Avalaunch XAVA-protocol Audit

January 2022

By CoinFabrik

Introduction

CoinFabrik was asked to audit the contracts for the XAVA Protocol project. First we will provide a summary of our discoveries and then we will show the details of our findings. We next follow with fix recommendations and the issue status after fixes were implemented.

Scope

The contracts audited are from the git repository with this URI: https://github.com/avalaunch-app/xava-protocol/. The audit is based on the commit `bef2af73a6fe4898e9c6ef099aa112c1d07e7d89`. A revision was made on commit `65b3779be43df907b1a14ce046dd990b68d19550`.

The audited contracts are:

- `AllocationStaking.sol`,
- `AvalaunchBadgeFactory.sol`,
- `sales/AvalaunchSales.sol`,
- `sales/SalesFactory.sol`,
- `airdrop/Airdrop.sol`,
- `airdrop/AirdropAVAX.sol`,
- `airdrop/AirdropSale.sol`.

The scope of the audit is limited to those files. No other files in this repository were audited. Its dependencies are assumed to work according to their documentation. Also, no tests were reviewed for this audit.

Analyses

Without being limited to them, the audit process included the following analyses:

- Arithmetic errors,
- Outdated version of Solidity compiler,
- Race conditions,
- Reentrancy attacks,
- Misuse of block timestamps,
- Denial of service attacks,
- Excessive gas usage,
- Missing or misused function qualifiers,
- Needlessly complex code and contract interactions,

- Poor or nonexistent error handling,
- Insufficient validation of the input parameters, and
- Incorrect handling of cryptographic signatures

Privileged Roles

These contacts use a standard Admin interface with an admin role, exposing the `isAdmin()` function. Both implement the `onlyAdmin()` modifier. Sales can only be created via the `SalesFactory` contract, and only 'admin' can call the non-view functions in this contract.

As for the `AvalaunchSale` contract, the admin can set and modify parameters in each of the sales. Every sale is assigned an owner, the party with this special role can deposit the tokens in the sale, and withdraw earnings and leftovers when the sale is over.

The `AllocationStaking` contract inherits from OpenZeppelin's `OwnableUpgradeable` and uses the `onlyOwner` modifier. It is only the owner who can set certain parameters (`saleFactory`, `depositFee`, allocationPoint and post-sale penalties), modify the admin address and add new liquidity pools. The admin role is the only valid signer: every withdrawal order must be signed by the admin.

Finally, the `AvalaunchBadgeFactory` contract inherits from `ERC1155PausableUpgradeable` and the `IAdmin` interface. Most tasks are reserved to the admin, who can pause, unpause, set URI, set the contract URI, create and mint badges.

Summary of Findings

Security Issues

ID	Title	Severity	Status
CR-01	Invalid signature in `withdrawTokens()`	Critical	Solved
ME-01	Admin With Zero Address Leading to Accept Invalid Signatures	Medium	Solved
ME-02	Token Loss Via Token Modification	Medium	Acknowledged

ME-03	Reentrancies in `AirdropSale.sol`	Medium	Acknowledged
MI-01	Front-running Attacks the Admin in AvalaunchSales	Minor	Acknowledged
MI-02	Double Spending in `depositTokens()`	Minor	Solved

Security Issues Found

Severity Classification

Security risks are classified as follows:

- **Critical:** These are issues that we manage to exploit. They compromise the system seriously. They must be fixed **immediately**.
- **Medium:** These are potentially exploitable issues. Even though we did not manage to exploit them or their impact is not clear, they might represent a security risk in the near future. We suggest fixing them **as soon as possible**.
- **Minor:** These issues represent problems that are relatively small or difficult to take advantage of but can be exploited in combination with other issues. These kinds of issues do not block deployments in production environments. They should be taken into account and be fixed **when possible**.

Issues Status

An issue detected by this audit can have four distinct statuses:

- **Unresolved**: The issue has not been resolved.
- **Acknowledged**: The issue remains in the code but is a result of an intentional decision.
- **Resolved**: Adjusted program implementation to eliminate the risk.
- **Mitigated**: Implemented actions to minimize the impact or likelihood of the risk

Critical Severity Issues

CR-01 Invalid Signature in `withdrawTokens()`

Location:

- `xava-protocol/contracts/AirdropSale.sol:63`

In the `AirdropSale.sol` contract, the function `withdrawTokens()` needs a signature to authorize any token withdrawal. This signature is composed of a hash of the beneficiary, the amount and the contract address, as we can see in `AirdropSale.sol:103`

```
bytes32 hash = keccak256(abi.encodePacked(beneficiary, amount, address(this)));
```

The problem is that the `withdrawTokens()` function receives an array of amounts (one amount for each token to withdraw) but the signature only supports a single amount.

So the code only checks the first amount in the amount array (`amount[0]`) at `AirdropSale.sol`:63:

```
require(checkSignature(signature, beneficiary, amounts[0]), "Not eligible to claim
tokens!");
```

This means that only the first amount is authorized, and the rest of the amounts can be freely modified by the caller, and the signature check will pass.

A malicious caller can withdraw any amount of tokens in this way.

Recommendation

The fix for this bug requires modifying the signature so every token amounts is hashed, and not only the first in the array

Status

Solved. The signature is now over the hashed amounts, which hence requires that none of the amounts is modified to be valid.

Medium Severity Issues

ME-01 Admin With Zero Address Leading to Accept Invalid Signatures

When calling the Admins constructor one may add the zero address (`address(0)`). This does not happen when calling `addAdmin()` which verifies the additions to be nonzero. A problem arises when calling `ECDSA.recover()` (e.g., within `AvalaunchSale.checkParticipationSignature()` or `AvalaunchSale.checkRegistrationSignature()`) returns `address(0)` if a signature is invalid. So that if the zero address is included in the admins array,

invalid signatures pass the checks. A malicious user could forge arbitrary transactions requiring signatures.

Recommendation
Check that addresses are nonzero in the `Admin` constructor.

Status
Solved. The suggestion was followed.

ME-02 Token Loss Via Token Modification

In AvalaunchSale the token might be set via `setSalesParams()` or `setSaleToken()` and then modified by the latter many times if the *gate* is not manually closed. For example, it could happen that the token is set with token A, a number of token As is deposited via depositToken() and then the token is changed to token B.

Recommendation
The logic should be modified so that the token may be set only once; or in any case that the token may not be modified after some tokens have been deposited.

Status
Acknowledged.

ME-03 Reentrancies in `AirdropSale.sol`
Location:

- `xava-protocol/contracts/AirdropSale.sol:75`
- `xava-protocol/contracts/AirdropSale.sol:89`

The function `withdrawTokens()` in the `AirdropSale.sol` contract is susceptible to reentrancy when calling `token.transfer()`. Notice that if the attacker controls the token contract, he can implement reentrancy attacks in the `transfer()` call at `AirdropSale.sol:89` function by calling `withdrawTokens()` recursively with different beneficiaries. The attacker can use the reentrancy to skip emitting the `SentERC20` event, and to avoid updating the `tokenToTotalWithdrawn` variable. A second point of reentrancy exists at `AirdropSale.sol:75` but with no security impact.

Recommendation
Use reentrancy guard or place reentrancy checks.

Status
Acknowledged.

Minor Severity Issues

MI-01 Front-running Attack the Admin in AvalaunchSales

The use of `block.timestamp` and the ability to modify parameters using the block timestamp may pose a vulnerability.

If the admin wanted to modify (reduce) `maxParticipation` a few seconds before the first round's start time, a malicious user could create a block adding these few seconds to the timestamp and participate using the old cap.

Recommendation

Only allow changes some time before the deadline, e.g., a minute before `startTime`. Alternatively, consider adding documentation to the contract preventing admins of the dangers of using sensitive timestamps.

Status

Acknowledged.

MI-02 Double Spending in depositTokens()

Location:

- `AvalaunchSale.sol:496-509`

The owner may call `depositTokens()` more than once. If this happened, funds would be inaccessible until the sale ends and he can call `withdrawEarningsAndLeftover()`.

Recommendations

Require that `saleTokensDeposited==false` before depositing.

Status

Solved. The issue was fixed according to the recommendations.

Enhancements

These items do not represent a security risk. They are best practices that we suggest implementing.

ID	Title	Status
EN-01	admin Array Not Checked For Repetitions During Construction	Not implemented
EN-02	Replace Integers by Constants	Not implemented
EN-03	Suboptimal Checks in setRounds()	Implemented
EN-04	Unnecessary Checks in createBadges()	Not implemented
EN-05	Old Solidity Compiler Version	Not implemented
EN-06	Pool Duplication in AllocationStaking	Not implemented
EN-07	Unused Mapping in AirdropSale	Not implemented

EN-01 `admin` Array Not Checked For Repetitions During Construction

Location:

- `Admin.sol:18-24.`

In the constructor of `Admin.sol` it could happen that an address is repeated in the input array `_addresses`. If this happens, the same address would be pushed more than once to the array `admins` and may create inconsistencies, e.g., when calling `removeAdmin()`. We recommend you check `isAdmin[_admins[i]] = false` before turning it to `true`.

Status

Not implemented.

EN-02 Replace Integers by Constants

In `FarmingXava.sol` replace occurrences of `1e18` and `1e36` by a call or a constant (e.g., `IERC20Metadata(address(token)).decimals()` as it is done in the `AvalaunchSale.sol` contract).

Status
Not implemented.

EN-03 Suboptimal Checks in setRounds()

Location:

- `sales/AvalaunchSale.sol:319-358,`

Each inequality introduced in a requirement at `AvalaunchSale.setRounds()` has a non-negligible gas cost. For example, since each iteration of the loop includes the code:

```
require(startTimes[i] > lastTimestamp);
lastTimestamp = startTimes[i];
```

only `require(startTimes[0] >= block.timestamp);` should be executed out of the loop. The same happens with `require(startTimes[i] > registration.registrationTimeEnds);` and `require(startTimes[i] < sale.saleEnd);`

Recommendations
Some of the requirements need only happen in the first list item (e.g., before the for-loop) and other checks for the last item (e.g., after the loop). Consider reordering the code accordingly.

Status
Implemented.

EN-04 Unnecessary Checks in createBadges()

In the function `createBadges()` an array of `badgeIds` is used as input, but these Ids are checked to be consecutive, so if this is a requirement, then only the first one is needed.

Status
Not implemented.

EN-05 Old Solidity Compiler Version

The audited contracts use an old version of solidity (v0.6.12). Consider updating the code to compile with the latest version.

Status
Not implemented.

EN-06 Pool Duplication in AllocationStaking

A user may add more than once a pool (with the same pool ID), as explained in the comments this would create problems with the contact's logic. This may be prevented, e.g., by requiring in add() that a pool with this pid doesn't exist, say

```
require(poolInfo[pid].lastRewardTimestamp != 0, "Pool already exists");
```

Status

Not implemented.

EN-07 Unused Mapping in AirdropSale

The `tokenToTotalWithdrawn` mapping is updated in every call to `withdrawTokens()` but is currently unused anywhere else. Consider removing.

Status

Not implemented.

Other Considerations

Two safeMath Libraries

Some contracts import OpenZeppelin's safeMath library and others use a custom safeMath library (`math/safeMath.sol`) which is a copy of OpenZeppelin's safeMath with some modifications. Also, note that some contracts, like AirdropSale.sol, are importing safeMath but not using it. In that case, consider removing and saving gas.

Typos

We list some types:

- There is a typo in the AllocationStaking.sol contract, line 224:

```
// Set user.tokensUnlockTime only if new timestamp is grater
```

  Change 'grater' to 'greater'.

Code Readability

- Reorder functions dividing view / nonview functions to help code readability.
- Also, we see instances of uint and uint256 interchangeably, and although uint is an alias for uint256 we suggest using one throughout the code.

Changelog

- 2021-12-29 – Initial report based on commit `bef2af73a6fe4898e9c6ef099aa112c1d07e7d89`.
- 2022-03-15 - Revision commit `65b3779be43df907b1a14ce046dd990b68d19550`.

Disclaimer: This audit report is not a security warranty, investment advice, or an approval of the Avalaunch project since CoinFabrik has not reviewed its platform. Moreover, it does not provide a smart contract code faultlessness guarantee.