ChainLight

Emblem Vault Security Audit

：Emblem Vault v2 Solidity Contracts

Mar 3, 2025

Revision 1.0

ChainLight@Theori

Table of Contents

Executive Summary

Beginning on January 31, 2025, ChainLight of Theori conducted a two-week security audit of Emblem Vault's v2 Solidity contracts. The primary goal of the audit was to identify critical security vulnerabilities and evaluate potential impacts.

Note: Scenarios involving compromised trusted witnesses or off-chain backend systems were excluded from the scope of this audit.

Summary of Findings

The audit revealed a total of nine issues, categorized by severity as follows:

- **Critical:** 1 issue (A critical field is missing from the signature)
- **High:** 1 issue (Minting may be possible without fees)
- **Medium:** 1 issue (Unreferenced metadata corruption)
- **Low:** 2 issues
- **Informational:** 4 issues

Audit Overview

Scope

Name	Emblem Vault Security Audit
Target / Version	• Git Repository (EmblemCompany/V2_SolidityContracts): commit `d9737534f4f079970e8412930eee8dfbde3db33a`
Application Type	Smart contracts
Lang. / Platforms	Smart contracts [Solidity]

Code Revision

N/A

Severity Categories

Severity	Description
Critical	The attack cost is low (not requiring much time or effort to succeed in the actual attack), and the vulnerability causes a high-impact issue. (e.g., Effect on service availability, Attacker taking financial gain)
High	An attacker can succeed in an attack which clearly causes problems in the service's operation. Even when the attack cost is high, the severity of the issue is considered "high" if the impact of the attack is remarkably high.
Medium	An attacker may perform an unintended action in the service, and the action may impact service operation. However, there are some restrictions for the actual attack to succeed.
Low	An attacker can perform an unintended action in the service, but the action does not cause significant impact or the success rate of the attack is remarkably low.
Informational	Any informational findings that do not directly impact the user or the protocol.
Note	Neutral information about the target that is not directly related to the project's safety and security.

Status Categories

Status	Description
Reported	ChainLight reported the issue to the client.
WIP	The client is working on the patch.
Patched	The client fully resolved the issue by patching the root cause.
Mitigated	The client resolved the issue by reducing the risk to an acceptable level by introducing mitigations.
Acknowledged	The client acknowledged the potential risk, but they will resolve it later.
Won't Fix	The client acknowledged the potential risk, but they decided to accept the risk.

Finding Breakdown by Severity

Category	Count	Findings
Critical	1	• EMBLEMVAULT-001
High	1	• EMBLEMVAULT-003
Medium	1	• EMBLEMVAULT-006
Low	2	• EMBLEMVAULT-004 • EMBLEMVAULT-005
Informational	4	• EMBLEMVAULT-002 • EMBLEMVAULT-007 • EMBLEMVAULT-008 • EMBLEMVAULT-009
Note	0	• N/A

Findings

Summary

#	ID	Title	Severity	Status
1	EMBLEMVAULT-001	`serialNumber` is not included in the signature used in `buyWithSignedPrice()`	Critical	Patched
2	EMBLEMVAULT-002	`mintWithSerial()` Should Use a Consistent Encoding Scheme for Varying Serial Number Counts	Informational	Patched
3	EMBLEMVAULT-003	`mintWithSerial()` May Allow Minting ERC1155 Tokens Without Fees Due to Missing Zero-Amount Check	High	Patched
4	EMBLEMVAULT-004	Direct burns of tokens must be restricted to prevent freeze of funds	Low	Patched
5	EMBLEMVAULT-005	Users Are Not Refunded for Overpayment	Low	Patched
6	EMBLEMVAULT-006	`_tokenSerials` Mapping Can Be Corrupted in `ERC1155VaultImplementation`	Medium	Patched
7	EMBLEMVAULT-007	Missing EIP-165 Check in `burnRouter()`	Informational	Patched
8	EMBLEMVAULT-008	Incorrect Interface ID Defined for `IIsSerialized`	Informational	Patched
9	EMBLEMVAULT-009	Minor Suggestions	Informational	Patched

#1 `EMBLEMVAULT-001` `serialNumber` is not included in the signature used in `buyWithSignedPrice()`

ID	Summary	Severity
`EMBLEMVAULT-001`	The `serialNumber` is not included in the witness signature required for the `buyWithSignedPrice()` function of `EmblemVaultMintFacet`, potentially enabling attackers to steal funds.	**Critical**

Description

The witness signature required for the `buyWithSignedPrice()` function in `EmblemVaultMintFacet` does not include the `serialNumber`. This allows attackers, who have acquired a valid signature and its corresponding message, to modify the `serialNumber`. As a result, an attacker can mint tokens linked to vaults of higher value than originally authorized. Subsequently, when legitimate vault owners attempt to mint tokens, their transactions fail with an error such as `SerialNumberAlreadyUsed`.

Impact

Critical

Attackers can mint tokens tied to vaults of higher value than authorized. However, exploiting this vulnerability at scale for significant profit could be challenging in practice, as vaults already associated with minted tokens are not vulnerable.

Recommendation

Include the `serialNumber` in the witness signature.

Remediation

Patched

The issue has been patched as recommended.

#2 `EMBLEMVAULT-002` `mintWithSerial()` Should Use a Consistent Encoding Scheme for Varying Serial Number Counts

ID	Summary	Severity
`EMBLEMVAULT-002`	The `mintWithSerial()` function in `ERC1155VaultImplementation` decodes single-serial and multi-serial `serialNumberData` differently. This increases the complexity for the caller's implementation.	Informational

Description

In the `mintWithSerial()` function of `ERC1155VaultImplementation`:

- When handling a single-serial input (mint amount <= 1), the code decodes `serialNumberData` (provided as `bytes`) into a `uint256`.
- When handling a multi-serial input (mint amount > 1), it decodes `serialNumberData` as a `uint256[]`.

A caller might use a `uint256[]` of length 1 for a single-serial scenario. In that case, the function interprets the byte length `0x20` as a value, causing the `serialNumber` to be `0x20` and leading to failures for the single mint from the same caller. If the gas usage difference is not significant, using a unified format is recommended to reduce complexity.

Impact

Informational

This slightly increases complexity and the likelihood of caller implementation errors. In case of an error, the first single-serial mint might succeed with an incorrect serial number, while subsequent mints of the same type could fail.

Recommendation

Update `mintWithSerial()` to consistently decode `serialNumberData` as a `uint256[]`.

Remediation

Patched

The function has been updated to take a serial number argument in a `uint256[]` type rather than `bytes`.

#3 `EMBLEMVAULT-003` `mintWithSerial()` May Allow Minting ERC1155 Tokens Without Fees Due to Missing Zero-Amount Check

ID	Summary	Severity
`EMBLEMVAULT-003`	The `mintWithSerial()` function in `ERC1155VaultImplementation` lacks validation against a zero-value `amount` parameter. This enables attackers to mint ERC1155 tokens without paying fees.	High

Description

The `mintWithSerial()` function within `ERC1155VaultImplementation` does not properly reject transactions with an `amount` parameter of zero. An attacker could exploit this by submitting valid parameters with `amount` set to zero, which might still be signed by a witness if additional off-chain checks are not enforced. The internal `_mintWithSerials()` method treats any `amount` less than or equal to one (`amount <= 1`) as a request for a single token mint, inadvertently allowing an `amount` of zero to result in one token being minted.

This enables an attacker to mint ERC1155 tokens without incurring fees, as the off-chain fee calculation (`price`) might incorrectly compute a total fee of zero due to the zero `amount`.

Impact

High

If off-chain validation does not explicitly reject zero-value `amount` transactions, attackers can mint ERC1155 tokens without paying the required fees.

(Behavior of off-chain components is assumed since they are out of scope.)

Recommendation

Implement a strict check in `mintWithSerial()` to revert the transaction when `amount == 0`. Even if the backend already enforces this validation, on-chain enforcement provides a vital additional safeguard.

Remediation

Patched

It has been patched as recommended.

#4 `EMBLEMVAULT-004` Direct burns of tokens must be restricted to prevent freeze of funds

ID	Summary	Severity
`EMBLEMVAULT-004`	Direct burning of `ERC1155` or `ERC721` tokens must be restricted, as burning them outside the unvaulting process can lead to frozen funds.	Low

Description

If a user calls `burn()` directly on `ERC1155` or `ERC721` tokens, the `EmblemVaultUnvaultFacet`'s `unvault` process will fail for those. This is because the checks and burn operation during unvaulting require the token to exist and be owned by the caller. While recovery is possible, it requires either a contract upgrade or an unusual signature from a witness.

Impact

Low

A user would have to initiate the action and the damage is isolated to themselves, though it is relatively easy to trigger. Recovery is possible.

Recommendation

Restrict the `burn()` function so it can only be invoked within the unvault workflow.

Remediation

Patched

It has been patched as recommended.

#5 `EMBLEMVAULT-005` Users Are Not Refunded for Overpayment

ID	Summary	Severity
`EMBLEMVAULT-005`	The `EmblemVaultMintFacet` contract lacks mechanisms to refund excess funds in the `batchBuyWithSignedPrice()` and `_processMint()` functions. As a result, any overpayment remains locked in the contract.	Low

Description

Within the `EmblemVaultMintFacet` contract, both `batchBuyWithSignedPrice()` and `_processMint()` do not validate whether the user sends more ETH than required. They also do not return any surplus amount, causing any excess to stay locked in the contract. However, these locked funds are not lost, and can be recovered through a contract upgrade.

Impact

Low

Users who overpay will have their extra funds locked in the contract. Although the amount cannot be spent by others, it remains inaccessible unless it is rescued via a contract upgrade.

Recommendation

Implement a mechanism to refund the remainder if users send more ETH than needed.

Remediation

Patched

A check requiring the exact payment amount has been added.

#6 EMBLEMVAULT-006 _tokenSerials Mapping Can Be Corrupted in ERC1155VaultImplementation

ID	Summary	Severity
EMBLEMVAULT-006	Due to the way it is managed, the _tokenSerials mapping in ERC1155VaultImplementation can be incorrectly overwritten.	Medium

Description

Within ERC1155VaultImplementation, serial numbers stored in _tokenSerials can be overwritten if the same token ID is minted to multiple addresses. Each address has its own _ownerTokenSerials[owner][tokenId].length, starting at 0, which is used as an index in the second dimension of _tokenSerials. This setup can cause unintended overwriting of existing serial numbers for a given token ID.

Impact

Medium

Although _tokenSerials is only accessed by getSerial(), and getSerial() is not called anywhere within the current codebase, there could still be a financial impact if an external contract or off-chain component relies on this function.

Recommendation

Prevent overwriting by adding the owner address as a key in the _tokenSerials mapping. Update the relevant code accordingly, for example:

```
_tokenSerials[tokenId][owner][index] = serialNumber;
```

Remediation

Patched

The _tokenSerials mapping has been deprecated in favor of a _serialOwners mapping.

#7 `EMBLEMVAULT-007` Missing EIP-165 Check in `burnRouter()`

ID	Summary	Severity
`EMBLEMVAULT-007`	The `burnRouter()` function in the `EmblemVaultUnvaultFacet` contract invokes methods from the `IIsSerialized` interface without verifying interface support via EIP-165.	Informational

Description

Within `EmblemVaultUnvaultFacet`'s `burnRouter()`, the contract calls functions defined by the `IIsSerialized` interface but does not check whether the target contract actually supports this interface. (There is no explicit EIP-165 check.) As a result, the transaction may revert without providing a clear error message.

Impact

Informational

Interacting with NFT collections that do not implement `IIsSerialized` would cause the transaction to revert without a clear error message.

Recommendation

Add an explicit EIP-165 check within `burnRouter()` to verify that the target contract implements `IIsSerialized` before calling its functions.

Remediation

Patched

It has been patched as recommended.

#8 EMBLEMVAULT-008 Incorrect Interface ID Defined for IIsSerialized

ID	Summary	Severity
EMBLEMVAULT-008	The LibInterfaceIds library incorrectly defines the interface ID for IIsSerialized, leading to incorrect results for EIP-165 checks based on it.	Informational

Description

The LibInterfaceIds library incorrectly defines the constant INTERFACE_ID_SERIALIZED for the IIsSerialized interface, causing incorrect results for EIP-165 checks such as isSerialized(). The recommended approach is to use type(IIsSerialized).interfaceId or define a precomputed constant that accurately matches the interface signature.

Impact

Informational

Logic relying on LibInterfaceIds.isSerialized() may fail to detect contracts that implement the IIsSerialized interface. However, this function is currently unused.

Recommendation

Use type(IIsSerialized).interfaceId or a precomputed value of it. For consistency, consider using type(...).interfaceId for all interface ID usages.

Remediation

Patched

It has been patched as recommended.

#9 EMBLEMVAULT-009 Minor Suggestions

ID	Summary	Severity
EMBLEMVAULT-009	The description includes multiple suggestions for preventing incorrect settings caused by operational mistakes, mitigating potential issues, and improving code maturity and readability.	Informational

Description

Operational Risk Mitigation / Sanity Check

1. `LibInterfaceIds.recoverSigner()` should confirm that the signature's `s` is within the lower half of the secp256k1 curve, which is a standard malleability check.
2. For `ERC721` or `ERC721A` tokens, `internalTokenId` becomes `serialNumber` for event emission in `burnRouter()`. Since this may lead to erroneous event parsing, a separate event should be created or it should be documented properly.
3. The `_mintRouter()` and `_batchMintRouter()` functions in `EmblemVaultMintFacet` should return `false` for unrecognized collection types.
4. Similarly, the `burnRouter()` in `EmblemVaultUnvaultFacet` should return `(false, 0, "")` for unrecognized collection types.
5. Add an expiration timestamp field (`deadline`) for witness signatures (`verifyStandardSignature` and `verifyLockedSignature` in `LibSignature`).

Code Maturity

1. In `EmblemVaultUnvaultFacet.burnRouter()`, instead of setting `data = ""` in each branch, make the final return statement `return (true, serialNumber, "")`.
2. `_uintToStrOptimized()` in `EmblemVaultMintFacet` is not referenced and can be removed for cleaner code.
3. `LibEmblemVaultStorage.initializeVaultStorage()` is never used; `EmblemVaultInitFacet.initialize` already covers vault storage initialization.
4. `EmblemVaultCoreFacet` uses a different error handling. Adopting `LibErrors` uniformly would align it with the rest of the codebase.
5. In `ERC1155VaultImplementation.beacon()`, a constant `bytes32 private constant BEACON_SLOT` should be used instead of re-declaring it as a local variable.

Gas Optimizations

1. In `LibDiamond.replaceFunctions()`, a delete operation right before the update can be omitted.
2. In `LibDiamond.removeFunctions()`, instead of decrementing `ds.totalSelectors--` each iteration, use a single subtraction at the end.
3. In `LibDiamond.addFunctions()`, replace `ds.totalSelectors++` in loop with a single addition after the loop.

Impact

Informational

Recommendation

Consider applying the suggestions in the description above.

Remediation

Patched

Most items were patched as recommended.

Revision History

Version	Date	Description
1.0	Mar 2, 2025	Initial version
1.1	Mar 3, 2025	Revised impacts section for certain issues

Theori, Inc. (“We”) is acting solely for the client and is not responsible to any other party. Deliverables are valid for and should be used solely in connection with the purpose for which they were prepared as set out in our engagement agreement. You should not refer to or use our name or advice for any other purpose. The information (where appropriate) has not been verified. No representation or warranty is given as to accuracy, completeness or correctness of information in the Deliverables, any document, or any other information made available. Deliverables are for the internal use of the client and may not be used or relied upon by any person or entity other than the client. Deliverables are confidential and are not to be provided, without our authorization (preferably written), to entities or representatives of entities (including employees) that are not the client, including affiliates or representatives of affiliates of the client.