NeoTokyo Land Deeds Rarity Exploit

spectre {Twitter: @spectre_111}

January 9, 2022

1 Introduction

This paper documents a NeoTokyo rarity exploit. NeoTokyo is an NFT collection created by Alex Becker, with a market value in the multi-digit millions. The exploit enables the prediction of rarities of unminted Land Deeds. If this exploit were to be known by the community, it would strongly affect the NeoTokyo ecosystem, since minting isn't random and no one will mint unless there is a rare Land Deed coming up. This would reduce the function of $BYTES greatly and increase its inflation rate as no more $BYTES are burned for minting.

2 Technical Analysis

Severity: Medium

Description:
The tokenID of the Land Deed to be minted next is always one larger than the tokenID of the most recently (already) minted Land Deed. Thus the tokenID of the next token to be minted is predictable. Further, the attributes and thus the rarity of each tokenID are determined by a function in a separate contract, no matter if minted yet or not. The solidity code of this second contract is not published, hence we can only analyse the bytecode which is stored on the blockchain. The normal $tokenURI()$ function implements a requirement, which states that the token with the specified tokenID has to be minted before calling the external contract to get the tokenURI or location attribute. The issue is that the second contract itself has no requirement for the token to be minted before accessing the state. This enables us to simulate the main contract by removing the $require(exists(tokenID), \dots)$ check before calling the external contract. Combining the above, we can figure out what the future tokenIDs will be and fetch their tokenURI. Hence, we can fully determine all the attributes of future mints. This can be done without minting a single token on mainnet.
To run a simulation in a local environment, we copied the Land Deed source code [1] and removed the $require(exists(tokenID), \dots)$ statement on line 1298 in the $getLocation()$ function. We can simply execute $getLocation(x)$ for an arbitrary tokenID x to determine its location. The same steps can be carried out for $tokenURI(x)$. This exploit allows the preparation of bots, which will instantaneously mint a rare Land Deed, as soon as its predecessor was minted.
People not aware of this technique will only ever get common Land Deeds, since bots will snipe the rare ones. Hence, creating a very unfair experience for many users. At the time of writing the cost of 500 $BYTES, which are needed to buy a Land Deed, is approximately 15 ETH. Since the value of $BYTES is somewhat tied to the value of what you can buy with it, this could also negatively impact the value of the currency.

3 Reproduction

We have created a git repository to easily verify our claims. The repository contains the *NTLandDeedDeploy* contract [1], where line 1301 is commented out to enable queries targeting unminted tokenIDs, the test script which will deploy the contract and call *getLocation*(), and the config file which takes care of forking mainnet using the hardhat development environment and the alchemy API. All our testing was done on Linux. Following steps need to be performed to run the code:

1. Install hardhat
2. Create a new hardhat project
3. Get the files from our github
4. Paste the edited contract into the *contracts/* and the test script into the *test/* folder
5. Get and add the *hardhat.config.js* file to the project folder
6. Navigate into the project folder with a terminal
7. Run "npx hardhat test test/getRarity.js" in console
8. The contracts will be simulated and the results will be printed to console

4 Solution

The most effective way to solve this issue is to implement Chainlink VRF for tokenID generation or tokenURI mapping. If the tokenID of the next mint is non-deterministic it is useless to figure out which tokenID has which rarity, and thus the whole exploit falls apart. Since secure randomness is difficult to code, especially when the source code is visible on-chain, implementing the popular Chainlink VRF is an easy and secure way to get a tamper-proof RNG. A less secure solution which still increases the difficulty to execute this exploit is to use multiple parameters like block timestamp and block hash to generate the tokenURI randomly at mint time.

5 References

[1] Thrasher66099, NTLandDeploy Contract, **2021**, https://etherscan.io/address/0x3C54b798b3aAD4F6089533aF3b (visited on 01/09/2022).