Unicorns Shadowcorns contract related to how the contract uses randomness to determine the statistics of minted Shadowcorns.

Even though the contract is using Chainlink VRF to get random numbers, the method it uses is exploitable because of how `fulfillRandomness`, `retryHatching` , and ERC721 `onERC721Recieved` work. An attacker would be able to deploy a smart contract that could revert the transaction in the `onERC721Recieved` callback in case they received statistics that they were not pleased with.

The Crypto Unicorns Shadowcorns contract is an ERC721 token contract that uses EIP-2535 (multi-faceted Diamond proxy) to make the contract upgradable.
The shadowcorn diamond proxy address: 0xa7D50EE3D7485288107664cf758E877a0D351725
The Hatching facet address: 0xA26d862130599300A7086056CbCe5653F01Abc37

How the hatching works:

1) You call `beginHatching`, the contract takes the minting fees.

```
function beginHatching(uint256 terminusPoolId) external {
        //takes minting fees
        //mints new token
       //calls requestRandomness to chainlink VRF contract
 }
```

2) When Chainlink VRF fulfills the request for randomness, inside `rawFulfillRandomness`, the contract sets the DNA of the Shadowcorn and transfers the token to user via `safeTransferFrom`
3) If Chainlink VRF failed to deliver the random number or there was problem with making random DNA for the shadowcorn, the caller may retry the hatching by calling retryHatching. You will need to wait for the `retryBlockDeadline` number of blocks between retries.

The exploit:
When `safeTransferFrom` is called, according to the ERC721 standard, if the receiver of the token is a contract, the callback `onERC721Recieved` from the receiver contract is called.
By deploying an exploit contract, it is possible to get shadowcorn's attributes and revert the transaction inside this callback.
The fact that there are no public endpoints to get dna/attributes makes it very difficult to perform the exploit. However there are ways around this:

1) The `tokenUri` endpoint directly returns JSON metadata (see contract level metadata). By parsing the JSON inside solidity (there are some libraries that could be used)
2) Wait until the public dna endpoint is added. The shadowcorn tokens are intended to be used in on-chain minigames. This makes it likely that endpoints which make DNA viewable will be added in the near future.

Testing the exploit on mainnet.

After understanding the potential for this exploit, I wrote a Proof of Concept to test it. The POC exploit contract doesn't perform the full exploit with attributes/stats sniping; it has a flag which will turn on/off reverting of the transaction.

```
function onERC721Received(
        address operator,
        address from,
        uint256 tokenId,
        bytes calldata data
    ) external returns (bytes4) {
        if (needToRevert) {
            // Here should be checked the stats of shadowcorn
            revert("");
        }
        return this.onERC721Received.selector;
    }
```

The contract address and the transactions proving the exploit:
https://polygonscan.com/address/0xfa88d40bfa0a0ad72dcde0e5fb07b76b9c7052e0
I have retried hatching 4 times.

Full source code is available here:
https://github.com/Yhtiyar/shadowcorns-exploit

Possible mitigations:

1. Follow the Chainlinks security advice, don't make it possible to revert the `fulfillRandomness` callback. In the case of Shadowcorns the Crypto Unicorns team could achieve this as follows:
    a. determine the dna
    b. don't transfer the token on the same transaction
    c. force the user user to submit a second transaction to get their token

Laguna Games team response:

The Laguna Games team were warned about this exploit after I tested it on Polygon mainnet.

Since the contract is an upgradable proxy, the Laguna Games team promptly fixed the issue by deploying a new version of the contract.

Response from Laguna Games team:
Exploit was patched in Shadowcorns contract v1.0.3

- The exploit was added to our automated test suite to avoid regression
- The hatching library was updated in three ways:
    - Disallow external calls to `beginHatching` by external contracts - `msg.sender` must be an EOA
    - Disallow external calls to `retryHatching` by external contracts - `msg.sender` must be an EOA
    - Transfer of the finalized NFT no longer depends on `onERC721Received`
- The hatching facet was replaced with a new version: `0xa6cdb2c7c0ce324b272AAD7D197e2435243A5Ca4`
- The RNG facet was replaced with a new version: `0x153D9656DE210D39F1352827D3bFA64131920642`

The issue was fully resolved in this transaction:
https://polygonscan.com/tx/0xbcb3395f55b74428694d2caca894162b9ba1faed5091a5e8116db35f75a737bd