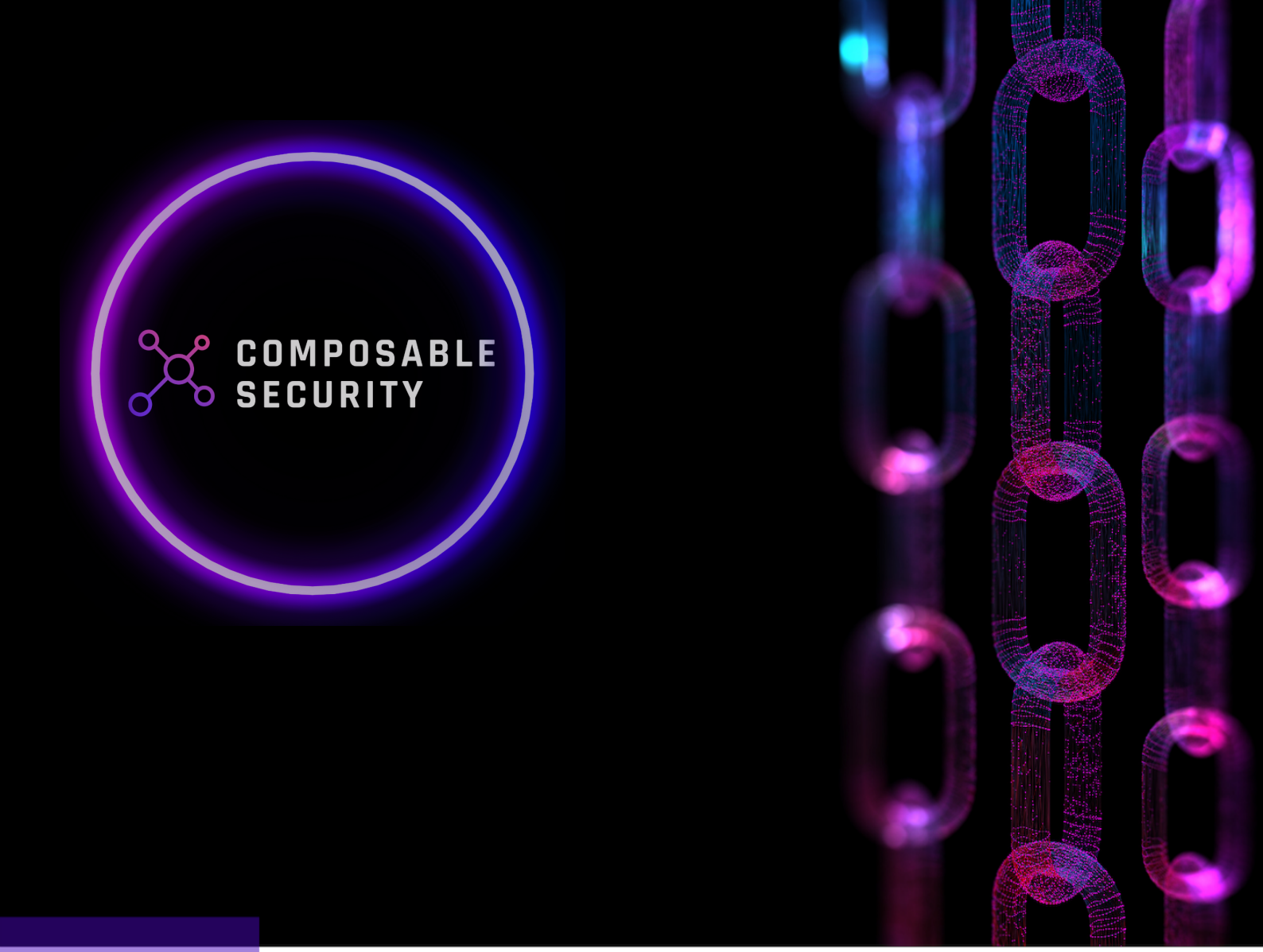


COMPOSABLE SECURITY

# REPORT

Smart contract security review for Arcade Services Inc.

Prepared by: Composable Security
Report ID: ARC-b5d2776
Test time period: 2024-07-22 - 2024-07-26
Retest time period: 2024-08-20 - 2024-10-08
Report date: 2024-10-08
Version: 1.2
Visit: composable-security.com

# Contents

**1. Retest summary (2024-10-08) 3**
1.1 Results . . . 3
1.2 Scope . . . 4

**2. Current findings status 5**

**3. Security review summary (2024-07-26) 7**
3.1 Client project . . . 7
3.2 Results . . . 7
3.3 Scope . . . 8

**4. Project details 9**
4.1 Project's goal . . . 9
4.2 Agreed scope of tests . . . 9
4.3 Threat analysis . . . 9
4.4 Testing methodology . . . 10
4.5 Disclaimer . . . 11

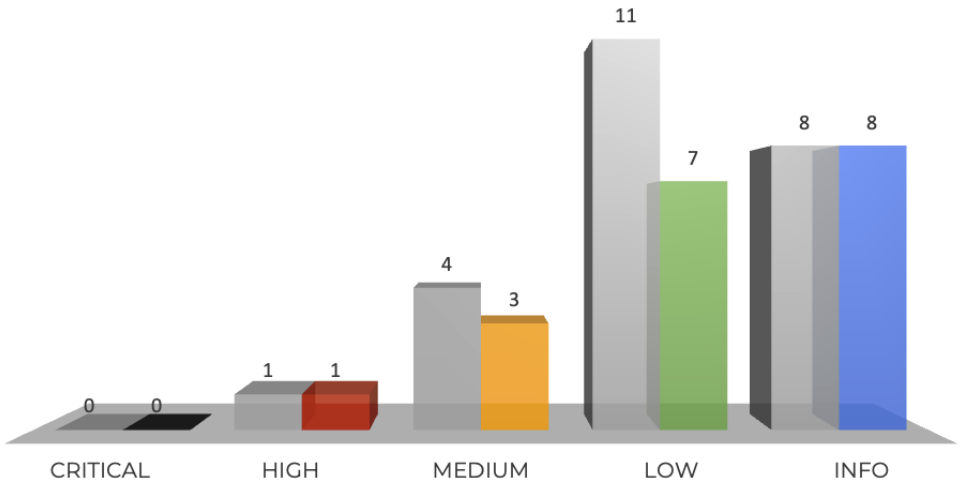
**5. Vulnerabilities 12**
[ARC-b5d2776-H01] Lack of required mission statuses . . . 12
[ARC-b5d2776-M01] All lenders are treated the same way . . . 14
[ARC-b5d2776-M02] The early withdrawal fee non-compliant with the documentation 15
[ARC-b5d2776-M03] Invalid mission data emitted . . . 16
[ARC-b5d2776-M04] Excessively powerful roles . . . 17
[ARC-b5d2776-L01] Lack of protocol limits for administrative updateMissionPoolDistributions function . . . 18
[ARC-b5d2776-L02] Full control over rewards and no incentive to reward mission by MPO . . . 20
[ARC-b5d2776-L03] Incorrect handling of non-standard ERC20 tokens . . . 21
[ARC-b5d2776-L04] The maxMpcInvestment limit can be easily bypassed through multiple addresses . . . 22
[ARC-b5d2776-L05] Inconsistent signature verification for admin . . . 23
[ARC-b5d2776-L06] Event not emitted . . . 24
[ARC-b5d2776-L07] The closeMission may close the mission with an incorrect result 25
[ARC-b5d2776-L08] Issues with rounding down to zero . . . 26
[ARC-b5d2776-L09] Funds can be early withdrawn at any moment . . . 28
[ARC-b5d2776-L10] Invalid analytics results . . . 29
[ARC-b5d2776-L11] Lack of safeTransfer . . . 30

**6. Recommendations** **32**
[ARC-b5d2776-R01] Use correct data types . . . . . 32
[ARC-b5d2776-R02] Optimize TVL calculation . . . . . 32
[ARC-b5d2776-R03] Remove unnecessary code . . . . . 33
[ARC-b5d2776-R04] Remove deprecated libraries . . . . . 34
[ARC-b5d2776-R05] Use constants instead of magic numbers . . . . . 35
[ARC-b5d2776-R06] Use encodeCall to keep the code type safe . . . . . 35
[ARC-b5d2776-R07] Import specific contracts from the file . . . . . 36
[ARC-b5d2776-R08] Update comment . . . . . 37

**7. Impact on risk classification** **39**

**8. Long-term best practices** **40**
8.1 Use automated tools to scan your code regularly . . . . . 40
8.2 Perform threat modeling . . . . . 40
8.3 Use Smart Contract Security Verification Standard . . . . . 40
8.4 Discuss audit reports and learn from them . . . . . 40
8.5 Monitor your and similar contracts . . . . . 40

# 1. Retest summary (2024-10-08)




The description of the current status for each retested vulnerability and recommendation has been added in its section.

## 1.1. Results

The **Composable Security** team was involved in a one-time iteration of verification whether the vulnerabilities detected during the tests (between 2024-07-22 and 2024-07-26) were removed correctly and no longer appear in the code.

The current status of detected issues is as follows:

- 1 **high** vulnerability has not been removed from the code.
- 4 vulnerabilities with a **medium** impact on risk were handled as follows:
  - 1 has been fixed,
  - 3 have been acknowledged.
- 11 vulnerabilities with a **low** impact on risk were handled as follows:
  - 6 have been acknowledged,
  - 1 has been partially fixed,
  - 4 have been fixed.
- 8 security **recommendations** were handled as follows:
  - 8 have been acknowledged.
- One vulnerability (ARC-b5d2776-L08) was removed in separate commits:
  - `edb78a876eb5db2e1556440d0054e49e3687dca6`,
  - `fde66201c2c1763e4ceccf418c747e6540a6a3b5`.

The team has resolved a subset of the identified vulnerabilities, with plans to address additional issues in upcoming releases. Some risks are being mitigated by relying on excessively

powerful roles, though there remains room for improvement and further enhancement of security.

## 1.2. Scope

The retest scope included the same contracts, on a different commit in the same repository.

**GitHub repository:** https://github.com/ArcadeDevelopment/arcade-evm-mp-contracts
**CommitID:** ee3e98e451b716a08e40ed917b50cd9725d98ebb

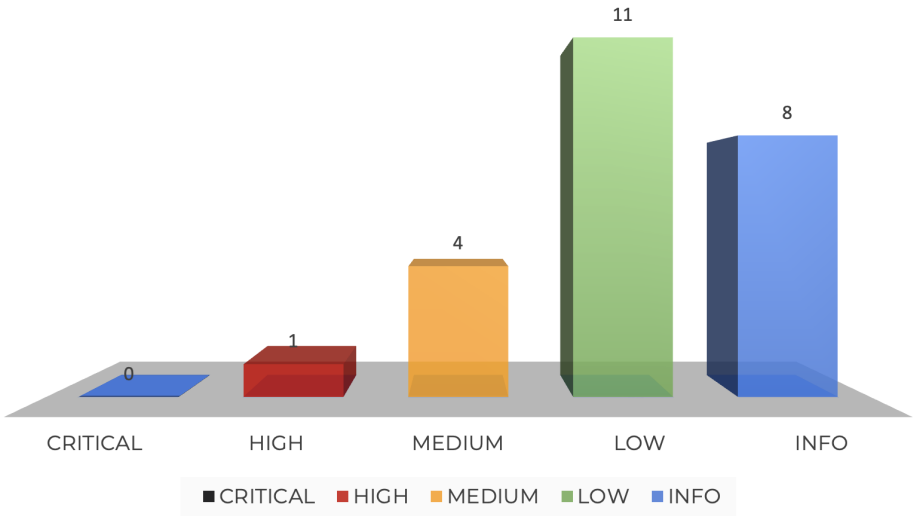
# 2. Current findings status

| ID | Severity | Vulnerability | Status |
|---|---|---|---|
| ARC-b5d2776-H01 | HIGH | Lack of required mission statuses | ACKNOWLEDGED |
| ARC-b5d2776-M01 | MEDIUM | All lenders are treated the same way | ACKNOWLEDGED |
| ARC-b5d2776-M02 | MEDIUM | The early withdrawal fee non-compliant with the documentation | ACKNOWLEDGED |
| ARC-b5d2776-M03 | MEDIUM | Invalid mission data emitted | FIXED |
| ARC-b5d2776-M04 | MEDIUM | Excessively powerful roles | ACKNOWLEDGED |
| ARC-b5d2776-L01 | LOW | Lack of protocol limits for administrative updateMissionPoolDistributions function | PARTIALLY FIXED |
| ARC-b5d2776-L02 | LOW | Full control over rewards and no incentive to reward mission by MPO | ACKNOWLEDGED |
| ARC-b5d2776-L03 | LOW | Incorrect handling of non-standard ERC20 tokens | ACKNOWLEDGED |
| ARC-b5d2776-L04 | LOW | The maxMpcInvestment limit can be easily bypassed through multiple addresses | ACKNOWLEDGED |
| ARC-b5d2776-L05 | LOW | Inconsistent signature verification for admin | FIXED |
| ARC-b5d2776-L06 | LOW | Event not emitted | FIXED |
| ARC-b5d2776-L07 | LOW | The closeMission may close the mission with an incorrect result | ACKNOWLEDGED |
| ARC-b5d2776-L08 | LOW | Issues with rounding down to zero | FIXED |
| ARC-b5d2776-L09 | LOW | Funds can be early withdrawn at any moment | ACKNOWLEDGED |
| ARC-b5d2776-L10 | LOW | Invalid analytics results | FIXED |
| ARC-b5d2776-L11 | LOW | Lack of safeTransfer | ACKNOWLEDGED |

| ID | Severity | Recommendation | Status |
|---|---|---|---|
| ARC-b5d2776-R01 | INFO | Use correct data types | NOT IMPLEMENTED |
| ARC-b5d2776-R02 | INFO | Optimize TVL calculation | NOT IMPLEMENTED |
| ARC-b5d2776-R03 | INFO | Remove unnecessary code | NOT IMPLEMENTED |
| ARC-b5d2776-R04 | INFO | Remove deprecated libraries | NOT IMPLEMENTED |

| | | | |
|---|---|---|---|
| ARC-b5d2776-R05 | INFO | Use constants instead of magic numbers | NOT IMPLEMENTED |
| ARC-b5d2776-R06 | INFO | Use encodeCall to keep the code type safe | NOT IMPLEMENTED |
| ARC-b5d2776-R07 | INFO | Import specific contracts from the file | NOT IMPLEMENTED |
| ARC-b5d2776-R08 | INFO | Update comment | NOT IMPLEMENTED |

# 3. Security review summary (2024-07-26)




## 3.1. Client project

The **Arcade Mission Pools** project is a GameFi platform, built on Ethereum that allows MPO and MPC to interact with Mission Pools, which have a wide range of applications. Each mission can be an event, quest or task. MPOs create a mission which they will then try to complete successfully, while MPCs can participate in it as its investors and thus not only observe their favorite player, but also participate in prizes.

## 3.2. Results

The **Arcade Services Inc.** engaged Composable Security to review security of **Arcade Mission Pools**. Composable Security conducted this assessment over 1 person-week with 2 engineers.

The summary of findings is as follows:

- 1 vulnerability with a **high** impact on risk was identified. Its potential consequence is:
  - Stealing rewards as MPC.
- 4 vulnerabilities with a **medium** impact on risk were identified.
- 11 vulnerabilities with a **low** impact on risk were identified.
- 8 **recommendations** have been proposed that can improve overall security and help implement best practice.
- The most important issues detected concern business logic and enforcement of white paper assumptions. Some assumptions are not or are just partially implemented and allow for abuse or work in a different way than intended.

- The documentation is not consistent. There are discrepancies between the comments and the implemented code and white paper. Comments require thorough review and updating.
- Some vulnerabilities from the previous report have still not been removed and have therefore been included in this report.
- The protocol relies heavily on the trust to the Admin and selected MPOs, who can harm users in many ways when, for example, their private key is compromised. Therefore, it is very important to introduce additional restrictions and, if possible, mitigate the associated risks and provide appropriate security measures for trusted operator roles.

Composable Security recommends that **Arcade Services Inc.** complete the following:

- Address all reported issues.
- Update and unify documentation, including comments and White Paper.
- According to the documentation, mission pool funding can be done using XP via web2 components, they should also be audited.
- Extend unit tests with scenarios that cover detected vulnerabilities where possible.
- Consider whether the detected vulnerabilities may exist in other places (or ongoing projects) that have not been detected during engagement.

## 3.3. Scope

The scope of the tests included selected contracts from the following repository.

**GitHub repository:** https://github.com/ArcadeDevelopment/arcade-evm-mp-contracts
**CommitID:** b5d2776ea87a8ca8ae70119aabb787d3c629cddc

The detailed scope of tests can be found in Agreed scope of tests.

# 4. Project details

## 4.1. Project's goal

The Composable Security team focused during this audit on the following:

- Perform a tailored threat analysis.
- Ensure that smart contract code is written according to security best practices.
- Identify security issues and potential threats both for **Arcade Services Inc.** and their users.
- The secondary goal is to improve code clarity and optimize code where possible.

## 4.2. Agreed scope of tests

The subjects of the test were selected contracts from the **Arcade Services Inc.** repository.

**GitHub repository:**
https://github.com/ArcadeDevelopment/arcade-evm-mp-contracts
**CommitID:** b5d2776ea87a8ca8ae70119aabb787d3c629cddc

Files in scope:

```
.
└── src
    ├── ArcadeMissionPools.sol
    ├── ArcadeMissionPoolsAnalytics.sol
    ├── IArcadeMissionPools.sol
    └── lib
        ├── DistributionsUtility.sol
        └── Signatory.sol
```

**Documentation:** Delivered in the form of White Paper: `https://www.arcade2earn.io/wp-content/uploads/2024/03/Arcade-Whitepaper.pdf`.

## 4.3. Threat analysis

This section summarizes the potential threats that were identified during initial threat modeling performed before the audit. The tests were focused, but not limited to, finding security issues that could be exploited to achieve these threats.

Key assets that require protection:

- Users' funds.

Potential attackers goals:

- Theft of user's tokens.
- Theft of rewards from Mission Pool.
- Lock users' funds in the Mission Pool.
- Funding after the specified deadline or before start.
- Bypassing imposed investment limits.
- Bypassing the early withdrawal penalty.
- Block the contract, so that others cannot use it.
- Lack of the white paper assumptions enforcement.

Potential scenarios to achieve the indicated attacker's goals:

- Pre-funding non-existing mission.
- Funding without transfer.
- Invalid calculations of fees.
- Assuming the transfer value based on declared amount instead of the balances difference.
- Unfair reward distribution.
- Influence or bypass the business logic of the system.
- Take advantage of arithmetic errors.
- Privilege escalation through incorrect access control to functions or badly written modifiers.
- Existence of known vulnerabilities (e.g., front-running, re-entrancy).
- Design issues.
- Excessive power, too much in relation to the declared one.
- Private key compromise, rug-pull.
- Withdrawal of more funds than expected.
- Participating in multiple mission pools with the same tokens.
- Using multiple accounts or a smart contract to bypass limits.

## 4.4. Testing methodology

Smart contract security review was performed using the following methods:

- Q&A sessions with the **Arcade Services Inc.** development team to thoroughly understand intentions and assumptions of the project.
- Initial threat modeling to identify key areas and focus on covering the most relevant scenarios based on real threats.
- Automatic tests using slither.
- Custom scripts (e.g. unit tests) to verify scenarios from initial threat modeling.
- **Manual review of the code.**

## 4.5. Disclaimer

Smart contract security review **IS NOT A SECURITY WARRANTY**.

During the tests, the Composable Security team makes every effort to detect any occurring problems and help to address them. However, it is not allowed to treat the report as a security certificate and assume that the project does not contain any vulnerabilities. Securing smart contract platforms is a multi-stage process, starting from threat modeling, through development based on best practices, security reviews and formal verification, ending with constant monitoring and incident response.

***Therefore, we encourage the implementation of security mechanisms at all stages of development and maintenance.***

# 5. Vulnerabilities

## [ARC-b5d2776-H01] Lack of required mission statuses

HIGH ACKNOWLEDGED

**Retest (2024-08-20)**

The vulnerability has been acknowledged.
Team's response: *We are only using minimum required statuses on the smart contract and other validations of statuses are verified in our API and prevent usage without the signatory verification.*

### Affected files

- IArcadeMissionPools.sol#L9

### Description

Currently, only 3 mission statuses are possible

```
9  enum MissionStatus {
10   FundingStage,
11   MissionEndedFailed,
12   MissionEndedSuccess
13 }
```

There are no statuses specifying the moment when funding ended, a mission is in progress or a default status when funding has not started. Some of these statuses may be considered optional, but due to the lack of an intermediate status between funding stage and mission ended stage, the current implementation introduces a high risk.

Currently, the reward proportion is only calculated based on the amount invested by MPC and nothing else.

```
34 function calculateMpcDistribution(
35   uint256 _totalFunded,
36   uint256 _mpcFundedAmount,
37   uint256 _missionRewardAmount,
38   uint256 _mpcDistributionBasisPoints
39 ) internal pure returns (uint256) {
40   // Calculate the MPC percentage shares to 2 decimal places pre basis point
       division
```

```
41      uint256 mpcPercentShares = (_mpcFundedAmount * BASIS_POINTS) / _totalFunded;
42
43      // The total reward amount based on the MPC percentage shares
44      uint256 mpcPredistributionRewardShare = (_missionRewardAmount *
45        mpcPercentShares) / BASIS_POINTS;
46
47      // Calculate the MPC distribution based on mission pool distribution basis
            points
48      uint256 mpcDistributionAmount = (mpcPredistributionRewardShare *
49        _mpcDistributionBasisPoints) / BASIS_POINTS;
50
51      return mpcDistributionAmount;
52    }
```

Therefore, the MPC can wait until the last moment of the game to be sure of the outcome of whether the mission will be successful and, just before `missionClose` is called, deposit a huge sum to get a larger share of the rewards than the other MPCs who supported the MPO from the beginning.

## Attack scenario

The attackers might take the following steps in turn:

① MPO creates a new mission pool.
② Some MPCs start funding the mission.
③ The attacker waits until it is known whether the mission will be successful and then they fund it with maximum amount to collect the rewards.

**Result of the attack:** Stealing part of the rewards.

**Recommendation**

Consider more statuses, especially `FundingNotStarted`, `FundingEnded` or `MissionStarted` status, after which users will no longer be able to invest.
Once `poolTarget` is reached during mission funding, the status should automatically change to `FundingEnded` and disallow further funding. Consider also increasing the penalty for early withdrawals after the funding has ended (status `FundingEnded`).

## References

1. SCSVS G4: Business logic

# [ARC-b5d2776-M01] All lenders are treated the same way

MEDIUM ACKNOWLEDGED

**Retest (2024-08-20)**

The vulnerability has been acknowledged, the team plans to remove it in future releases.
Team's response: *Low priority, not required until Lender full implementation, and currently are not anticipating bundling of lenders assets for any one type of mission.*

## Affected files

- ArcadeMissionPools.sol#L1000

## Description

The reward for lenders is equal for each lender, even if they lent a different amount, token of a different value or for a longer period of time. Moreover, if one lender lends twice, it will not be possible to add them again to the list or increase their share. There is no way to reward them based on their contribution.

## Vulnerable scenario

The following steps lead to the described result:

① The MPO has created a mission.
② Lender A has lent 1000 tokens or 1 NFT for the mission.
③ Lender B has lent 3000 tokens or 3 NFT for the mission.
④ After mission is closed, both lenders A and B receive the same portion of the reward.

**Result:** Unfair distribution of rewards for asset lenders.

**Recommendation**

Allow the MPO to differentiate the contribution of each asset lender and distribute the rewards fairly.

## References

1. SCSVS G4: Business Logic

# [ARC-b5d2776-M02] The early withdrawal fee non-compliant with the documentation

MEDIUM ACKNOWLEDGED

**Retest (2024-08-20)**

The vulnerability has been acknowledged, the team plans to fix it in future releases. Team's response: *For future state implementation, currently keeping a flat rate early withdrawal fee.*

## Affected files

- ArcadeMissionPools.sol#L734

## Description

According to WP, the early withdrawal mechanism should range from 0% to 15% depending on several variables, including but not limited to:

- The overall duration of the pool,
- The amount of time remaining at the time of the withdrawal,
- The withdrawing MPC's share size of the total $xARC staked in that Mission Pool,
- The total amount of $xARC staked in that Mission Pool.

However, overall duration of the pool and the amount of time remaining at the time of the withdrawal is not considered. The `calculateEarlyWithdrawalPenalty` function simply calculates the share based on the amount deposited by MPC.

```
16  function calculateEarlyWithdrawalPenalty(
17    uint256 _mpcFundedAmount,
18    uint256 _missionFeeBasisPoints
19  ) internal pure returns (uint256) {
20    // Calculate the early withdrawal penalty based on mission fee basis points
21    uint256 earlyWithdrawalPenalty = (_mpcFundedAmount *
22      _missionFeeBasisPoints) / BASIS_POINTS;
23
24    return earlyWithdrawalPenalty;
25  }
```

This means that the MPC who decides to invest at the end has as much to win as the user who has supported MPO since the mission was created, and that the potential penalty for both MPCs will be the same regardless of when they withdraw.

**Result:** Lack of compliance with the business assumptions.

**Recommendation**

Update the `calculateEarlyWithdrawalPenalty` function to reflect your business assumptions or update the documentation

### References

1. SCSVS G4: Business logic

## [ARC-b5d2776-M03] Invalid mission data emitted

MEDIUM FIXED

**Retest (2024-08-20)**

The vulnerability has been removed. The `creatingMissionId` variable still exists, but in the current implementation it does not pose a threat, and the event emits correct data due to the fix in L404.

### Affected files

- ArcadeMissionPools.sol#L395

### Description

The contract emits the invalid mission ID on creation. The incremented ID is emitted which is the ID of a mission that will be created in the next call.

### Vulnerable scenario

The following steps lead to the described result:

① No missions were created yet.
② The MPO creates the first mission that should gets the ID equal to 0 in the `missions` variable.
③ The mission is saved under the ID qual to 0.
④ The mission ID returned by the function is correct and equal to 0.
⑤ The event `MissionCreated` contains the incremented value of mission ID equal to 1.

**Result:** Invalid mission ID emitted in the event.

**Recommendation**

Use `creatingMissionId` to store mission in mapping and as parameter in the `MissionCreated` event. Additionally, consider removing the `currentMissionId` storage variable and get the value from `missionIdCounter` variable to return the number of missions and get the next free ID for a mission.

### References

1. SCSVS G4: Business Logic

## [ARC-b5d2776-M04] Excessively powerful roles

MEDIUM ACKNOWLEDGED

**Retest (2024-08-20)**

The vulnerability has been acknowledged.

### Affected files

- ArcadeMissionPools.sol#L551

### Description

The current implementation of the system is highly centralized. Roles such as `DEFAULT_ADMIN_ROLE` have significant privileges that allow them to call functions that have a strong impact on operation and changes are made immediately. In some cases, this allows for complete takeover of the user's tokens. Although the team plans to use multi-sig, which significantly reduces the threat, it is not an ideal solution that fully protects the user.

Another role with significant powers is the MPO role. They may decide not to complete the mission and not distribute the reward as they receive the entire prize and are not obligated to share it. They may also not add the lender or remove them just before closing the mission.

**Note:** The severity of the vulnerability has been downgraded to MEDIUM. This is predicated on the understanding that the protocol is intentionally centralized. However, it is crucial to acknowledge that incidents of rug pulls have eroded trust in centralized DeFi projects. Consequently, excessive power within these functionalities may act as a deterrent to prospective users.

**Result of the attack:** Immediate changes of critical parameters that in worst case could lead to theft of contributors' tokens or Denial of Service.

**Recommendation**

**Short-term:** Continue the planned use of multisig. Use timelocks to give users time to react to upcoming changes. Add limits on functions that support fees. Communicate the threats coming from centralization in the documentation.
**Long-term:** Propose a more decentralized approach or delegate ownership to DAO.

Functions that are powerful and heavily impact business logic:

As Admin

- `withdrawLeftoverFunds`
- `removeMissionPoolOperator`
- `approveMissionPoolToken`
- `revokeMissionPoolToken`
- `updateStandardFee`
- `setSignatory`
- `setPlatformFeeVault`
- `updateMissionPool`
- `updateMissionPoolDistributions`

As MPO

- `closeMission`
- `addAssetLender`
- `removeAssetLender`

### References

1. SCSVS G1: Architecture, design and threat modeling

## [ARC-b5d2776-L01] Lack of protocol limits for administrative updateMissionPoolDistributions function

LOW PARTIALLY FIXED

**Retest (2024-08-20)**

The vulnerability has been partially removed. The `MAX_PLATFORM_FEE` is verified in L1262-L1264, but it has been set to a rather high value of 80%. The `totalDistributionPoints` is verified in L1254-L1260.

The update of the distributions for a mission pool is still made without using a

timelock.

## Affected files

- ArcadeMissionPools.sol#L1252

## Description

The `updateMissionPoolDistributions` function allows the admin to make any changes without any limits. In case of compromise of private key, this may lead to theft of funds by, for example, setting a 100% fee for the platform.

```
1252  function updateMissionPoolDistributions(
1253    uint256 _missionId,
1254    uint256 _platformFeePoints,
1255    uint256 _mpoFeePoints,
1256    uint256 _mpcRewardPoints,
1257    uint256 _lenderRewardPoints
1258  )
1259    public
1260    isMissionFunding(_missionId)
1261    onlyRole(DEFAULT_ADMIN_ROLE)
1262    nonReentrant
1263  {
1264    Mission storage mission = missions[_missionId];
1265    if (mission.distributions.platformFeePoints != _platformFeePoints) {
1266      mission.distributions.platformFeePoints = _platformFeePoints;
1267    }
1268    if (mission.distributions.mpoFeePoints != _mpoFeePoints) {
1269      mission.distributions.mpoFeePoints = _mpoFeePoints;
1270    }
1271    if (mission.distributions.mpcRewardPoints != _mpcRewardPoints) {
1272      mission.distributions.mpcRewardPoints = _mpcRewardPoints;
1273    }
1274    if (mission.distributions.lenderRewardPoints != _lenderRewardPoints) {
1275      mission.distributions.lenderRewardPoints = _lenderRewardPoints;
1276    }
1277
1278    emit MissionDistributionsUpdated(
1279      _missionId,
1280      _platformFeePoints,
1281      _mpoFeePoints,
1282      _mpcRewardPoints,
```

```
1283        _lenderRewardPoints
1284      );
1285    }
```

**Result:** Theft of user funds.

**Recommendation**

- Enforce protocol limits (especially the fee limits).
- Updating the distribution should not be treated as an emergency function. Updates should be performed in a controlled manner, preferably using timelock.

### References

1. SCSVS G4: Business logic

## [ARC-b5d2776-L02] Full control over rewards and no incentive to reward mission by MPO

LOW ACKNOWLEDGED

**Retest (2024-08-20)**

The vulnerability has been acknowledged. The MPO is considered trusted and this ability is considered a feature.

### Affected files

- ArcadeMissionPools.sol#L551
- ArcadeMissionPools.sol#L965

### Description

The MPO controls the value of the reward when closing the mission. Additionally, there is not incentive or penalty for the MPO to close a mission. This can lead to unwanted behavior, especially when the reward is large. Such implementation might be a disincentive for contributors to fund the mission.

### Attack scenario

The malicious MPO might take the following steps in turn:

① Creates a mission pool.
② MPCs fund the mission by locking their tokens.

③ The mission is completed and generates a reward.
④ The MPO does not call the `closeMission` or call it with no reward.
⑤ MPCs does not earn rewards

**Result of the attack:** Lack of rewards for contributors or lenders.

**Recommendation**

- Consider requiring to specify the final or at least minimal reward value when the mission is created by the operator.
- Consider requiring to stake some tokens by the MPO to create a mission that is unstaked on closing the mission.

### References

1. SCSVS G4: Business Logic

## [ARC-b5d2776-L03] Incorrect handling of non-standard ERC20 tokens

LOW ACKNOWLEDGED

**Retest (2024-08-20)**

The vulnerability has been acknowledged. The team does not plan to support non-standard ERC20 tokens. The risk of freezing such tokens as USDT is accepted.
Team's response: *We are not using ERC20s that are non-standard for Mission Pools. Tokens to use for Mission Pools will be vetted to ensure they are not fee-on-transfer tokens, rebase tokens, freezable tokens or tokens that revert on zero transfers.*

### Affected files

- ArcadeMissionPools.sol#L17

### Description

Using non-standard ERC20 tokens, such as **fee-on-transfer tokens**, **rebase tokens**, **freezable tokens** or **tokens that revert on zero transfers**, can lead to inability to withdraw the funds for contributors or inability to close the mission and distribute rewards.

**Note:** The team has created a list of tokens that are first accepted and only then can be used, but the vulnerability is reported to maintain awareness of existing threats to non-standard tokens.

### Vulnerable scenario

The example for the `closeMission` function and a token that reverts on zero transfer:

① The admin accepts a token that reverts on zero amount transfers.
② The MPO creates a new mission.
③ The MPCs fund the mission.
④ The MPO tries to close the mission with a reward.
⑤ The contract calculates the platform reward and MPO reward and one of them is zero.
⑥ The contract reverts the call and mission cannot be closed.

Similar case exists for `earlyWithdrawalFromMission` function, when `earlyWithdrawalFee` can be set to 0.

**Result:** Inability to execute functions that might make zero transfers (e.g. `closeMission`, `earlyWithdrawalFromMission`) and to withdraw the funds for contributors.

**Recommendation**

If you plan to support non-standard ERC20 tokens, consider one of the following:

- Check the balance before and after the transfer. Another solution might be to validate whether the actual transferred amount is the same as the passed amount parameter.
- To support tokens that reverts on zero transfer, check if the amount is 0 and do not call the transfer function in such case.
- To support rebase tokens, make sure the cached balances are atomically updated as part of the rebase procedure. If you do not plan to support non-standard ERC20 tokens, create a policy to verify used tokens.

Additionally, you will have to accept the risk for freezable tokens (e.g. USDT).

### References

1. SCSVS I2: Token
2. SCSVS G4: Business Logic

## [ARC-b5d2776-L04] The maxMpcInvestment limit can be easily bypassed through multiple addresses

LOW ACKNOWLEDGED

**Retest (2024-08-20)**

The vulnerability has been acknowledged.
Team's response: *We have KYC implementation on our platform and this is not a priority to restrict at this time.*

### Affected files

- ArcadeMissionPools.sol#L461

### Description

The `maxMpcInvestment` was created to counteract whales investments and accommodate a larger number of different MPCs. However, checking this limit per address does not mitigate this threat. A large investor can easily distribute their funds to many different addresses and thus increase their limit.

To clearly define the limit per person, KYC procedures should be considered.

### Attack scenario

The attackers might take the following steps in turn:

① There is a `maxMpcInvestment` of 10,000 tokens per MPC.
② Whale investor divides their tokens between 5 different addresses.
③ They invests 10,000 tokens from each address separately, investing a total of 50,000 tokens. That is, 5x more than they should be able to.

**Result of the attack:** Bypassing `maxMpcInvestment` limit.

**Recommendation**

Consider introducing KYC and on-chain verification to effectively impose a limit.

### References

1. G1: Architecture, design and threat modeling

## [ARC-b5d2776-L05] Inconsistent signature verification for admin

LOW FIXED

**Retest (2024-08-20)**

The vulnerability was removed as recommended. Signature verification was removed from `endMissionTopOffFunding` function.

### Affected files

- ArcadeMissionPools.sol#L494-L512

### Description

The `endMissionTopOffFunding` function is protected with `onlyRole` modifier and can be called only by admin. However, it also requires a `_signature` parameter that is later verified. Other admin functions do not require signature and can be called by admin without any additional access control checks.

### Vulnerable scenario

The following steps lead to the described result:

① The admin tries to call any other administrative functions.
② The functions are executed correctly.
③ The admin tries to call `endMissionTopOffFunding` function without signature of the `SIGNATORY`.
④ The function call reverts.

**Result:** Inability to execute `endMissionTopOffFunding` function by the admin without signatory approval.

**Recommendation**

Either remove the signature verification for the `endMissionTopOffFunding` function or require signatures on other administrative functions (except `setSignatory`).

### References

1. SCSVS G5: Access control

## [ARC-b5d2776-L06] Event not emitted

LOW FIXED

**Retest (2024-08-20)**

The vulnerability has been removed. The event is emitted in L989.

### Affected files

- ArcadeMissionPools.sol#L965

### Description

The `addAssetLender` function is supposed to emit the `AssetLenderAdded` event according to the documentation. The event is defined, but it is never emitted.

**Result:** The `AssetLenderAdded` is never emitted.

**Recommendation**

Emit the `AssetLenderAdded` event in the `addAssetLender` function.

### References

1. SCSVS G1: Architecture, design and threat modeling

## [ARC-b5d2776-L07] The closeMission may close the mission with an incorrect result

LOW ACKNOWLEDGED

**Retest (2024-08-20)**

The vulnerability has been acknowledged.
Team's response: *This is intended for the `closeMission`. We do not want to allow mission to be closed multiple times. Our API and signatory validation prevents this issue of closing a mission without the proper reward if it is successful.*

### Affected files

- ArcadeMissionPools.sol#L639

### Description

Calling a mission before its completion when the reward is still zero will change the status to `MissionEndedFailed`.

```
638 } else {
```

```
639     missions[_missionId].status = MissionStatus.MissionEndedFailed;
640 }
```

This will prevent the function from being called again and closing properly due to the `isMissionFunding` modifier. Funding and early withdrawals are dynamic and can now be made at any time, and this may affect the result during the `closeMission`.

### Vulnerable scenario

The following steps lead to the described result:

① Premature call of the `closeMission` function, before rewards have been calculated or funding has not ended.
② Result due to 0 rewards will be set to 0 and considered failed ( `bool result = _reward > 0;` )
③ If result is considered as failed, `MissionStatus.MissionEndedFailed` will be set.
④ Calling `closeMission` again will fail when rewards have already been calculated correctly due to the `isMissionFunding` modifier.

**Result:** Inability to close the mission properly.

**Recommendation**

Allow the mission to be closed only when its status is appropriate, when funding has ended.

### References

1. SCSVS G4: Business logic

## [ARC-b5d2776-L08] Issues with rounding down to zero

LOW FIXED

**Retest (2024-10-08)**

The vulnerability has been removed as recommended.
A new private constant variable `HIGH_PRECISION` has been introduced in a separate commit (`edb78a876eb5db2e1556440d0054e49e3687dca6`). The following functions have been modified to maintain high precision of calculations:

- `DistributionsUtlity`,
  - `calculateEarlyWithdrawalPenalty`,
  - `calculateMpcDistribution`,

    - `calculateMpoDistribution`,
    - `calculateLenderDistribution`,
    - `calculatePlatformDistribution`.

What is more, in a separate commit (`fde66201c2c1763e4ceccf418c747e6540a6a3b5`), the following functions return the rates with decimals, allowing to calculate the float numbers on the front-end side of the application:

- `ArcadeMissonPoolAnalytics`,
    - `getIndividualRetentionRate`,
    - `getAverageRetentionRate`,

## Affected files

- ArcadeMissionPoolsAnalytics.sol#L117
- ArcadeMissionPoolsAnalytics.sol#L141
- DistributionsUtility.sol#L41

## Description

The protocol performs calculations that include division which can lead to loosing the precision of the result.

Here are the identified cases:

- The `getIndividualRetentionRate` is used to get the retention rate of a specific mission pool contributor (MPC) across specific missions. The base amount used to protect from rounding down in Solidity is 100 (the number of missions is multiplied by 100). However, the edge case with big number of mission IDs in the parameter and very small number of missions that MPC has invested in can lead to invalid (lower than actual) retention rate.
- In the `getAverageRetentionRate` function, in case of big number of unique addresses and small number of investments the returned value can be incorrectly rounded down to 0.
- In the `calculateMpcDistribution` function the basis point used for calculations is 10_000. Such value can lead to rounding down to zero a reward of a contributor who has funded the mission with a small value compared to the poo amount.

## Vulnerable scenario

The following steps lead to the described result for the first mentioned case:

① MPC invests in one mission pool.
② The application calculates the retention rate asking for 101 mission pools.
③ The retention rate returned is 0 (due to rounding down of 1*100/101).

The following steps lead to the described result for the third mentioned case:

① MPO creates a mission to collect 1_000_000 tokens.
② MPC invests 99 tokens.
③ MPO closes mission with 1000 tokens reward.
④ MPC gets no reward due to rounding down to zero of his share (`99 * 10_000 / 1_000_000`).

**Result:** Depending on the case:

- Retention rate lower than actual.
- No reward for the low-value investor.

**Recommendation**

Use greater base value. The exact value should be selected based on the expected values of numerators and denominators (e.g. number of mission IDs used as parameter). For the token's amount calculation it is a good practice to use 18 decimals for calculations to keep the precision.

### References

1. SCSVS G7: Arithmetic

## [ARC-b5d2776-L09] Funds can be early withdrawn at any moment

LOW ACKNOWLEDGED

**Retest (2024-08-20)**

The vulnerability has been acknowledged.
Team's response: *This is intended functionality and our API and signatory validation prevents early withdrawal when the mission is not in funding period.*

### Affected files

- ArcadeMissionPools.sol#L734

### Description

The comments state that `earlyWithdrawalFromMission` should only be possible after funding has been completed or just before the mission is completed.

```
720     * @notice Function for MPC to withdraw funds early from Mission
721     * Pool after funding completes and before mission end date
```

However, there is no validation of this statement and MPC can withdraw its funds both during the funding period and after mission close.

**Note:** No scenario has been detected in which such an action would bring unfair benefits to MPC, but it is an action inconsistent with business assumptions.

**Result:** MPC can withdraw funds using the `earlyWithdrawalFromMission` function at any time, including after closing a mission.

**Recommendation**

Add validation checking whether a given mission has a status in which this operation should be performed.

### References

1. SCSVS G4: Business logic

## [ARC-b5d2776-L10] Invalid analytics results

LOW FIXED

**Retest (2024-08-20)**

The vulnerability has been fixed as recommended. The `getMissionDebriefStats` function now use 2 new functions in L69-72. The `_filterExistingMissions` to verify whether `missionId` exists (L304-L332), and `_filterFinishedMissions` to return only missions that ended successfully or failed (L337-L370).
The `getAverageDistributionRate` rely on `_filterExistingMissions` to count only missions that ended successfully or failed (L304-L332).

### Affected files

- ArcadeMissionPoolsAnalytics.sol#L80
- ArcadeMissionPoolsAnalytics.sol#L147

### Description

The `getMissionDebriefStats` function accepts a list of mission IDs and calculates how many of them were successful and failed. Then sums up their rewards. However, the function does

not check whether the missions are finished and therefore will include missions that were not yet created or are still in progress as failed missions.

The similar case is in the `getAverageDistributionRate` where missions are not checked for existence (e.g. mission ID lower than `currentMissionId`). The non existing mission IDs are used to divide the sums of calculated distribution parameters.

### Vulnerable scenario

The following steps lead to the described result:

① The user or external application calls the `getMissionDebriefStats` function with a list of 5 successfully finished missions and other 6 not yet created missions.
② The function returns 5 successful missions and **6 failed mission**.

**Result:** Invalid results returned by the functions:

- Bigger than actual number of failed missions.
- Lower than actual distribution parameters

**Recommendation**

Add checks whether the mission exists and is finished (in case of `getMissionDebriefStats` function).

### References

1. SCSVS G4: Business Logic

## [ARC-b5d2776-L11] Lack of safeTransfer

LOW ACKNOWLEDGED

**Retest (2024-08-20)**

The vulnerability has been acknowledged. The team does not plan to support tokens that do not revert.
Team's response: *This is understood and we have no short term intentions on adding other ERC20 tokens for mission pools.*

### Affected files

- ArcadeMissionPools.sol#L734

## Description

Although the main token contract to be used is `xArcadeToken` or `ArcadeToken`, the contracts can accept other tokens from approved list that could not revert on invalid transfers, but return false. The mission pool and vault contracts would not detect that and assumed that the transfer was correct.

**Result:** Invalid or lack of token transfer.

**Recommendation**

Use the `safeTransfer` function (from SafeERC20 library) to transfer tokens from the contract that calls this function.

The vulnerability occurs in the following places:

- ArcadeMissionPools.sol#L772
- ArcadeMissionPools.sol#L775
- ArcadeMissionPools.sol#L875
- ArcadeMissionPools.sol#L1338
- ArcadeMissionPools.sol#L1385

## References

1. SafeERC20
2. SCSVS I2: Token

# 6. Recommendations

## [ARC-b5d2776-R01] Use correct data types

INFO NOT IMPLEMENTED

**Retest (2024-08-20)**

The recommendation has not been implemented, the team will consider this in the future release.

### Description

The fields `mpoWithdrawn` and `adminWithdrawn` of `Mission` struct are of type `EnumerableSet.AddressSet` while they can contain only one address. Therefore, the `bool` type is enough.

**Recommendation**

Use `bool` type for those fields.

- IArcadeMissionPools.sol#L62
- IArcadeMissionPools.sol#L63

### References

1. SCSVS G1: Architecture, design and threat modeling

## [ARC-b5d2776-R02] Optimize TVL calculation

INFO NOT IMPLEMENTED

**Retest (2024-08-20)**

The recommendation has not been implemented, the team will consider this in the future release.

### Description

In the current implementation of the `getTVL_MPC` function the following steps are taken in a loop over the mission IDs:

1. check whether the MPC has invested in given mission (iteration over mission investors),
2. check whether the mission token is the specified token,

3. get the investment value (retrieving investment from the investments field).

```
36  function getTVL_MPC(
37    address mpc,
38    uint256[] calldata missionIds,
39    address token
40  ) external view returns (uint256 tvl) {
41    uint256 missionsIdsLength = missionIds.length;
42
43    for (uint256 i = 0; i < missionsIdsLength; ++i) {
44      uint256 missionId = missionIds[i];
45
46      if (_isMissionContributor(missionId, mpc)) {
47        IArcadeMissionPools.MissionWithOmitedInvestors
48          memory mission = arcadeMissionPools.getMission(missionId);
49
50        if (address(mission.token) == token) {
51          tvl += arcadeMissionPools.getMissionInvestment(missionId, mpc);
52        }
53      }
54    }
55  }
```

Such construction is gas inefficient and can be simplified by removing the first check because in such case the third step would return 0.

**Recommendation**

Use only `getMissionInvestment` to get the investment of particular MPC and sum it up (it is 0 if the MPC has not invested).

- ArcadeMissionPoolsAnalytics.sol#L36-L55

### References

1. SCSVS G1: Architecture, design and threat modeling
2. SCSVS G10: Gas usage & limitations

## [ARC-b5d2776-R03] Remove unnecessary code

INFO NOT IMPLEMENTED

**Retest (2024-08-20)**

The recommendation has not been implemented, the team will consider this in the future release.

### Description

The `fundMission` function updates the value of funder's investment that is stored in `mission.investments` variable. It however updates also the `mpcInvestment` memory variable which is not necessary in this case.

```
468     mission.investors.add(msg.sender);
469     mission.investments.set(msg.sender, mpcInvestment += _value);
470     mission.poolAmount += _value;
471     mission.token.safeTransferFrom(msg.sender, address(this), _value);
```

**Recommendation**

Change to `mpcInvestment + _value`.

- ArcadeMissionPools.sol#L469

### References

1. SCSVS G4: Business Logic

## [ARC-b5d2776-R04] Remove deprecated libraries

**INFO** **NOT IMPLEMENTED**

**Retest (2024-08-20)**

The recommendation has not been implemented, the team will consider this in the future release.

### Description

The `DistributionsUtility` library and `ArcadeMissionPools` contract import `SafeMath` library and use it for `uint256` even though the project uses 0.8 version of Solidity.

Here are the places where it is imported and assigned to `uint256`:

- ArcadeMissionPools.sol#L7
- ArcadeMissionPools.sol#L25
- DistributionsUtility.sol#L4
- DistributionsUtility.sol#L7

**Recommendation**

Remove the `SafeMath` library.

### References

1. SCSVS G1: Architecture, design and threat modeling

## [ARC-b5d2776-R05] Use constants instead of magic numbers

INFO NOT IMPLEMENTED

**Retest (2024-08-20)**

The recommendation has not been implemented, the team will consider this in the future release.

### Description

It is a best practice to define and use constants or enums instead of using the numbers explicitly.

Here is the list of places when hard-coded numbers are used:

- ArcadeMissionPools.sol#L180
- ArcadeMissionPools.sol#L359
- ArcadeMissionPoolsAnalytics.sol#L117
- ArcadeMissionPools.sol#L108

**Recommendation**

Use constants and enum instead of numbers explicitly.

### References

1. SCSVS G11: Code clarity

## [ARC-b5d2776-R06] Use encodeCall to keep the code type safe

INFO NOT IMPLEMENTED

**Retest (2024-08-20)**

The recommendation has not been implemented, the team will consider this in the future release.

### Description

When using `abi.encodeWithSignature`, it is possible to include a typo for the correct function signature. When using `abi.encodeWithSignature` or `abi.encodeWithSelector`, it is also possible to provide parameters that are not of the correct type for the function.

Here is the list of occurences:

- ArcadeMissionPools.sol#L317
- ArcadeMissionPools.sol#L435
- ArcadeMissionPools.sol#L500
- ArcadeMissionPools.sol#L559
- ArcadeMissionPools.sol#L679
- ArcadeMissionPools.sol#L742
- ArcadeMissionPools.sol#L817
- ArcadeMissionPools.sol#L981
- ArcadeMissionPools.sol#L1027

**Note:** The finding is reported as recommendation because no typo has been found.

**Recommendation**

Use `abi.encodeCall` instead or `abi.encodeWithSignature` or `abi.encodeWithSelector`.

### References

1. SCSVS G1: Architecture, design and threat modeling

## [ARC-b5d2776-R07] Import specific contracts from the file

INFO NOT IMPLEMENTED

**Retest (2024-08-20)**

The recommendation has not been implemented, the team will consider this in the future release.

### Description

Import declarations should import specific identifiers, rather than the whole file. Using import declarations of the form `import <identifier_name> from "some/file.sol"` avoids polluting the symbol namespace making flattened files smaller, and speeds up compilation (but does not save any gas).

**Recommendation**

Import specific identifiers from files.

### References

1. SCSVS G11: Code clarity

## [ARC-b5d2776-R08] Update comment

INFO NOT IMPLEMENTED

**Retest (2024-08-20)**

The recommendation has not been implemented, the team will consider this in the future release.

### Description

There are places in the code where comments are incorrect or missing.

Instances with missing NatSpec or invalid comments:

- ArcadeMissionPools.sol#L534-L550
- ArcadeMissionPools.sol#L880-L891
- ArcadeMissionPools.sol#L476-L486
- ArcadeMissionPools.sol#L1130-L1138
- ArcadeMissionPools.sol#L644-L655
- ArcadeMissionPools.sol#L797
- ArcadeMissionPools.sol#L801

**Recommendation**

Review the indicated instances and correct the comments so that they correspond to the operation of the function and describe all the parameters it accepts as an input. All comments should be reviewed, as the team didn't focus on the comments any further after several inconsistencies were discovered.

## References

1. SCSVS G11: Code clarity

# 7. Impact on risk classification

Risk classification is based on the one developed by OWASP [1], however it has been adapted to the immutable and transparent code nature of smart contracts. The Web3 ecosystem forgives much less mistakes than in the case of traditional applications, the servers of which can be covered by many layers of security.

Therefore, the classification is more strict and indicates higher priorities for paying attention to security.

| OVERALL RISK SEVERITY | | | | |
|---|---|---|---|---|
| Impact on risk | HIGH | CRITICAL | HIGH | MEDIUM |
| | MEDIUM | MEDIUM | MEDIUM | LOW |
| | LOW | LOW | LOW | INFO |
| | | HIGH | MEDIUM | LOW |
| | | Likelihood | | |

[1] OWASP Risk Rating methodology

# 8. Long-term best practices

## 8.1. Use automated tools to scan your code regularly

It's a good idea to incorporate automated tools (e.g. slither) into the code writing process. This will allow basic security issues to be detected and addressed at a very early stage.

## 8.2. Perform threat modeling

Before implementing or introducing changes to smart contracts, perform threat modeling and think with your team about what can go wrong. Set potential targets of the attacker and possible ways to achieve them, keep it in mind during implementation to prevent bad design decisions.

## 8.3. Use Smart Contract Security Verification Standard

Use proven standards to maintain a high level of security for your contracts. Treat individual categories as checklists to verify the security of individual components. Expand your unit tests with selected checks from the list to be sure when introducing changes that they did not affect the security of the project.

## 8.4. Discuss audit reports and learn from them

The best guarantee of security is the constant development of team knowledge. To use the audit as effectively as possible, make sure that everyone in the team understands the mistakes made. Consider whether the detected vulnerabilities may exist in other places, audits always have a limited time and the developers know the code best.

## 8.5. Monitor your and similar contracts

Use the tools available on the market to monitor key contracts (e.g. the ones where user's tokens are kept). If you have used code from another project, monitor their contracts as well and introduce procedures to capture information about detected vulnerabilities in their code.


## Damian Rusinek

Smart Contracts Auditor

@drdr_zz

damian.rusinek@composable-security.com



## Paweł Kuryłowicz

Smart Contracts Auditor

@wh01s7

pawel.kurylowicz@composable-security.com