BURRA SEC

Probable Security Review

Reviewed by: Goran Vladika, DemoreXTess, 0xAlix2, Shred Security

11th November - 13 November, 2025

Probable Security Review Report

Burra Security

November 19, 2025

Introduction

A time-boxed security review of the **Probable** protocol was done by **Burra Security** team, focusing on the security aspects of the smart contracts.

Disclaimer

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

About Burra Security

Burra Sec offers security auditing and advisory services with a special focus on cross-chain and inter-operability protocols and their integrations.

About UmaCtfAdapter Fork

The contracts form a prediction market (fork of Polymarket) where they utilise the UMA oracle V2 on Polygon and resolve the markets on BSC. Layerzero is used to bridge over the answer from Polygon to BSC.

`UmaCrossChainAdapter.sol` is the adapter contract on Polygon that handles the interaction with UMA and bridge over the payouts through LayerZero.

`UmaCtfCrossChainAdapter.sol` is the adapter contract on BSC that receives the answer from LayerZero and settles the market and ctf tokens.

Severity classification

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

Impact - The technical, economic, and reputation damage from a successful attack

Likelihood - The chance that a particular vulnerability gets discovered and exploited

Severity - The overall criticality of the risk

Informational - Findings in this category are recommended changes for improving the structure, usability, and overall effectiveness of the system.

Findings Summary

ID	Title	Severity	Status
L-1	`getExpectedPayouts` cannot be used for copied questions	Low	Ack
L-2	Mismatch between LayerZero quote and send	Low	Resolved
I-1	`payInLzToken` is unneeded in `quoteResolve`	Info	Resolved
I-2	`flag`/`unflag` mechanism is redundant under a trusted-admin model	Info	Ack
I-3	Incorrect comments in UmaCrossChainAdapter related to ignore price	Info	Resolved
I-4	UmaCtfCrossChainAdapter storage optimization	Info	Ack
I-5	Immutable collateral whitelist and optimistic oracle can be updated in the future in UMA	Info	Ack
I-6	Unused internal function in UmaCrossChainAdapter	Info	Resolved
I-7	Resolving external questions may fail even if it actually can be resolved	Info	Ack

Detailed Findings

[L-01] `getExpectedPayouts` cannot be used for copied questions

Target

- UmaCrossChainAdapter.sol

Severity

- Impact: Low
- Likelihood: Low

Description

The `getExpectedPayouts` function reverts for questions that were not created by the protocol. This happens because it begins by fetching storage from the local `questions` mapping and enforcing:

```
if (!_isInitialized(questionData)) revert NotInitialized();
if (_isFlagged(questionData)) revert Flagged();
if (questionData.paused) revert Paused();
if (!_hasPrice(questionData)) revert PriceNotAvailable();
```

Externally sourced questions are not initialized in the local mapping, so `_isInitialized(questionData)` will always return false, causing the function to revert before the Oracle lookup.

As a result, external questions cannot query or view their final UMA prices using this method.

Recommendation

Consider introducing a function to fetch expected payouts for externally questions by directly querying the Oracle, bypassing local question state checks.

Probable

Acknowledged, if it was not initialized, it can be directly queried in optimisticOracle contract.

BurraSec

Acknowledged

[L-02] Mismatch between LayerZero quote and send

Target

- UmaCrossChainAdapter.sol

Severity

- Impact: Low
- Likelihood: Medium

Description

`UmaCrossChainAdapter::quoteResolve`, used to quote the fee for sending LayerZero message, encodes cross-chain message like this:

```
1 bytes memory message = abi.encode(payouts);
```

But the actual message sent in `_sendPayouts` is endcoded like this:

```
1 bytes memory message = abi.encode(questionID, payouts);
```

Since quoting does not encode `questionID`, the quoted fee amount will always be lower than the fee that will be actually required. The difference is not significant (our benchmark shows 0.14% lower amount), however even 1 wei diff on the lower side would cause `resolve` to revert if the fee provided via msg.value exactly matches the quoted fee.

On the other hand, the typical flow for admin would look something like this:

- admin calls `quoteResolve` off-chain to get the expected fee
- admin adds value buffer on top of the quoted amount (ie. 5%), to account for possible fee change due to the changing gas prices between quoting and sending
- admin calls `resolve`, the fee buffer most probably covers for the difference in the underestimated fee

Thus we consider this bug a low severity issue.

Recommendation

Update `quoteResolve` implementation to include the `questionId` in the encoded message. Consider having internal function which prepares the message and can be used by both `quoteResolve` and `_sendPayouts`.

Probable

Fixed

BurraSec

Fix verified

[I-01] `payInLzToken` is unneeded in `quoteResolve`

Target

- UmaCrossChainAdapter.sol

Severity

INFO

Description

`quoteResolve` exposes a `payInLzToken` flag and forwards it to `_quote(...)`. However, the actual send path in `_sendPayouts` always pays fees in **native** via:

```
_lzSend(
  dstEid,
  message,
  combineOptions(dstEid, SEND, options),
  MessagingFee(msg.value, 0), // native fee only; ZRO always 0
  payable(msg.sender)
);
```

Because `_sendPayouts` hardcodes `MessagingFee(msg.value, 0)`, any `payInLzToken=` **`true`** quote is not honored and is thus unneeded.

Recommendation

Consider removing the `payInLzToken` parameter from `quoteResolve` and always pass **`false`** to `_quote(...)`.

Probable

We will remove it and hardcode it as false

BurraSec

Fix verified

[I-02] `flag/unflag` mechanism is redundant under a trusted-admin model

Target

- UmaCrossChainAdapter.sol

Severity

INFO

Description

In the original UMA CTF Adapter, any user could trigger the resolution process once the UMA Optimistic Oracle finalized a price. Because resolution was permissionless, the `flag()` mechanism acted as a safety switch it allowed the protocol to temporarily halt or delay public resolutions until an admin could step in and manually resolve.

However, in `UmaCrossChainAdapter`, only the admin can call `resolve()`, which means the admin already has full control over when and how resolutions occur. `flag` has additional pause feature however it can already be achieved by calling `pause` function. As a result, the `flag()` mechanism is redundant.

Recommendation

Consider removing the `flag()` and `unflag()` mechanism.

Probable

Acknowledged, will not be updated

BurraSec

Acknowledged

[I-03] Incorrect comments in UmaCrossChainAdapter related to ignore price

Target

- UmaCrossChainAdapter.sol

Severity

INFO

Description

The resolve for own questions will reset the question if price is ignorePrice:

```
// If the OO returns the ignore price, reset the question
if (price == _ignorePrice()) return _reset(address(this), questionID,
    true, questionData);
```

For the external questions, TX reverts in case OO returns the ignore price. But the code comment is incorrect:

```
// If the OO returns the ignore price, reset the question
if (price == _ignorePrice()) revert InvalidOOPrice();
```

Recommendation

Update the misleading comment. Also consider updating the `resolve`'s natspec which currently covers only the own-question variant:

```
/// Resets the question if the price returned by the OO is the Ignore
    price
...
function resolve(
```

Probable

Fixed

BurraSec

Fix verified

[I-04] UmaCtfCrossChainAdapter storage optimization

Target

- UmaCtfCrossChainAdapter.sol

Severity

INFO

Description

`UmaCtfCrossChainAdapter`'s storage layout can be optimized to save a lot of unecessary gas spending.

`UmaCtfCrossChainAdapter` stores questions in a map:

```
mapping(bytes32 => QuestionData) public questions;
```

where this is the struct:

```
struct QuestionData {
    /// @notice The address of the question creator
    address creator;
    /// @notice Data used to resolve a condition
    bytes ancillaryData;
    /// @notice Whether the question has been resolved
    bool resolved;
}
```

Field `creator` is unused and CAN be fetched off-chain from `QuestionInitialized` event, so no need to store it on-chain.

Also no need to store `ancillaryData`, which takes multiple slots for every question stored. Hash of `ancillaryData` is used as mapping key, while raw data is used only for initialization check:

```
    function _isInitialized(QuestionData storage questionData) internal
        view returns (bool) {
        return questionData.ancillaryData.length > 0;
```

```
3       }
```

But initialization check could be done with a simple bool instead, ie.:

```
1  struct QuestionData {
2      bool initialized;
3      bool resolved;
4  }
```

Recommendation

Consider simplifying `UmaCtfCrossChainAdapter`'s storage layout to save on gas spent by admin.

Probable

We think it is fine, we will not update it.

BurraSec

Acknowledged

[I-05] Immutable collateral whitelist and optimistic oracle can be updated in the future in UMA

Target

- UmaCrossChainAdapter

Severity

INFO

Description

At deployment, we initialize 3 contract addresses: Finder, Collateral Whitelist and Optimistic Oracle V2 and initialized as immutable. However, this can be updated in UMA side. UMA always use finder to get other contract addresses and admins has ability to change in the future.

```
    /// @notice Optimistic Oracle
    IOptimisticOracleV2 public immutable optimisticOracle;

    /// @notice Collateral Whitelist
    IAddressWhitelist public immutable collateralWhitelist;

...

// UMA
    function _getCollateralWhitelist() internal view returns (
        AddressWhitelist) {
        return AddressWhitelist(finder.getImplementationAddress(
            OracleInterfaces.CollateralWhitelist));
    }

    function _getStore() internal view returns (StoreInterface) {
        return StoreInterface(finder.getImplementationAddress(
            OracleInterfaces.Store));
    }
```

It's very unlikely to see it's changed in the future but if it happens UMACrossChainAdapter can't update this addresses due to immutable design.

Recommendation

Consider adding internal functions to get up-to-date addresses of collateral whitelist and optimistic oracle v2.

Probable

Acknowledged, will not be updated

BurraSec

Acknowledged

[I-06] Unused internal function in UmaCrossChainAdapter

Target

- UmaCrossChainAdapter.sol

Severity

INFO

Description

This internal function in `UmaCrossChainAdapter` is unused and can be removed:

```
function _hasPrice(address requester, uint256 timestamp, bytes memory
    appendedAncillaryData)
     internal
     view
     returns (bool)
{
     return optimisticOracle.hasPrice(requester, YES_OR_NO_IDENTIFIER,
        timestamp, appendedAncillaryData);
}
```

Recommendation

Remove the unused internal function

Probable

Function is removed

BurraSec

FIx verified

[I-07] Resolving external questions may fail even if it actually can be resolved

Target

- UmaCrossChainAdapter

Severity

INFO

Description

Resolving external questions can fail due to following check:

```
if (
    optimisticOracle.getState(creator, YES_OR_NO_IDENTIFIER, timestamp,
        appendedAncillaryData)
        != State.Settled
) revert NotReadyToResolve();
```

This check actually is unnecessary because if it has a price for question, it means we can resolve it by permissionless `settle` function in V2 Oracle.

Recommendation

If it's not settled but it has a price, instead consider calling `settle` function in Oracle for external question and then continue resolving process.

Probable

Acknowledged, will not be fixed

BurraSec

Acknowledged