SHERLOCK

Security Review For Atleta Network

Collaborative Audit Prepared For: **Atleta Network**
Lead Security Expert(s): **hildingr**
vinica_boy
Date Audited: **October 9 - October 11, 2025**

Introduction

VestingAtla is a Merkle-based ATLA vesting contract with role-gated administration and month-granular schedules starting from a fixed, immutable start time.

It includes the following features and design choices:

- Merkle-proof allocations: Beneficiaries and their total claimable amounts are verified using Merkle proofs, enabling gas-efficient, off-chain list management without storing recipient lists on-chain.
- Cliff and periodic unlocks: Supports a cliff (in months), an initial release at TGE, and subsequent periodic unlocks expressed as precise percentages over a defined number and length of periods.
- Calendar-aware time math: Month calculations are aligned to real-world calendars, safely adding months and computing month differences to avoid timestamp drift.
- Deterministic start: All schedules are anchored to a predetermined timestamp, ensuring consistent behavior across networks and time.
- Per-beneficiary accounting: Tracks amounts already paid and TGE status to prevent double claims while allowing progressive vesting over time.
- Single and batch claiming: Claims can be made individually or in batches, with validation and the ability to query both claimable amounts and remaining time to the next unlock.
- Strict initialization: Each vesting configuration can be initialized only once, and total configured distribution is validated to not exceed the allowed maximum.
- Role-based access control: Administrative operations are restricted to designated roles, separating governance/ownership from day-to-day vesting management.
- ATLA treasury with safeguards: The contract can hold ATLA, supports controlled withdrawals, and enforces sufficient balance and valid proofs for all claims.

In short, VestingAtla provides a robust, calendar-month-based ATLA vesting system with Merkle-verified allocations, precise percentage unlocks, and secure role-gated administration.

Scope

Repository: potemkinViktor/AtletaVesting

Audited Commit: 7c03e98cfa9fa21dbce94396dba7ff41e9ec8153

Final Commit: b99b79615a510ece5d8c7d1d7eb85d1452e2360a

Files:

- contracts/libs/BokkyPooBahsDateTimeLibrary.sol

- contracts/VestingAtla.sol

Final Commit Hash

b99b79615a510ece5d8c7d1d7eb85d1452e2360a

Findings

Each issue has an assigned severity:

- High issues are directly exploitable security vulnerabilities that need to be fixed.
- Medium issues are security vulnerabilities that may not be directly exploitable or may require certain conditions in order to be exploited. All major issues should be addressed.
- Low/Info issues are non-exploitable, informational findings that do not pose a security risk or impact the system's integrity. These issues are typically cosmetic or related to compliance requirements, and are not considered a priority for remediation.

Issues Found

High	Medium	Low/Info
0	0	5

Issues Not Fixed and Not Acknowledged

High	Medium	Low/Info
0	0	0

Issue L-1: Incorrect calculation of passed periods can lead to a month earlier vesting distribution [RESOLVED]

Source: https://github.com/sherlock-audit/2025-10-atleta-vesting-contracts-oct-9th/issues/3

Summary

To calculate the number of periods elapsed since the start of vesting, we use a `diffMonths` function that accounts only for calendar month differences, not a full ~30-day month difference. Depending on date on which the vesting has started, the distribution can be distributed earlier than expected.

Vulnerability Detail

In `checkClaimableAmount()` and `_calculateClaimableAmount()` we calculate the passed periods the following way:

```
if (block.timestamp >= tsAfterCliff) {
    passedPeriods = TimeLib.diffMonths(tsAfterCliff, block.timestamp) /
        data.durationPeriodInMonths +
        1; // +1 because the first period starts from the moment of the start of
        ↪  the vesting
}
```

`diffMonths` would give us calendar months difference:

```
function diffMonths(uint fromTimestamp, uint toTimestamp) internal pure returns
↪  (uint _months) {
    require(fromTimestamp <= toTimestamp, "BokkyPooBahsDateTimeLibrary: from ts
    ↪  should be greater then to ts");
    (uint fromYear, uint fromMonth,) = _daysToDate(fromTimestamp / SECONDS_PER_DAY);
    (uint toYear, uint toMonth,) = _daysToDate(toTimestamp / SECONDS_PER_DAY);
    _months = toYear * 12 + toMonth - fromYear * 12 - fromMonth;
}
```

Meaning that if starting timestamp is 31st of October and current timestamp is 1st of November, we would account for one passed between the two timestamps.

Impact

The impact is only limited to a ~ month earlier if the timestamp from which we start to count periods is near the end of the month. It can be problematic for shorter vestings and if it affects the number of funds expected to be available in the contract.

Code Snippet

https://github.com/sherlock-audit/2025-10-atleta-vesting-contracts-oct-9th/blob/f351c1f82f3b491ff2d6e2c852d408f0b0dfe0ae/AtletaVesting/contracts/VestingAtla.sol#L217C7-L221C10

https://github.com/sherlock-audit/2025-10-atleta-vesting-contracts-oct-9th/blob/f351c1f82f3b491ff2d6e2c852d408f0b0dfe0ae/AtletaVesting/contracts/libs/BokkyPooBahsDateTimeLibrary.sol#L279C3-L284C6

Tool Used

Manual Review

Recommendation

If this is a problem for the vesting, consider accounting for days difference between the start and current timestamp and add a period only if days passed is > ~30.

Issue L-2: Incorrect calculation of remainingToNextPeriod in checkClaimableAmount() [RESOLVED]

Source: https://github.com/sherlock-audit/2025-10-atleta-vesting-contracts-oct-9th/issues/4

Summary

The `checkClaimableAmount()` function returns incorrect `remainingToNextPeriod`

Vulnerability Detail

When `checkClaimableAmount()` is called and `block.timestamp >= tsAfterCliff` we calculate the `remainingToNextPeriod`

```
if (passedPeriods < data.totalPeriods) {
        remainingToNextPeriod = TimeLib.addMonths(block.timestamp, 1) -
        ↪  block.timestamp;
    }
```

This does not calculate the time until the next period starts, instead it calculates the time for 1 month in the future.

Impact

The `checkClaimableAmount()` returns incorrect data. Function is currently not used for any on-chain action in the contract. Any serious consequence would be due to off-chain use of the incorrect information or due to an on-chain integration that relies on the function.

Code Snippet

https://github.com/sherlock-audit/2025-10-atleta-vesting-contracts-oct-9th/blob/f351c1f82f3b491ff2d6e2c852d408f0b0dfe0ae/AtletaVesting/contracts/VestingAtla.sol#L143

Tool Used

Manual Review

Recommendation

Calculate the time to the next period correctly:

```
TimeLib.addMonths(tsAfterCliff, passedPeriods * data.durationPeriodInMonths) -
↪  block.timestamp;
```

Issue L-3: Missing input validation in constructor [RESOLVED]

Source: https://github.com/sherlock-audit/2025-10-atleta-vesting-contracts-oct-9th/issues/5

Summary

Missing validation checks in constructor

Vulnerability Detail

Consider adding checks here:

https://github.com/sherlock-audit/2025-10-atleta-vesting-contracts-oct-9th/blob/f351c1f82f3b491ff2d6e2c852d408f0b0dfe0ae/AtletaVesting/contracts/VestingAtla.sol#L69-L70

`VESTING_START_TIME` should not be in the past and `BASE_PERCENT` should not be less than 10000

Impact

The checks will protect agains creating a vesting contract with incorrect parameters

Code Snippet

https://github.com/sherlock-audit/2025-10-atleta-vesting-contracts-oct-9th/blob/f351c1f82f3b491ff2d6e2c852d408f0b0dfe0ae/AtletaVesting/contracts/VestingAtla.sol#L69-L70

Tool Used

Manual Review

Recommendation

Add validation checks

Issue L-4: Missing validation in setVestingData() [RESOLVED]

Source: https://github.com/sherlock-audit/2025-10-atleta-vesting-contracts-oct-9th/issues/6

Summary

To protect against overwrites `setVestingData()` should revert after `VESTING_START_TIME`.

Vulnerability Detail

`setVestingData()` can now be called with the same ID and overwrite the previous `VestingData`.

https://github.com/sherlock-audit/2025-10-atleta-vesting-contracts-oct-9th/blob/f351c1f82f3b491ff2d6e2c852d408f0b0dfe0ae/AtletaVesting/contracts/VestingAtla.sol#L248-L257

By blocking after `VESTING_START_TIME` we allow for changes until the vesting process starts but not after.

Impact

Potential overwrite `VestingData` which can result in incorrect claims

Code Snippet

https://github.com/sherlock-audit/2025-10-atleta-vesting-contracts-oct-9th/blob/f351c1f82f3b491ff2d6e2c852d408f0b0dfe0ae/AtletaVesting/contracts/VestingAtla.sol#L244-L260

Tool Used

Manual Review

Recommendation

Add

```
require(block.timestamp < VESTING_START_TIME, ...);
```

Issue L-5: data.totalAmount is never deducted from. [RESOLVED]

Source: https://github.com/sherlock-audit/2025-10-atleta-vesting-contracts-oct-9th/issues/7

Summary

The `data.totalAmount` parameter is currently not deducted from at any point. It would be reasonable to use this parameter to limit the total amount possible to claim for each ID even if the a merkle proof is faulty

Vulnerability Detail

`data.totalAmount` is currently only used to check if it has been set to >0 here: https://github.com/sherlock-audit/2025-10-atleta-vesting-contracts-oct-9th/blob/f351c1f82f3b491ff2d6e2c852d408f0b0dfe0ae/AtletaVesting/contracts/VestingAtla.sol#L212

We are missing deduction of this parameter when each user claims if this parameter was intended to be used as another layer of security against a total drain of the vesting pool in case of a single faulty merkle proof.

Impact

Missing safety check

Code Snippet

https://github.com/sherlock-audit/2025-10-atleta-vesting-contracts-oct-9th/blob/f351c1f82f3b491ff2d6e2c852d408f0b0dfe0ae/AtletaVesting/contracts/VestingAtla.sol#L212

Tool Used

Manual Review

Recommendation

Deduct from `data.totalAmount` a when user claims.

Disclaimers

Sherlock does not provide guarantees nor warranties relating to the security of the project.

Usage of all smart contract software is at the respective users' sole risk and is the users' responsibility.