Arcana Security Review

Pashov Audit Group

Conducted by: pashov

December 16th, 2022

Contents

1. About pashov

Krum Pashov, or **pashov**, is an independent smart contract security researcher. Having found numerous security vulnerabilities in various protocols, he does his best to contribute to the blockchain ecosystem and its protocols by putting time and effort into security research & reviews. Check his previous work here or reach out on Twitter @pashovkrum.

2. Disclaimer

A smart contract security review can never verify the complete absence of vulnerabilities. This is a time, resource and expertise bound effort where I try to find as many vulnerabilities as possible. I can not guarantee 100% security after the review or even if the review will find any problems with your smart contracts. Subsequent security reviews, bug bounty programs and on-chain monitoring are strongly recommended.

3. Introduction

A time-boxed security review of the **Arcana** protocol was done by **pashov**, with a focus on the security aspects of the application's smart contracts implementation.

4. About Arcana

The code was reviewed for a total of 10 hours.

5. Risk Classification

Severity	**Impact: High**	**Impact: Medium**	**Impact: Low**
Likelihood: High	Critical	High	Medium
Likelihood: Medium	High	Medium	Low
Likelihood: Low	Medium	Low	Low

5.1. Impact

- High - leads to a significant material loss of assets in the protocol or significantly harms a group of users.
- Medium - only a small amount of funds can be lost (such as leakage of value) or a core functionality of the protocol is affected.
- Low - can lead to any kind of unexpected behavior with some of the protocol's functionalities that's not so critical.

5.2. Likelihood

- High - attack path is possible with reasonable assumptions that mimic on-chain conditions, and the cost of the attack is relatively low compared to the amount of funds that can be stolen or lost.
- Medium - only a conditionally incentivized attack vector, but still relatively likely.
- Low - has too many or too unlikely assumptions or requires a significant stake by the attacker with little or no incentive.

5.3. Action required for severity levels

- Critical - Must fix as soon as possible (if already deployed)
- High - Must fix (before deployment if not already deployed)
- Medium - Should fix
- Low - Could fix

6. Security Assessment Summary

***review commit hash* - 51fc65fdd6474c9632975294c560ddee24135f2f**

Scope: `ArcanaPrime.sol`

7. Executive Summary

Over the course of the security review, pashov engaged with Arcana to review Arcana. In this period of time a total of **3** issues were uncovered.

Protocol Summary

Protocol Name	Arcana
Date	December 16th, 2022

Findings Count

Severity	**Amount**
High	1
Medium	1
Low	1
Total Findings	**3**

Summary of Findings

ID	Title	Severity	Status
[H-01]	There is no way to withdraw the ETH paid by minters	High	Resolved
[M-01]	If address is a smart contract that can't handle ERC721 tokens they will be stuck after a whitelisted mint	Medium	Resolved
[L-01]	Usage of ecrecover should be replaced with usage of OpenZeppelin's ECDSA library	Low	Resolved

8. Findings

8.1. High Findings

[H-01] There is no way to withdraw the ETH paid by minters

Proof of Concept

There is currently no possible way for the contract deployer/owner to withdraw the ETH that was paid by miners. This means that value will be stuck & lost forever. This is also the case for the `ERC721A` standard, which this project actually extends as well, but it was verified in a conversation with the developer that the `ArcanaPrime` contract is expected to be used as-is, without a need for inheritance/extension.

Impact

This will mean hundreds of thousands of dollars (since `MAX_SUPPLY = 10_000` and `MINT_PRICE = 0.08 ether`) will be irretrievable, essentially drying the runway of the NFT project, so it is High severity.

Recommendation

Add a method to withdraw the value in the contract, for example

```
function withdrawBalance() external onlyOwner {
    (bool success, ) = msg.sender.call{value: address(this).balance}("");
    require(success);
  }
```

8.2. Medium Findings

[M-01] If address is a smart contract that can't handle ERC721 tokens they will be stuck after a whitelisted mint

Proof of Concept

The `mintPublic` method has a check that allows only EOAs to call it

```
if (tx.origin != msg.sender) revert ContractsNotAllowed();
```

but it is missing in the whitelisted mint methods (`mintArcanaList`, `mintAspirantList`, `mintAllianceList`). This means that if the address that is whitelisted is a contract and it calls those functions but it can't handle ERC721 tokens correctly, they will be stuck. This problem is usually handled by using `_safeMint` instead of `_mint` but all `mint` functionality in `ArcanaPrime` uses `_mint`.

Impact

This can result in a user losing his newly minted tokens forever, which is a potential values loss. It requires the user to be using a smart contract that does not handle ERC721 properly, so it is Medium severity.

Recommendation

In `mintArcanaList`, `mintAspirantList` and `mintAllianceList` change the `_mint` call to `_safeMint`. Keep in mind this adds a reentrancy possibility, so it is best to add a `nonReentrant` modifier as well.

8.3. Low Findings

[L-01] Usage of `ecrecover` should be replaced with usage of OpenZeppelin's `ECDSA` library

Signature malleability is one of the potential issues with ecrecover. Even though it is not a threat to the current implementation using the highest security standards is always good. `ECDSA` is already imported, but not actually used. Replace the usage of `ecrecover` with the `ECDSA.recover` functionality.