dClimate Audit

Source: https://github.com/dClimate/dclimate-monorepo/tree/dev/packages/hardhat

1. _metadata not used in Assets.sol `publish`

Although the "_metadata" input parameter is used in the event logging of the `publish` function of Assets.sol, it is not stored in a state variable on chain or used for other operations in this function. Removing this input parameter could save gas.

Proof of concept/Steps to Reproduce

The unused input parameter:
https://github.com/dClimate/dclimate-monorepo/blob/78797283b2e8da0cfa363c3eb364d277bde64ff3/packages/hardhat/contracts/Assets.sol#L191

Impact

Gas savings

Recommendation

Remove the "*metadata" input parameter from the* `publish` *function. The* "metadataNFT" IPFS address, which is stored on chain, contains the necessary metadata.

2. No checks on _skillScore <= 5.000

Documentation states the "_skillScore" value should remain between 0 and 5.000 in the following text:

> `Upon dClimate DAO's trigger, `updateSkillScore()` will be called via the `SkillscoreClient` contract which will update the associated skillscore of an asset ranging from 0 <= x <= 5.000 based on a transparent script.`

But there are no checks to verify the skillShare value remains in this range and the `fulfillupdateSkillScore`. In the `fulfillupdateSkillScore` function of SkillscoreClient.sol, which calls `updateSkillScore` in Assets.sol, there should be a require statement to validate the skillScore value remains in this range.

Impact

Medium

Recommendation

Add a require statement to limit skillShare values:
`require(skillShare <= 5000, "skillShare value outside limits")`

3. Use onlyGovernance modifier consistently

The `setGovernance` function of Providers.sol has a require statement to check that msg.sender == governance, but other functions in the contract use the "onlyGovernance" modifier for the same purpose.

Proof of concept/Steps to Reproduce

This code repeats the result of using the "onlyGovernance" modifier: https://github.com/dClimate/dclimate-monorepo/blob/78797283b2e8da0cfa363c3eb364d277bde64ff3/packages/hardhat/contracts/Providers.sol#L80

Impact

Informational

Recommendation

Use the onlyGovernance modifier instead of adding individual `require(msg.sender == governance)` statements to functions

4. Deprecated `_setupRole` function used

The comments in the OpenZeppelin AccessControl.sol contract include a statement about the `_setupRole` function stating `NOTE: This function is deprecated in favor of {_grantRole}`. The `_setupRole` function is used in the Providers.sol constructor. Unless there is a good reason to wait, it would be best to migrate to `_grantRole` sooner than later.

Proof of concept/Steps to Reproduce

`_setupRole` call: https://github.com/dClimate/dclimate-monorepo/blob/78797283b2e8da0cfa363c3eb364d277bde64ff3/packages/hardhat/contracts/Providers.sol#L71

OpenZeppelin deprecated comment: https://github.com/OpenZeppelin/openzeppelin-contracts/blob/c239e1af8d1a1296577108dd6989a17b57434f8e/contracts/access/AccessControl.sol#L195

Impact

Informational, possible gas savings

Recommendation

Use non-deprecated functions to avoid future errors with dependencies

5. Missing checks can lead to owner rugging assets

The `slashAsset` function has the "onlyGovernance" modifier, but this is the only check from preventing any asset in the probationary period from getting slashed. Malicious governance or a rough insider could lead to all users getting rugged. Although the governance team may consider themselves trusted actors, the function as it is currently written may not give users confidence in the safety of their assets.

Proof of concept/Steps to Reproduce

Impact

Medium, depending on trust assumptions

Recommendation

Instead of allowing governance to immediately slash assets, user trust could be improved by requiring the asset to be slashed must pass a test of confirmed malicious intent. This could be done by a vote of trusted parties, but a more robust long-term approach could borrow from the solution of how pBFT (Practical Byzantine Fault Tolerance) replaces a leader that is no longer trusted.

6. Providers.sol missing _revokeRole function

The `whitelistProviderCreator` function of Providers.sol does not have a corresponding function that removes addresses from the whitelist. The inability to revoke the provider role could make it difficult to stop an address that suddenly goes rogue.

Proof of concept/Steps to Reproduce

A call to `grantRole` exists in Providers.sol: https://github.com/dClimate/dclimate-monorepo/blob/78797283b2e8da0cfa363c3eb364d277bde64ff3/packages/hardhat/contracts/Providers.sol#L116

But no corresponding `_revokeRole` call exists in any function of the dClimate project.

Impact

Medium, depending on trust assumptions

Recommendation

Instead of allowing governance to immediately slash assets, user trust could be improved by requiring the asset to be slashed must pass a test of confirmed malicious intent. This could be done by a vote of trusted parties, but a more robust long-term approach could borrow from the solution of how pBFT (Practical Byzantine Fault Tolerance) replaces a leader that is no longer trusted.

7. Providers.sol `setGovernance` function doesn't transfer "DEFAULT_ADMIN_ROLE" role

The Providers.sol constructor sets msg.sender as the governance role and calls `_setupRole` to grant msg.sender the "DEFAULT_ADMIN_ROLE" role. The `setGovernance` function sets the "governance" variable to a new address, but the "DEFAULT_ADMIN_ROLE" role assignment is not changed in this function.

Proof of concept/Steps to Reproduce

The call to `_setupRole` is at:
https://github.com/dClimate/dclimate-monorepo/blob/78797283b2e8da0cfa363c3eb364d277bde64ff3/packages/hardhat/contracts/Providers.sol#L71

But the "DEFAULT_ADMIN_ROLE" role is not transferred in the `setGovernance` function:
https://github.com/dClimate/dclimate-monorepo/blob/78797283b2e8da0cfa363c3eb364d277bde64ff3/packages/hardhat/contracts/Providers.sol#L78

Impact

Low, only because I don't see any existing functions where "DEFAULT_ADMIN_ROLE" gives special privileges, but later code changes that rely on the `onlyRole(DEFAULT_ADMIN_ROLE)` modifier could elevate this to high risk

Recommendation

Add these two lines to the end of the `setGovernance` function:

```
_setRoleAdmin(DEFAULT_ADMIN_ROLE, _governance)
_revokeRole(DEFAULT_ADMIN_ROLE, msg.sender)
```

Because these functions emits their own events, the emit can be removed from `setGovernance`

8. Leak of symmetric key can allow "data piracy"

The current architecture describes a process whereby paid data is uploaded to IPFS in encrypted form and the customer purchasing the data receives a symmetric encryption key to decrypt the data. Using a symmetric key to encrypt the data and sharing the same key across customers means that it is not possible to prevent the symmetric key from being shared with parties who have not paid for the data. In theory, one data customer

could pay for the key, then sell the key at half price to other users of the data.

Impact

High, possible bypass of publisher profit model if user expends effort

Recommendation

Providing each data user with their own key, whether that is a separate symmetric key or using public key encryption, could prevent this at the cost of uploading many different encrypted copies of the data. Another option is to issue an ERC721 to each customer that acts as an authentication token for the token owner. Unfortunately this type of fraud is hard to stop and can require entire fraud departments to handle at larger corporations that are common targets.

A convention web2 solution might be to require an API key to start a data streaming session, with the server limiting each API key to one active session, but I don't think this type of solution exists with web3.

9. No disincentive to prevent rogue oracles after probationary period

The primary disincentive that nudge providers to provide correct data is the `slashAsset` function in Assets.sol and the "skillScore" state variable, which is periodically set by governance. If a malicious oracle provides correct data for the stakeGracePeriod but then provides incorrect data for some time period after, I don't see any major penalty to this malicious party other than an eventual change of "skillScore". The only instance of the code `delete assets[_assetId]` is in the `slashAsset` function, so the asset cannot be deleted after the grace period ends.

Impact

Medium

Recommendation

A long-term mitigation to handle rogue providers might help. That could involve an admin function controlled by governance to call `delete assets[_assetId]` with a timelock, or that could involve special handling of assets with a skillScore of zero.

10. NFTdClim "unnecessary"?

There is a lot of overlap between the assets mapping in Assets.sol and the NFTdClim.sol ERC721 contract. Minor edits could remove the need for NFTdClim entirely. nftdclim is only used once in Marketplace.sol while Assets.sol only uses it to store the metadata URI.

However, if selling the NFT on OpenSea is considered one of the defining features of dClimate, it may be possible to more heavily rely on the NFTdClim ERC721 to replace some of the places where the assets mapping is used.

Impact

Informational

Recommendation

Consider removing the NFTdClim.sol ERC721 contract and relying on the assets state variable in Assets.sol

11. Order history checking not needed

order.buyer in L308 of Marketplace.sol already exists as a method of tracking buyers, gas savings are available if we remove the mapping at L86. This can also be removed at Publisher.sol L91

The following two lines are storing data in state variable for no apparent reason besides logging purposes. This data could be logged in an emit event instead, which could lead to gas savings.
https://github.com/dClimate/dclimate-monorepo/blob/78797283b2e8da0cfa363c3eb364d277bde64ff3/packages/hardhat/contracts/Marketplace.sol#L301-L302

12. SkillScore is not used in other parts of the contract

A specific governance address is currently allowed to execute all SkillScore decisions. This can be a potential attack vector. Also, assuming that each "Asset" has a SkillScore, unless there is a limited number of "Assets" it may be hard to set a skill score for all of them by governance. SkillScore is an unused variable besides for being a part of Asset specific metadata.

Impact

Informational

13. Hardcode chainlink contracts & check for job requests handling

On certain chains (such as Polygon), jobs may not be handled properly by Chainlink oracles. Chainlink oracles and LINK token addresses also vary by chain

Impact

Informational.

14. Typos

Comment in NFTdClim.sol "IPFS address including the metadat" -> metadat should be metadata
https://github.com/dClimate/dclimate-monorepo/blob/78797283b2e8da0cfa363c3eb364d277bde64ff3/packages/hardhat/contracts/NFTdClim.sol#L36

Impact

Informational

Recommendation

Fix typos

End of Findings

Misc Chainlink Notes

Link to Chainlnk Data Feed docs: https://docs.chain.link/docs/using-chainlink-reference-contracts/

- There are 4 main functions of Chainlink: data feeds (bringing off-chain data on-chain in a decentralized way), VRF (random number generator, which is hard to do on chain), Chainlink keepers (different from the Keep3r network used by Yearn Finance), and Chainlink APIs (connect to existing external web2 APIs)
- Chainlink Data Feeds use the popular decentralized data model, but Chainlink APIs use a 1-to-1 basic request process, essentially an API request to a web2 API
- The contract or EOA requesting Chainlink data must pay LINK for the data

- Data sources are listed on Chainlink Marketplace. Data providers advertise here to get users (and LINK fees from those users). This site lets you view nodes, data feeds, etc.: https://market.link/search/nodes
- Chainlink operates on many chains (ETH, MATIC, BNB, xDAI, etc.)
- Practically all the big name DeFi apps use Chainlink Data Feeds for price data, and price feeds is currently the main use case for Chainlink
- The Chainlink oracle network data consists of "oracle operators" running "oracle nodes" that send data to an "aggregator". The nodes that work together to provide a certain data feed are in the same "oracle network". Data is updated in "rounds" (which is a non-sequential counter). An overview of running a Chainlink node is at: https://blog.chain.link/what-is-a-chainlink-node-operator/
- In order for the Chainlink data to update, a minimum number of nodes must respond. Different data feeds have different properies, so this minimum value depends on the data feed
- Because the many Chainlink nodes may provide slightly different results, an on-chain FluxAggregator contract handles the multiple node data points. It is possible the data provided does not result in a consensus, in which case the data won't update that round. The "Deviation threshold" and "Heartbeat threshold" parameters impact the data aggregation process. The former triggers a data update with a certain amount of change in the data, while the latter triggers an update after a certain amount of time since the last update. The values of these parameters can be found for different feeds: https://data.chain.link/
- Each data point is associated with a roundId, so by querying old/existing roundId values, historical data can be queried
- The FluxAggregator approach is being phased out (or already is?) in favor of a new Off-Chain Reporting (OCR) process, which improves scalability while using less gas for data providers and data consumers. An explanatory whitepaper exists: https://docs.chain.link/docs/off-chain-reporting/ At a high level, OCR lets nodes reach a consensus off-chain, and a transmitter migrates the result on-chain. OCR allows up to ⅓ of nodes to fail or be malicious, like the pBFT consensus algorithm.
- engn33r note #1: Trusting the data from Chainlink without verifying whether it is stale (hasn't updated in a while) is a known issue. The round number and timestamp of the Chainlink data should be checked to prevent (re)use of old data: https://consensys.net/diligence/audits/2021/09/fei-protocol-v2-phase-1/#chainlinkoraclewrapper-latestrounddata-might-return-stale-results Some data

feeds may provide low quality or manipulated data, especially if there is only one data provider for the data feed you are using https://docs.chain.link/docs/selecting-data-feeds/

- engn33r note #2: There are different versions of Chainlink contracts. AggregatorV3Interface (for data feeds) and VRFv2 (for random numbers) are examples of the latest releases, but deprecated Chainlink functions are often called out in audit reports