

LeagueTech Audit Report

Dec 22, 2023



Table of Contents

| | |
|--|-----------|
| Summary | 2 |
| Overview | 3 |
| Issues | 4 |
| [WP-C1] Wrong implementation of <code>initialPrice</code> allows an attacker to drain the protocol. | 4 |
| [WP-H2] Wrong implementation of <code>buySubscription()</code> allows the user to buy subscription without paying the full price | 7 |
| [WP-M3] Subscriptions subjects should be able to set their own <code>subscriptionPrice</code> . | 10 |
| [WP-L4] Unreachable code | 12 |
| [WP-L5] Missing sanity check for setter functions. | 13 |
| [WP-G6] Using constants instead of storages can save some gas | 14 |
| Appendix | 17 |
| Disclaimer | 18 |

Summary

This report has been prepared for LeagueTech smart contract, to discover issues and vulnerabilities in the source code of their Smart Contract as well as any contract dependencies that were not part of an officially recognized library. A comprehensive examination has been performed, utilizing Static Analysis and Manual Review techniques.

The auditing process pays special attention to the following considerations:

- Testing the smart contracts against both common and uncommon attack vectors.
- Assessing the codebase to ensure compliance with current best practices and industry standards.
- Ensuring contract logic meets the specifications and intentions of the client.
- Cross referencing contract structure and implementation against similar smart contracts produced by industry leaders.
- Thorough line-by-line manual review of the entire codebase by industry experts.

Overview

Project Summary

| | |
|--------------|---|
| Project Name | LeagueTech |
| Codebase | https://github.com/leaguetechn/contracts |
| Commit | b716dcc33ba044fe4e0ee187d1e28ebfea60fb17 |
| Language | Solidity |

Audit Summary

| | |
|-------------------|--------------------------------|
| Delivery Date | Dec 22, 2023 |
| Audit Methodology | Static Analysis, Manual Review |
| Total Issues | 6 |

[WP-C1] Wrong implementation of `initialPrice` allows an attacker to drain the protocol.

Critical

Issue Description

1. `getPrice()` returns the total price of the transaction for a given `amount` of shares.
2. Therefore, `initialPrice` applies to the entire transaction, not the starting price per share.

An attacker can purchase multiple shares at the total price of just `initialPrice`. As long as they can buy more than 1 share with `initialPrice`, they can sell each share individually and receive `initialPrice` as a return.

<https://github.com/leaguetechnology/contracts/blob/59d3e75b6c0dd728a8fd83e7aa343b90da9580b9/contracts/LeagueTechV1.sol#L181-L215>

```

181  /// @notice Sell an `amount` of `sharesSubject` shares' for the caller
182  /// @dev Once the first share is created, the last share cannot be sold
183  /// @param sharesSubject Address of the user to sell shares of
184  /// @param amount The amount of shares to sell
185  function sellShares(address sharesSubject, uint256 amount) public payable
    nonReentrant whenNotPaused {
186      require(amount > 0, "NO_ZERO_SHARE_SELL");
187      uint256 supply = sharesSupply[sharesSubject];
188      require(supply > amount, "LAST_SHARE_CANNOT_BE_SOLD");
189      uint256 price = getPrice(supply - amount, amount);
190      tvl[sharesSubject] -= price;
191
192      uint256 protocolFee = (price * protocolFeePercent) / 1 ether;
193      uint256 subjectFee = (price * subjectFeePercent) / 1 ether;
194      uint256 referralFee = (price * referralFeePercent) / 1 ether;
195      require(sharesBalance[sharesSubject][msg.sender] >= amount,
        "NOT_ENOUGH_SHARES");
196      sharesBalance[sharesSubject][msg.sender] =
        sharesBalance[sharesSubject][msg.sender] - amount;
197      sharesSupply[sharesSubject] = supply - amount;
198
199      sendToSubject(msg.sender, price - protocolFee - subjectFee - referralFee);

```

```

200     sendToProtocol(protocolFee);
201     sendToSubject(sharesSubject, subjectFee);
202     sendToReferrer(msg.sender, referralFee);
203
204     emit Trade(
205         msg.sender,
206         sharesSubject,
207         false,
208         amount,
209         price,
210         protocolFee,
211         subjectFee,
212         referralFee,
213         supply - amount
214     );
215 }

```

<https://github.com/leaguetechn/contracts/blob/59d3e75b6c0dd728a8fd83e7aa343b90da9580b9/contracts/LeagueTechV1.sol#L274-L292>

```

274  /// @notice Gets the total price of an `amount` of shares with a specified
275      `supply`
276  /// @param supply The current supply of shares
277  /// @param amount The amount of shares to buy or sell
278  function getPrice(uint256 supply, uint256 amount) public view returns (uint256) {
279      uint256 adjustedSupply = supply + 2;
280      if (adjustedSupply == 0) {
281          return initialPrice;
282      }
283      uint256 sum1 = ((adjustedSupply - 1) * (adjustedSupply) * (2 * (adjustedSupply
284      - 1) + 1)) / 6;
285      uint256 sum2 = ((adjustedSupply - 1 + amount) *
286      (adjustedSupply + amount) *
287      (2 * (adjustedSupply - 1 + amount) + 1)) / 6;
288      uint256 summation = (80 ether / 100) * (sum2 - sum1);
289      uint256 price = ((50 ether / 100) * summation * initialPrice) / 1 ether / 1
290      ether;
291      if (price < initialPrice) {
292          return initialPrice;
293      }
294      return price;

```

```
292 }
```

Recommendation

```
274 /// @notice Gets the total price of an `amount` of shares with a specified
275     `supply`
276 /// @param supply The current supply of shares
277 /// @param amount The amount of shares to buy or sell
277 function getPrice(uint256 supply, uint256 amount) public view returns (uint256) {
278     if (supply == 0) {
279         return initialPrice * amount;
280     }
281     uint256 adjustedSupply = supply + 2;
282     uint256 sum1 = ((adjustedSupply - 1) * (adjustedSupply) * (2 * (adjustedSupply
- 1) + 1)) / 6;
283     uint256 sum2 = ((adjustedSupply - 1 + amount) *
284         (adjustedSupply + amount) *
285         (2 * (adjustedSupply - 1 + amount) + 1)) / 6;
286     uint256 summation = (80 ether / 100) * (sum2 - sum1);
287     uint256 price = ((50 ether / 100) * summation * initialPrice) / 1 ether / 1
ether;
288     uint256 pricePerShare = price / amount;
289     if (pricePerShare < initialPrice) {
290         return initialPrice * amount;
291     }
292     return price;
293 }
```

Status

✓ Fixed

[WP-H2] Wrong implementation of `buySubscription()` allows the user to buy subscription without paying the full price

High

Issue Description

`subscriptionPrice` should go to `subscriptionsSubject` instead of `msg.sender` .

<https://github.com/leaguetechnology/contracts/blob/59d3e75b6c0dd728a8fd83e7aa343b90da9580b9/contracts/LeagueTechV1.sol#L243-L264>

```

243 function buySubscription(address subscriptionsSubject) public payable nonReentrant
    whenNotPaused {
244     require(subscriptionsSubject != msg.sender, "NO_SELF_SUBSCRIPTION");
245     require(subscriptionsEnabled[subscriptionsSubject] == 1,
        "SUBJECT_SUBSCRIPTIONS_NOT_ENABLED");
246     require(subscribers[subscriptionsSubject][msg.sender] < block.timestamp,
        "ALREADY_SUBSCRIBED");
247
248     uint256 price = subscriptionPrice;
249     uint256 protocolFee = (price * protocolFeePercent) / 1 ether;
250     uint256 subjectFee = (price * subjectFeePercent) / 1 ether;
251     uint256 referralFee = (price * referralFeePercent) / 1 ether;
252     require(msg.value >= price + protocolFee + subjectFee + referralFee,
        "NOT_ENOUGH_SENT");
253     uint256 endTime = block.timestamp + subscriptionDuration;
254     subscribers[subscriptionsSubject][msg.sender] = endTime;
255     setReferralIneligible(msg.sender);
256
257     sendToSubject(msg.sender, price - protocolFee - subjectFee - referralFee);
258     sendToProtocol(protocolFee);
259     sendToSubject(subscriptionsSubject, subjectFee);
260     sendToReferrer(msg.sender, referralFee);
261     sendToSubject(msg.sender, msg.value - (price + protocolFee + subjectFee +
        referralFee));
262
263     emit Subscription(msg.sender, subscriptionsSubject, endTime, protocolFee,
        subjectFee, referralFee);
264 }

```


Breakdown of current funds flow:

```

LeagueTechV1 Contract:
  + (msg.value)
  - (subscriptionPrice - protocolFee - subjectFee - referralFee)
  - (protocolFee)
  - (subjectFee)
  - (referralFee)
  - (msg.value - (subscriptionPrice + protocolFee + subjectFee + referralFee))
  == (protocolFee + subjectFee + referralFee)
Subject:
  + (subjectFee)
ProtocolFeeDestinaton:
  + (protocolFee)
Referrer:
  + (referralFee)
msg.sender:
  - (msg.value)
  + (subscriptionPrice - protocolFee - subjectFee - referralFee)
  + (msg.value - (subscriptionPrice + protocolFee + subjectFee + referralFee))
  == -2 * (protocolFee + subjectFee + referralFee)

```

In summary, the current implementation will send a large part of the funds back to the msg.sender and leave a part of the funds in the contract unexpectedly.

Recommendation

1. Send the `price` to `subject` .
2. Remove `subjectFee` .

```

function buySubscription(address subscriptionsSubject) public payable nonReentrant
whenNotPaused {
    require(subscriptionsSubject != msg.sender, "NO_SELF_SUBSCRIPTION");
    require(subscriptionsEnabled[subscriptionsSubject] == 1,
"SUBJECT_SUBSCRIPTIONS_NOT_ENABLED");
    require(subscribers[subscriptionsSubject][msg.sender] < block.timestamp,
"ALREADY_SUBSCRIBED");

    uint256 price = subscriptionPrice;
    uint256 protocolFee = (price * protocolFeePercent) / 1 ether;
    - uint256 subjectFee = (price * subjectFeePercent) / 1 ether;

```

```

    uint256 referralFee = (price * referralFeePercent) / 1 ether;
-   require(msg.value >= price + protocolFee + subjectFee + referralFee,
"NOT_ENOUGH_SENT");
+   require(msg.value >= price + protocolFee + referralFee, "NOT_ENOUGH_SENT");
    uint256 endTime = block.timestamp + subscriptionDuration;
    subscribers[subscriptionsSubject][msg.sender] = endTime;
    setReferralIneligible(msg.sender);

-   sendToSubject(msg.sender, price - protocolFee - subjectFee - referralFee);
+   sendToSubject(subscriptionsSubject, price);
    sendToProtocol(protocolFee);
-   sendToSubject(subscriptionsSubject, subjectFee);
    sendToReferrer(msg.sender, referralFee);
-   sendToSubject(msg.sender, msg.value - (price + protocolFee + subjectFee +
referralFee));
+   sendToSubject(msg.sender, msg.value - (price + protocolFee + referralFee));

-   emit Subscription(msg.sender, subscriptionsSubject, endTime, protocolFee,
referralFee);
+   emit Subscription(msg.sender, subscriptionsSubject, endTime, protocolFee,
referralFee);
}

```

Status

✓ Fixed

[WP-M3] Subscriptions subjects should be able to set their own `subscriptionPrice` .

Medium

Issue Description

The current implementation uses the same subscription price (a global storage variable) for all subjects.

<https://github.com/leaguetechnology/contracts/blob/59d3e75b6c0dd728a8fd83e7aa343b90da9580b9/contracts/LeagueTechV1.sol#L122-L125>

```
122  /// @dev Set the global `subscriptionPrice`
123  function setSubscriptionPrice(uint256 _subscriptionPrice) public onlyOwner {
124      subscriptionPrice = _subscriptionPrice;
125  }
```

<https://github.com/leaguetechnology/contracts/blob/59d3e75b6c0dd728a8fd83e7aa343b90da9580b9/contracts/LeagueTechV1.sol#L243-L264>

```
243  function buySubscription(address subscriptionsSubject) public payable nonReentrant
    whenNotPaused {
244      require(subscriptionsSubject != msg.sender, "NO_SELF_SUBSCRIPTION");
245      require(subscriptionsEnabled[subscriptionsSubject] == 1,
        "SUBJECT_SUBSCRIPTIONS_NOT_ENABLED");
246      require(subscribers[subscriptionsSubject][msg.sender] < block.timestamp,
        "ALREADY_SUBSCRIBED");
247
248      uint256 price = subscriptionPrice;
249      uint256 protocolFee = (price * protocolFeePercent) / 1 ether;
250      uint256 subjectFee = (price * subjectFeePercent) / 1 ether;
251      uint256 referralFee = (price * referralFeePercent) / 1 ether;
252      require(msg.value >= price + protocolFee + subjectFee + referralFee,
        "NOT_ENOUGH_SENT");
253      uint256 endTime = block.timestamp + subscriptionDuration;
254      subscribers[subscriptionsSubject][msg.sender] = endTime;
255      setReferralIneligible(msg.sender);
256
```

```
257     sendToSubject(msg.sender, price - protocolFee - subjectFee - referralFee);
258     sendToProtocol(protocolFee);
259     sendToSubject(subscriptionsSubject, subjectFee);
260     sendToReferrer(msg.sender, referralFee);
261     sendToSubject(msg.sender, msg.value - (price + protocolFee + subjectFee +
    referralFee));
262
263     emit Subscription(msg.sender, subscriptionsSubject, endTime, protocolFee,
    subjectFee, referralFee);
264 }
```

Recommendation

Consider allowing each subject to set their own `subscriptionPrice` .

Status

✓ Fixed

[WP-L4] Unreachable code

Low

Issue Description

<https://github.com/leaguetechnology/contracts/blob/59d3e75b6c0dd728a8fd83e7aa343b90da9580b9/contracts/LeagueTechV1.sol#L274-L292>

```

274  /// @notice Gets the total price of an `amount` of shares with a specified
    `supply`
275  /// @param supply The current supply of shares
276  /// @param amount The amount of shares to buy or sell
277  function getPrice(uint256 supply, uint256 amount) public view returns (uint256) {
278      uint256 adjustedSupply = supply + 2;
279      if (adjustedSupply == 0) {
280          return initialPrice;
281      }
    @@ 282,291 @@
292  }
```

`supply` is a variable of type `uint256` \implies `supply` \geq 0 \implies (`adjustedSupply` = `supply` + 2) \geq 2 \implies `adjustedSupply` \neq 0

Status

✓ Fixed

[WP-L5] Missing sanity check for setter functions.

Low

Issue Description

<https://github.com/leaguetechnology/contracts/blob/59d3e75b6c0dd728a8fd83e7aa343b90da9580b9/contracts/LeagueTechV1.sol#L106-L113>

```
106  function setProtocolFeePercent(uint256 _feePercent) public onlyOwner {
107      protocolFeePercent = _feePercent;
108  }
109
110  /// @dev Set the global `subjectFeePercent`
111  function setSubjectFeePercent(uint256 _feePercent) public onlyOwner {
112      subjectFeePercent = _feePercent;
113  }
```

Recommendation

Consider adding upper bounds to the fees.

Status

✓ Fixed

[WP-G6] Using constants instead of storages can save some gas

Gas

Issue Description

- `initialPrice` cannot be modified.
- `initialPrice` cannot be initialized arbitrarily. `initialize()` does not have any parameters.

Therefore, the functionality of `initialPrice` storage can be implemented using constants to avoid unnecessary gas consumption in the main functionality.

<https://github.com/leaguetechn/contracts/blob/59d3e75b6c0dd728a8fd83e7aa343b90da9580b9/contracts/LeagueTechV1.sol#L15>

```
15  uint256 public initialPrice;
```

<https://github.com/leaguetechn/contracts/blob/59d3e75b6c0dd728a8fd83e7aa343b90da9580b9/contracts/LeagueTechV1.sol#L71-L82>

```
71  function initialize() public initializer {
72      subjectFeePercent = 7 ether / 100;
73      protocolFeePercent = 2 ether / 100;
74      referralFeePercent = 1 ether / 100;
75      initialPrice = 1 ether / 250;
76      subscriptionPrice = 1 ether;
77      subscriptionDuration = 30 days;
78      paused = 1;
79      protocolDev = msg.sender;
80      __Ownable_init();
81      __ReentrancyGuard_init();
82  }
```

<https://github.com/leaguetechn/contracts/blob/59d3e75b6c0dd728a8fd83e7aa343b90da9580b9/contracts/LeagueTechV1.sol#L274-L292>

```

274  /// @notice Gets the total price of an `amount` of shares with a specified
    `supply`
275  /// @param supply The current supply of shares
276  /// @param amount The amount of shares to buy or sell
277  function getPrice(uint256 supply, uint256 amount) public view returns (uint256) {
278      uint256 adjustedSupply = supply + 2;
279      if (adjustedSupply == 0) {
280          return initialPrice;
281      }
282      uint256 sum1 = ((adjustedSupply - 1) * (adjustedSupply) * (2 * (adjustedSupply
- 1) + 1)) / 6;
283      uint256 sum2 = ((adjustedSupply - 1 + amount) *
284          (adjustedSupply + amount) *
285          (2 * (adjustedSupply - 1 + amount) + 1)) / 6;
286      uint256 summation = (80 ether / 100) * (sum2 - sum1);
287      uint256 price = ((50 ether / 100) * summation * initialPrice) / 1 ether / 1
ether;
288      if (price < initialPrice) {
289          return initialPrice;
290      }
291      return price;
292  }

```

<https://github.com/leaguetechn/contracts/blob/59d3e75b6c0dd728a8fd83e7aa343b90da9580b9/contracts/LeagueTechV1.sol#L144-L215>

```

@@ 144,147 @@
148  function buyShares(address sharesSubject, uint256 amount) public payable
    nonReentrant whenNotPaused {
    @@ 149,150 @@
151      uint256 price = getPrice(supply, amount);
    @@ 152,178 @@
179  }
180
    @@ 181,184 @@
185  function sellShares(address sharesSubject, uint256 amount) public payable
    nonReentrant whenNotPaused {

```



```
@@ 186,188 @@  
189     uint256 price = getPrice(supply - amount, amount);  
@@ 190,214 @@  
215 }
```

Status

✓ Fixed

Appendix

Timeliness of content

The content contained in the report is current as of the date appearing on the report and is subject to change without notice, unless indicated otherwise by WatchPug; however, WatchPug does not guarantee or warrant the accuracy, timeliness, or completeness of any report you access using the internet or other means, and assumes no obligation to update any information following publication.

Disclaimer

This report is based on the scope of materials and documentation provided for a limited review at the time provided. Results may not be complete nor inclusive of all vulnerabilities. The review and this report are provided on an as-is, where-is, and as-available basis. You agree that your access and/or use, including but not limited to any associated services, products, protocols, platforms, content, and materials, will be at your sole risk. Smart Contract technology remains under development and is subject to unknown risks and flaws. The review does not extend to the compiler layer, or any other areas beyond the programming language, or other programming aspects that could present security risks. A report does not indicate the endorsement of any particular project or team, nor guarantee its security. No third party should rely on the reports in any way, including for the purpose of making any decisions to buy or sell a product, service or any other asset. To the fullest extent permitted by law, we disclaim all warranties, expressed or implied, in connection with this report, its content, and the related services and products and your use thereof, including, without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement. We do not warrant, endorse, guarantee, or assume responsibility for any product or service advertised or offered by a third party through the product, any open source or third-party software, code, libraries, materials, or information linked to, called by, referenced by or accessible through the report, its content, and the related services and products, any hyperlinked websites, any websites or mobile applications appearing on any advertising, and we will not be a party to or in any way be responsible for monitoring any transaction between you and any third-party providers of products or services. As with the purchase or use of a product or service through any medium or in any environment, you should use your best judgment and exercise caution where appropriate. FOR AVOIDANCE OF DOUBT, THE REPORT, ITS CONTENT, ACCESS, AND/OR USAGE THEREOF, INCLUDING ANY ASSOCIATED SERVICES OR MATERIALS, SHALL NOT BE CONSIDERED OR RELIED UPON AS ANY FORM OF FINANCIAL, INVESTMENT, TAX, LEGAL, REGULATORY, OR OTHER ADVICE.