High

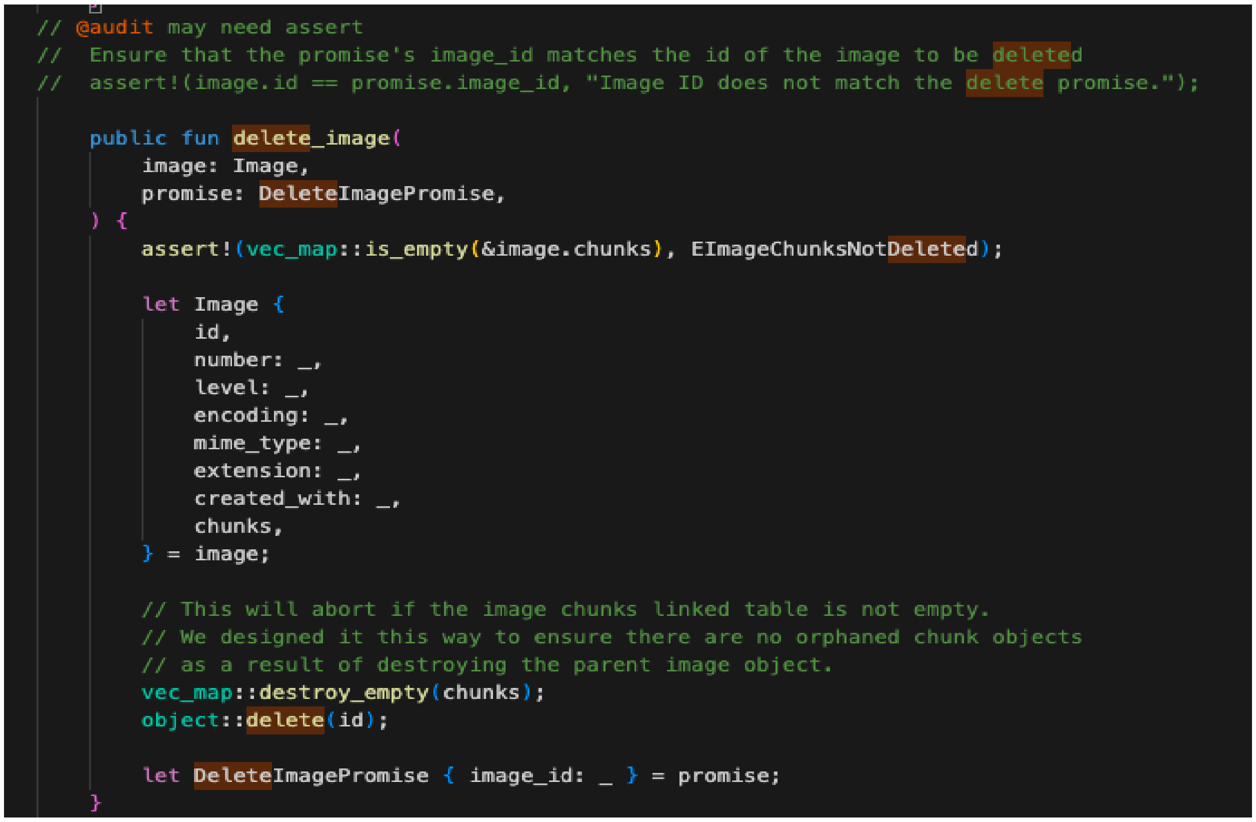
Impact

Data Integrity: Mismatched IDs could result in the wrong image being deleted, leading to loss of important data and affecting the consistency of the system's state.

System Reliability: Unintended deletions could undermine the reliability of the system, as users and administrators may lose trust in the system's ability to manage image assets correctly.

Security: If the mechanism to match image IDs is not robust, it could potentially be exploited to maliciously delete images, posing a security risk.

Fix



```
// @audit may need assert
//  Ensure that the promise's image_id matches the id of the image to be deleted
//  assert!(image.id == promise.image_id, "Image ID does not match the delete promise.");

    public fun delete_image(
        image: Image,
        promise: DeleteImagePromise,
    ) {
        assert!(vec_map::is_empty(&image.chunks), EImageChunksNotDeleted);

        let Image {
            id,
            number: _,
            level: _,
            encoding: _,
            mime_type: _,
            extension: _,
            created_with: _,
            chunks,
        } = image;

        // This will abort if the image chunks linked table is not empty.
        // We designed it this way to ensure there are no orphaned chunk objects
        // as a result of destroying the parent image object.
        vec_map::destroy_empty(chunks);
        object::delete(id);

        let DeleteImagePromise { image_id: _ } = promise;
    }
```

Validation Checks:

Implement strict validation checks in the delete_image function to ensure that the image_id in the DeleteImagePromise matches the id of the Image object before proceeding with the deletion.

Summary

The relationship between the image_id in DeleteImagePromise and the id in the Image object is crucial for the safe deletion of images within the system. Ensuring these IDs match is essential for maintaining data integrity and system reliability. By implementing robust validation checks, enhanced logging and monitoring, effective error handling, and strict access controls, the system can safeguard against unintended deletions and enhance overall security and trustworthiness.

Informational

- admin_add_to_mint_warehouse**:**

//@audit Concern: This function centralizes the power to add NFTs to the mint warehouse, which could be a risk if administrative access is compromised. Consider implementing decentralized mechanisms or multi-signature requirements for critical actions to distribute this power and mitigate risks.

- admin_add_to_migration_warehouse**:**

//@audit Concern: Similar to mint warehouse management, this function gives administrators significant control over the migration process. It is vital to ensure strict access controls and possibly introduce checks or balances like multi-sig approvals to reduce the risk of misuse.

- admin_destroy_migration_warehouse **and** admin_destroy_mint_warehouse**:**

//@audit Concern: The destruction of warehouses should be handled with extreme caution to prevent accidental loss of data or assets. Implement additional checks, confirmation steps, or a delay mechanism to allow for audit and reversal in case of an error or malicious action.

- admin_set_mint_phase**,** admin_set_mint_price**, and** admin_set_mint_status**:**

//@audit Concern: These functions allow administrators to directly influence the minting process's economic and operational parameters. To avoid centralization risks and potential manipulation, consider introducing governance mechanisms where such critical parameters can be voted on by the community or a group of stakeholders.

- admin_issue_migration_ticket **and** admin_issue_whitelist_ticket**:**

//@audit Concern: Issuing tickets is a sensitive operation that can affect the fairness and transparency of the minting and migration processes. Implementing transparent criteria and logs for ticket issuance, along

with oversight mechanisms, can help mitigate potential abuses of power.
admin_add_to_mint_warehouse**:**
//@audit Concern: This function centralizes the power to add NFTs to the mint warehouse, which could be a risk if administrative access is compromised. Consider implementing decentralized mechanisms or multi-signature requirements for critical actions to distribute this power and mitigate risks.

- admin_add_to_migration_warehouse**:**

//@audit Concern: Similar to mint warehouse management, this function gives administrators significant control over the migration process. It is vital to ensure strict access controls and possibly introduce checks or balances like multi-sig approvals to reduce the risk of misuse.

- admin_destroy_migration_warehouse **and** admin_destroy_mint_warehouse**:**

//@audit Concern: The destruction of warehouses should be handled with extreme caution to prevent accidental loss of data or assets. Implement additional checks, confirmation steps, or a delay mechanism to allow for audit and reversal in case of an error or malicious action.

- admin_set_mint_phase**,** admin_set_mint_price**, and** admin_set_mint_status**:**

//@audit Concern: These functions allow administrators to directly influence the minting process's economic and operational parameters. To avoid centralization risks and potential manipulation, consider introducing governance mechanisms where such critical parameters can be voted on by the community or a group of stakeholders.

- admin_issue_migration_ticket **and** admin_issue_whitelist_ticket**:**

//@audit Concern: Issuing tickets is a sensitive operation that can affect the fairness and transparency of the minting and migration processes. Implementing transparent criteria and logs for ticket issuance, along with oversight mechanisms, can help mitigate potential abuses of power.

- admin_refund_mint **and** admin_reveal_mint**:**

//@audit Concern: Refund and reveal operations are critical and sensitive, carrying the risk of financial loss or revealing incorrect information. These functions should have strict validation checks and perhaps a secondary confirmation to ensure their correctness and prevent abuse.

- admin_refund_mint **and** admin_reveal_mint**:**

//@audit Concern: Refund and reveal operations are critical and sensitive, carrying the risk of financial loss or revealing incorrect information. These functions should have strict validation checks and perhaps a secondary confirmation to ensure their correctness and prevent abuse.

The audit concerns highlight the risk of centralizing too much power in single administrative functions, which can lead to vulnerabilities if an admin address gets compromised. Implementing decentralized mechanisms, such as multi-signature requirements and checks or balances, is essential to mitigate these risks. These mechanisms ensure that no single entity has complete control over critical actions like adding NFTs to warehouses, managing the minting process, or executing refunds and reveals. Additionally, introducing governance mechanisms for setting important parameters and transparent criteria for ticket issuance can help in distributing power, increasing transparency, and preventing potential abuses. This approach not only enhances security but also promotes fairness and trust in the system by ensuring that critical actions are subject to collective decision-making and oversight.

Issue: The DeleteImagePromise and the Image object have separate image_id references, which need to match to ensure the correct image is being targeted for deletion. There is a potential risk that mismatches between these IDs could lead to unintended image deletions, affecting data integrity and system reliability.