Alchemy token audit by Mudit Gupta

Objective of the audit

The audit's focus was to verify that the smart contract is secure, resilient, and working according to its specifications. The audit activities can be grouped in the following three categories:

Security: Identifying security related issues within each contract and the system of contracts.

Sound Architecture: Evaluation of this system's architecture through the lens of established smart contract best practices and general software best practices.

Code Correctness and Quality: A full audit of the contract source code.

This smart contract in scope of audit was Alchemy.sol

Findings

During the audit, 0 Critical, 2 Major, and 4 Minor issues were found.

- A critical issue represents something that can be relatively easily exploited and will likely lead to loss of funds.
- A major issue represents something that can result in an unintended behavior of the smart contracts. These issues can also lead to loss of funds but are typically harder to execute than critical issues.
- A minor issue represents an oddity discovered during the audit. These issues are typically situational or hard to exploit.

Major

1. The `NFTDAOMint` function drips 100 Alc tokens regardless of the price of the token

Description

The `NFTDAOMint` function is programmed to give 100 Alc token to its caller to compensate for their GAS. However, it's a static number with no limits and therefore if the gas prices go down or Alc prices go up, a bot can empty the contract by calling this function multiple times.

2. Possible to have multiple NFTs with same tokenId

Description

The `addNft` function does not check if the `tokenId` is already used or not.

Minor

3. Unnecessary calculations

Description

The `balanceOf` function has unnecessarily multiplies and divides the balance with total supply.

4. Possible underflow/overflow in edge cases

Description

Missing SafeMath on L172, L173. These calculations are unlikely to overflow/underflow due to previous checks though.

5. Missing test cases

Description

There are very limited test cases that do not cover even the basic functionality. It's recommended to add more test cases.

6. Using non Standard ERC20 implementation

Description

It's recommended to use standard, audited ERC20 implementation to reduce the risk of hidden bugs.

Disclaimer

This report is not an endorsement or indictment of any particular project or team, and the report do not guarantee the security of any particular project. This report does not consider, and should not be interpreted as considering or having any bearing on, the potential economics of a token, token sale or any other product, service or other asset. Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty. This report does not provide any warranty or representation to any Third-Party in any respect, including regarding the bugfree nature of code, the business model or proprietors of any such business model, and the legal compliance of any such business. No third party should rely on this report in any way, including for the purpose of making any decisions to buy or sell any token, product, service or other asset. Specifically, for the avoidance of doubt, this report does not constitute investment advice, is not intended to be relied upon as investment advice, is not an endorsement of this project or team, and it is not a guarantee as to the absolute security of the project. I owe no duty to any Third-Party by virtue of publishing these Reports.

The scope of my audit is limited to a audit of Solidity code and only the Solidity code noted as being within the scope of the audit within this report. The Solidity language itself remains under development and is subject to unknown risks and flaws. The audit does not extend to the compiler layer, or any other areas beyond Solidity that could present security risks. Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty.

This audit does not give any warranties on finding all possible security issues of the given smart contracts, i.e., the evaluation result does not guarantee the nonexistence of any further findings of security issues. As one audit cannot be considered comprehensive, I always recommend proceeding with several independent audits and a public bug bounty program to ensure the security of smart contracts.