Reproducing the $APE Airdrop Flash Loan Arbitrage/Exploit

[Amber Group](#)

[Follow](#)

Amber Group

--

1

Listen

Share

On March 17th 2022, an airdrop equivalent to 15% of the total supply of ApeCoin was distributed to BAYC/MAYC owners [1]. To claim the airdropped ApeCoin, a BoredApe owner had to actively interact with the airdrop smart contract. This surge in interactions led the gas price of Ethereum to climb up to ~400 gwei, around 10 times the average gas price at the time. Five minutes after the airdrop was initiated [2], one well-prepared claimer [3] leveraged the BAYC liquidity on NFTX [4] for a pretty clever arbitrage/exploit [5]. Since this is likely the first flash loan arbitrage/exploit executed with ERC721s and an NFT AMM DEX, we've to reproduced and explained it in this blog.

0x00: AirdropGraphsToken Contract

A Bored Ape owner was able to invoke the claimTokens()

function of the AirdropGraphsToken contract [6] to claim the free ApeCoin. As shown in the code snippet below, alpha.balanceOf()

and beta.balanceOf()

are checked in line 105 to ensure that the caller is really a BAYC/MAYC owner. Later on, a checklist is used to mark the claimed tokenId (line 114–117) such that the getClaimableTokenAmountAndGammaToClaim()

function in line 110 would return only available amounts of ApeCoin to be claimed by the caller.

If you have claimed other airdrops, you may have noticed that it's not common practice to check the eligibility of a claimTokens()

caller through those balanceOf()

calls. In fact, many projects generate a merkle tree and commit only the merkle root onto the blockchain to achieve that. The balanceOf() calls enables a bad actor to borrow an Ape just for claiming the free ApeCoin. If the claimed ApeCoin is valued at more than the borrowing fee, the bad actor makes profit from nothing.

0x01: NFTX

With respect to ERC20s, a lot has been learned from hackers since early 2020 regarding the way flash loans can be used to leverage assets for personal gains. However, it's not common for ERC721s. NFTX is one of the few platforms which allows users to flash loan ERC721s for certain purposes. Specifically, the flashLoan()

function in the NFTVault contract allows a user to borrow an arbitrary amount of vToken which can be used to redeem the underlying NFT [7]. As shown in the code snippets below, the "amount" of vToken minted in line 998 is followed by the execution of the onFlashLoan()

callback function in line 999. When onFlashLoan() returns, the “amount + fee” vToken is burned in line 1003, which requires the caller of flashLoan() to prepare a little bit more than the borrowed vToken.

So, what can we do with vTokens? Well, if the BAYC NFTXVault has 10 Apes as liquidity, a user could 1) flash borrow 10 BAYC vTokens, 2) redeem 10 Apes, 3) claim the ApeCoins, 4) mint 10 BAYC vTokens, and 5) pay 10 BAYC vTokens back with the fee. Besides the current zero flash loan fee, NFTX charges for redeeming and minting. You need to pay 1.04 BAYC vTokens to redeem 1 BAYC NFT. After paying for 1 BAYC NFT, you only get 0.9 BAYC vTokens back. It is therefore important to carefully arrange the amount of tokens used for borrowing.

0x02: Exploiting the flawed airdrop logic

Let’s start by doing the math. From analyzing the arbitrage/exploit transaction, we know that 5 vTokens were burned to redeem 5 Apes. These steps actually burned 5.2 vTokens. Later on, 6 Apes (including the pre-owned ape) were sent to the NFTXVault to mint 5.4 vTokens. After paying the 5.2 loaned vTokens, 0.2 vTokens were swapped into ~14 ETH at SushiSwap.

Based on the aforementioned information, we can reproduce the exploit by purchasing some BAYC vTokens on SushiSwap and using those vTokens as redemption/minting fees. However, there are only ~0.9 BAYC vTokens in the SushiSwap WETH-BAYC pool, which means we need to pay a huge amount of ETH to buy vTokens because of how AMMs work. Hence the bad actor chose to spend 106 ETH to purchase the 1060 BAYC from OpenSea [8] to use it for redemption/minting fees.

With the 1060 BAYC in the Exp contract, we can redeem all available Apes from the NFTVault by calling the NFTVault.flashLoan()

as follows:

Since each redemption consumes 1.04 vTokens, the amount of vTokens we need to loan would be 1.04 times the number of Apes owned by the vault (line 56–57).

Inside the onFlashLoan()

callback function, we firstly get the list of Apes owned by the vault with the tokenOfOwnerByIndex()

helper (line 84–89). Then, we perform theredeem()

, claimTokens()

, and mint()

operations step-by-step as follows:

At the end of onFlashLoan()

, the execution returns to ourtrigger()

function with 0.2 vTokens balance. In the bottom-half of trigger(), we exchange all vTokens to ether at SushiSwap and collect those ETH and the claimed APE at the owner address as shown in the code snippets below.

Here comes the screenshot of simulating the arbitrage/exploit with eth-brownie+ganache:

The exploiter paid 106 ETH and walked away with 14.15 ETH + 60,564 APE.

0x03: Aftermath

According to CoinMarketCap, APE was trading above $10 during 3/18~3/20 [9] and ETH was around $2,760 when the exploiter purchased the 1060 BAYC. So, the profit comes out to (($10 * 60564) — ($2,760 * (106–14.15))) = ~ $350k.

With the help of our in-house blockchain data analytics platform, we identified 8,647 of 10k BAYCs as having been used to claim the free ApeCoin as of the end of Mar 21, 2022. This means at the time of writing, one can still collect some Apes, claim the ApeCoin airdrop, and make a profit.

References

1. https://twitter.com/apecoin/status/1504201556165644298
2. https://etherscan.io/tx/0xb4fa6d557d55183a8c583fa7d24b9f4916c1322a5253383d4f335255a2cfd729
3. https://twitter.com/wilburforce_/status/1504437189979119622
4. https://nftx.io/
5. https://blocksecteam.medium.com/the-short-analysis-of-the-flashloan-attack-to-the-ape-airdrop-490a7d6a1479
6. https://etherscan.io/address/0x025c6da5bd0e6a5dd1350fda9e3b6a614b205a1f
7. https://docs.nftx.io/tutorials/redeeming
8. https://etherscan.io/tx/0x38e5465f09999c815adc3a5e878ec62fb669abed0144323401481c7b1f4cc376
9. https://coinmarketcap.com/currencies/apecoin-ape/