AETHERAS

CryptoSports

Smart Contract Audit

Table Of Contents

1. Version

Version	Date	Description
0.1	Oct. 11, 2022	Initial Draft

2. Introduction

The following document provides the result of the audit performed by Aetheras at the customer's request. The audit goal is a general review of the smart contracts structure, critical/major bugs detection and issuing the general recommendations.

We have audited the CryptoSports repository, with the following files audited:

- engine/outright/BettingHandler.sol
- engine/outright/IOutrightBetEngineV1.sol
- engine/outright/OutrightBetEngineV1.sol
- engine/single/BettingHandler.sol
- engine/single/ISingleBetEngineV1.sol
- engine/single/SingleBetEngineV1.sol
- interface/oracle/IOracleV1.sol
- interface/outright/IOutrightBetV1.sol
- interface/single/ISingleBetV1.sol
- lib/oracle/BytesEventMap.sol
- lib/oracle/BytesOutrightMap.sol
- lib/oracle/BytesResult.sol
- lib/oracle/EventHandlerV1.sol
- lib/oracle/EventResultV1.sol
- lib/oracle/OutrightHandlerV1.sol
- lib/oracle/OutrightResultV1.sol
- lib/outright/BytesTicket.sol
- lib/outright/BytesTicketMap.sol
- lib/outright/OutrightBetTicketV1.sol
- lib/single/BytesTicket.sol
- lib/single/BytesTicketMap.sol
- lib/single/SingleBetTicketV1.sol
- lib/EnumerableSet.sol
- lib/EvolutionToken.sol
- lib/Receipt.sol
- lib/ResultHandler.sol
- lib/SettleConditionsHandler.sol
- lib/Settlement.sol
- lib/SettlementHandler.sol
- lib/SettlePatternHandler.sol
- lib/Signature.sol
- lib/Status.sol
- lib/TestNextfortuneToken.sol

- lib/Type.sol
- proxy/ERC1967Proxy.sol
- utils/BaseOwnable.sol
- utils/BaseOwnablePausable.sol
- utils/Contract.sol
- OracleV1.sol
- OutrightBetV1.sol
- SingleBetV1.sol

2.1 Methodology

The methodology is not a strict formal procedure, but rather a collection of methods and tactics that combined differently and tuned for every particular project, depending on the project structure and and used technologies, as well as on what the client is expecting from the audit. In current audit we use:

- **General Code Assessment**. The code is reviewed for clarity, consistency, style, and for whether it follows code best practices applicable to the particular programming language used. We check indentation, naming convention, commented code blocks, code duplication, confusing names, confusing, irrelevant, or missing comments etc. At this phase we also understand overall code structure.
- **Entity Usage Analysis**. Usages of various entities defined in the code are analyzed. This includes both: internal usages from other parts of the code as well as potential external usages. We check that entities are defined in proper places and that their visibility scopes and access levels are relevant. At this phase we understand overall system architecture and how different parts of the code are related to each other.
- **Access Control Analysis.** For those entities, that could be accessed externally, access control measures are analyzed. We check that access control is relevant and is done properly. At this phase we understand user roles and permissions, as well as what assets the system ought to protect.
- **Code Logic Analysis.** The code logic of particular functions is analyzed for correctness and efficiency. We check that code actually does what it is supposed to do, that algorithms are optimal and correct, and that proper data types are used. We also check that external libraries used in the code are up to date and relevant to the tasks they solve in the code. At this phase we also understand data structures used and the purposes they are used for.

3 Results

3.1 Lack of Documentation

- **Severity** Critical

Description: While there is a readme that lists out all function inputs and outputs, and a diagram that outlines what components interact with each other, there isn't a clear specification or whitepaper that describes how the system exactly works.
Client: We'll keep updating the readme, the official website and whitepaper is in progress.

3.2 Lack of Comments in Code

- **Severity** Critical

Description: There is currently no comments in any of the files. Ideally the code should be well documented not just for the sake of other developers, but for the maintainers themselves.
Client: Have added some comments in the code, and we'll add more code to explain what and how the system works and how to maintain these contracts.

3.3 Inconsistent folder structure

- **Severity** Minor

Description: Interfaces and contracts are mixed under the engine folder.
Recommendation: Since the contracts are not self-contained, and reference files from the lib folder, the files should be moved to folders that fit them. i.e. to interface and lib folders, etc.
Client: Have fixed according to the recommendations.

3.4 Unnecessary interface folders

- **Severity** Minor
- **Source**
  interface/oracle/
  interface/outright/
  interface/single/

Description: There are extra layers of folders under the interface folder. While it may be useful when there are numerous versions of the interface, the naming convention of the files themselves should keep it relatively organized already.
Recommendation: Consider removing that layer of folder structure.
Client: Have fixed according to the recommendations.

3.5 Redundant files

- **Severity** Minor
- **Source**

engine/outright/BettingHandler.sol
engine/single/BettingHandler.sol

Description: These two files appear to do the same exact thing and end up referencing *src/lib/Type.sol* despite referencing different files.
Recommendation: Unless there is an actual difference between the files, they should be consolidated into a single file.
Client: These two files may be slightly different after a few versions of update, we want to keep the logic separated to make sure we don't have to split them off soon, if there are no differences after a few updates, we'll consider to merge them into a single file.

3.6 Revert CustomErrors

- **Severity** Minor
- Source SingleBetV1.sol

Description: When possible, it is recommended to call `revert()` on CustomErrors instead of strings, since errors are encoded using only four bytes.
Recommendation: Use custom errors

Client: Have fixed according to the recommendations.
Line 128

```
    revert("OutrightBet: token transfer error");
```

3.7 Pausing is Not Used Properly

- **Severity** Critical
- Source SingleBetV1.sol

Description: Simply turning on / off pausing doesn't prevent users from calling the smart contract. The modifiers need to be used to restrict whether it can only be called when the contract is paused or not. From the documentation, "*Note that they will not be pausable by simply including this module, only once the modifiers are put in place.*"
Recommendation: Figure out which functions should only work when paused, and which functions should only work when unpaused.

Client: Have fixed according to the recommendations.

3.8 Dead Code

- **Severity** Minor
- Source utils/Contract.sol

Description: There is nothing here and nothing appears to be referencing this file.
Recommendation: Remove the file if it is truly dead code.

Client:

3.9 Unnecessary Parentheses

- **Severity** Minor
- Source SingleBetV1.sol

Description: It is unnecessary to add parentheses to modifiers that take no arguments. Specifically, `OnlyOwner` does not take any arguments.
Recommendation: Remove the parentheses.

Client: Have fixed according to the recommendations.

Line 181

3.10 Unnecessary Parentheses

- **Severity** Minor
- Source SingleBetV1.sol

Description: It is unnecessary to add parentheses to modifiers that take no arguments. Specifically, `OnlyOwner` does not take any arguments.
Recommendation: Remove the parentheses.

Client: Have fixed according to the recommendations.

Line 189

3.11 Unnecessary Parentheses

- **Severity** Minor
- Source SingleBetV1.sol

Description: It is unnecessary to add parentheses to modifiers that take no arguments. Specifically, `OnlyOwner` does not take any arguments.
Recommendation: Remove the parentheses.

Client: Have fixed according to the recommendations.

Line 197

3.12 Unnecessary Parentheses

- **Severity** Minor
- Source SingleBetV1.sol

Description: It is unnecessary to add parentheses to modifiers that take no arguments. Specifically, `OnlyOwner` does not take any arguments.
Recommendation: Remove the parentheses.

Client:

Line 205

3.13 Unnecessary Parentheses

- **Severity** Minor
- Source SingleBetV1.sol

Description: It is unnecessary to add parentheses to modifiers that take no arguments. Specifically, `OnlyOwner` does not take any arguments.
Recommendation: Remove the parentheses.

Client: Have fixed according to the recommendations.

Line 213

3.14 Unnecessary Parentheses

- **Severity** Minor
- Source SingleBetV1.sol

Description: It is unnecessary to add parentheses to modifiers that take no arguments. Specifically, `OnlyOwner` does not take any arguments.
Recommendation: Remove the parentheses.

Client: Have fixed according to the recommendations.

Line 221

3.15 Unnecessary Parentheses

- **Severity** Minor
- Source SingleBetV1.sol

Description: It is unnecessary to add parentheses to modifiers that take no arguments. Specifically, `OnlyOwner` does not take any arguments.
Recommendation: Remove the parentheses.

Client: Have fixed according to the recommendations.

Line 234

3.16 Unnecessary Parentheses

- **Severity** Minor
- Source SingleBetV1.sol

Description: It is unnecessary to add parentheses to modifiers that take no arguments. Specifically, `OnlyOwner` does not take any arguments.
Recommendation: Remove the parentheses.

Client: Have fixed according to the recommendations.

Line 312

3.17 Unnecessary Check and Return

- **Severity** Minor
- Source SingleBetV1.sol

Description: `_pause()` guarantees to succeed, so performing a check and returning a boolean afterwards is unnecessary. The only time it fails is if the contract is already paused.
Recommendation: Remove the if check and the return value.

Client: Have fixed according to the recommendations.

Lines 225-227

```
        if (paused()) {
            return true;
        }
        return false;
```

3.18 Unnecessary Check and Return

- **Severity** Minor
- Source SingleBetV1.sol

Description: `_unpause()` guarantees to succeed, so performing a check and returning a boolean afterwards is unnecessary. The only time it fails is if the contract is already paused.
Recommendation: Remove the if check and the return value.

Client: Have fixed according to the recommendations.

Lines 238-240

```
        if (!paused()) {
            return true;
        }
        return false;
```

3.19 Use Calldata instead of Memory When Possible

- **Severity** Minor
- **Source** SingleBetV1.sol

Description: When applicable, gas costs are lower when using `calldata` instead of `memory`, because the value is allocated by the caller.
Recommendation: Use calldata.

Client: Have fixed according to the recommendations.

Line 96

```
function placeBet(
    uint256 version,
    bytes memory signedTicketBytes
)
```

3.20 Use Calldata instead of Memory When Possible

- **Severity** Minor
- **Source** SingleBetV1.sol

Description: When applicable, gas costs are lower when using `calldata` instead of `memory`, because the value is allocated by the caller.
Recommendation: Use calldata.

Client: Have fixed according to the recommendations.

Line 137

```
function rejectBet(
    uint256[] memory transIds
)
```

3.21 Consider Checking if Transfers Fail

- **Severity** Critical
- **Source** SingleBetV1.sol

Description: Ideally, this should never fail. However, since we are not guaranteed which engine code will be run, there is potential for this transfer to fail. Consider checking and reverting if the transfer fails.
Recommendation: Add the check similar lines 116-119 of the same file.

Client: Have fixed according to the recommendations.

Line 71

```
for (uint8 j; j < settlement.payments.length; j++) {
    Settlement.ERC20PaymentV1 memory payment = settlement.payments[j];
    IERC20(payment.token).transfer(payment.receiver, payment.amount);
}
```

3.22 Potentially Redundant Files

- **Severity** Moderate
- **Source** SingleBetV1.sol
  OutrightBetV1.sol

Description: Based on the content, it appears that the only differences are in the naming and one snippet of code that is missing in OutrightBetV1, which probably should be included (this will be mentioned in another finding). All the function signature types, as well as external calls have the same signatures, which means these two contracts are essentially identical.

Another part that is different seems to be line 73 of OutrightBetV1.sol, but it appears that settlement.transId should be the same as transIds[i] in this case.
Recommendation: Consider consolidating them into one contract if possible, or if not, put the common code into a base contract to inherit from.

Client: We want to split the ticket storage of single bet and outright ticket to a different contract, and there might be some chance that SingleBet and OutrightBet will have some different function, so we'll still keep this with two contracts.

3.23 Should Revert if Failed to Transfer

- **Severity** Critical
- Source OutrightBetV1.sol

Description: Similar to SingleBetV1, the call should revert if one of the transfers fails.
Recommendation: Add the check and revert

Client: Have fixed according to the recommendations.
Lines: 115-118

3.24 Pausing is Not Used Properly

- **Severity** Critical
- Source OutrightBetV1.sol

Description: Simply turning on / off pausing doesn't prevent users from calling the smart contract. The modifiers need to be used to restrict whether it can only be called when the contract is paused or not. From the documentation, "*Note that they will not be pausable by simply including this module, only once the modifiers are put in place.*"

Recommendation: Figure out which functions should only work when paused, and which functions should only work when unpaused.
Client: Have fixed according to the recommendations.

3.25 Unnecessary Parentheses

- Severity Minor
- Source OutrightBetV1.sol

Description: It is unnecessary to add parentheses to modifiers that take no arguments. Specifically, `OnlyOwner` does not take any arguments.
Recommendation: Remove the parentheses.

Client: Have fixed according to the recommendations.

Line 175

3.26 Unnecessary Parentheses

- Severity Minor
- Source OutrightBetV1.sol

Description: It is unnecessary to add parentheses to modifiers that take no arguments. Specifically, `OnlyOwner` does not take any arguments.
Recommendation: Remove the parentheses.

Client: Have fixed according to the recommendations.

Line 188

3.27 Unnecessary Parentheses

- Severity Minor
- Source OutrightBetV1.sol

Description: It is unnecessary to add parentheses to modifiers that take no arguments. Specifically, `OnlyOwner` does not take any arguments.
Recommendation: Remove the parentheses.

Client: Have fixed according to the recommendations.

Line 201

3.28 Unnecessary Parentheses

- Severity Minor
- Source OutrightBetV1.sol

Description: It is unnecessary to add parentheses to modifiers that take no arguments. Specifically, `OnlyOwner` does not take any arguments.
Recommendation: Remove the parentheses.

Client: Have fixed according to the recommendations.

Line 212

3.29 Unnecessary Parentheses

- Severity Minor
- Source OutrightBetV1.sol

Description: It is unnecessary to add parentheses to modifiers that take no arguments. Specifically, `OnlyOwner` does not take any arguments.
Recommendation: Remove the parentheses.

Client: Have fixed according to the recommendations.

Line 220

3.30 Unnecessary Parentheses

- Severity Minor
- Source OutrightBetV1.sol

Description: It is unnecessary to add parentheses to modifiers that take no arguments. Specifically, `OnlyOwner` does not take any arguments.
Recommendation: Remove the parentheses.

Client: Have fixed according to the recommendations.

Line 228

3.31 Unnecessary Parentheses

- Severity Minor
- Source OutrightBetV1.sol

Description: It is unnecessary to add parentheses to modifiers that take no arguments. Specifically, `OnlyOwner` does not take any arguments.
Recommendation: Remove the parentheses.

Client: Have fixed according to the recommendations.

Line 236

3.32 Unnecessary Parentheses

- Severity Minor
- Source OutrightBetV1.sol

Description: It is unnecessary to add parentheses to modifiers that take no arguments. Specifically, `OnlyOwner` does not take any arguments.
Recommendation: Remove the parentheses.

Client: Have fixed according to the recommendations.

Line 309

3.33 Unnecessary Check and Return

- Severity Minor
- Source OutrightBetV1.sol

Description: `_pause()` guarantees to succeed, so performing a check and returning a boolean afterwards is unnecessary. The only time it fails is if the contract is already paused.
Recommendation: Remove the if check and the return value.

Client: Have fixed according to the recommendations.

Lines 179-182

```
        if (paused()) {
            return true;
        }
        return false;
```

3.34 Unnecessary Check and Return

- Severity Minor
- Source OutrightBetV1.sol

Description: `_unpause()` guarantees to succeed, so performing a check and returning a boolean afterwards is unnecessary. The only time it fails is if the contract is already paused.
Recommendation: Remove the if check and the return value.

Client: Have fixed according to the recommendations.

Lines 238-240

```
        if (!paused()) {
            return true;
        }
```

```
        return false;
```

3.35 Versioning is Not Fully Supported

- Severity Moderate
- Source OracleV1.sol

Description: Despite having an `eventVersionMap`, `eventId` will be using the default value of 0 if the version is greater than 1. This can potentially cause strange behaviors if other parts of the system are not prepared to handle this.0
Recommendation: For V1, add a `require()` call to make sure the version is 1. For future versions, the recommendation is to require that the version is the same or lower than the latest supported version.

Client: We'll update this contract and limit the maximum and minimum version, this fix will be applied in the next version of Oracle.

Lines 48-50

```
        if (version == 1) {
            eventId = EventHandlerV1.getEventId(resultBytes);
        }
```

3.36 Versioning is Not Fully Supported

- Severity Moderate
- Source OracleV1.sol

Description: Despite having an `outrightVersionMap`, `leagueId` will be using the default value of 0 if the version is not 1. So right now it's possible to set an `outrightVersionEngine` to 2, and when trying to `uploadOutrightResult`, the leagueId will always be set to the default of 0.

Recommendation: For V1, add a `require()` call to make sure the version is 1. For future versions, the recommendation is to require that the version is the same or lower than the latest supported version.

Client: We'll update this contract and limit the maximum and minimum version, this fix will be applied in the next version of Oracle.

Lines 77-79

```
        if (version == 1) {
            leagueId = OutrightHandlerV1.getLeagueId(resultBytes);
        }
```

3.37 Versioning is Not Fully Supported

- Severity Moderate
- Source OracleV1.sol

Description: Since we are aware that this version of the contract only supports version 1, we should restrict the version for `setEventResultEngine()` and `setOutrightResultEngine()`. **Recommendation:** Either add checks in these two functions, or change the function signatures to omit the version parameter.

Client: We'll update this contract and limit the maximum and minimum version, and make sure no other version of the engine will be set. This fix will be applied in the next version of Oracle.

Lines 92-112

3.38 Versioning is Not Fully Supported

- Severity Moderate
- Source OracleV1.sol

Description: A default ResultStatus is returned if the version is not 1.
Recommendation: For V1, add a `require()` call to make sure the version is 1. For future versions, the recommendation is to require that the version is the same or lower than the latest supported version. This may not be needed if the set and upload functions restrict versions.

Client: We may support few versions of event / outright result in the same version of Oracle, We'll update this contract and limit the maximum and minimum version, and make sure no other version of the engine will be set. This fix will be applied in the next version of Oracle.

Lines 135-138

```
        if (isExist) {
            if (result.version == 1) {
                return EventHandlerV1.getEventStatus(result.resultBytes);
            }
```

3.39 Versioning is Not Fully Supported

- Severity Moderate
- Source OracleV1.sol

Description: Default `homeScore` and `awayScore` are returned if the version is not 1.
Recommendation: For V1, add a `require()` call to make sure the version is 1. For future versions, the recommendation is to require that the version is the same or lower than the latest supported version. This may not be needed if the set and upload functions restrict versions.

Client: We may support few versions of event / outright result in the same version of Oracle, We'll update this contract and limit the maximum and minimum version, and make sure no other version of the engine will be set. This fix will be applied in the next version of Oracle.

Lines 162-165

```
        if (isExist) {
            if (result.version == 1) {
                return EventHandlerV1.getScore(betType, resource1, result.resultBytes);
            }
```

3.40 Versioning is Not Fully Supported

- Severity Moderate
- Source OracleV1.sol

Description: A default `ResultHash` is returned if the version is not 1.
Recommendation: For V1, add a `require()` call to make sure the version is 1. For future versions, the recommendation is to require that the version is the same or lower than the latest supported version. This may not be needed if the set and upload functions restrict versions.

Client: We may support few versions of event / outright result in the same version of Oracle, We'll update this contract and limit the maximum and minimum version, and make sure no other version of the engine will be set. This fix will be applied in the next version of Oracle.

Lines 187-190

```
if (isExist) {
    if (result.version == 1) {
    return EventHandlerV1.getResultHash(betType, resource1, result.resultBytes);
}
```

3.41 Versioning is Not Fully Supported

- Severity Moderate
- Source OracleV1.sol

Description: A default OutrightResult is returned if the version is not 1.
Recommendation: For V1, add a `require()` call to make sure the version is 1. For future versions, the recommendation is to require that the version is the same or lower than the latest supported version. This may not be needed if the set and upload functions restrict versions.

Client: We may support few versions of event / outright result in the same version of Oracle, We'll update this contract and limit the maximum and minimum version, and make sure no other version of the engine will be set. This fix will be applied in the next version of Oracle.

Lines 213-216

```
        if (isExist) {
            if (result.version == 1) {
                return OutrightHandlerV1.getOutrightResult(teamId, result.resultBytes);
            }
```

3.42 Use require instead if - revert

- Severity Minor
- Source OracleV1.sol

Description: Instead of having a `revert` in an else clause, it's clearer to just use `require`.
Recommendation: Use `require()` for code clarity.

Client: Have fixed according to the recommendations.

Lines 139-141

```
        } else {
            revert("OracleV1: result is not uploaded");
        }
```

3.43 Use require instead if - revert

- Severity Minor
- Source OracleV1.sol

Description: Instead of having a `revert` in an else clause, it's clearer to just use `require`.
Recommendation: Use `require()` for code clarity.

Client: Have fixed according to the recommendations.

Lines 166-168

```
        } else {
            revert("OracleV1: result is not uploaded");
        }
```

3.44 Use require instead if - revert

- Severity Minor
- Source OracleV1.sol

Description: Instead of having a `revert` in an else clause, it's clearer to just use `require`.
Recommendation: Use `require()` for code clarity.

Client: Have fixed according to the recommendations.

Lines 191-193

```
        } else {
            revert("OracleV1: result is not uploaded");
        }
```

3.45 Use require instead if - revert

- Severity Minor
- Source OracleV1.sol

Description: Instead of having a `revert` in an else clause, it's clearer to just use `require`.
Recommendation: Use `require()` for code clarity.

Client: Have fixed according to the recommendations.

Lines 217-219

```
        } else {
            revert("OracleV1: result is not uploaded");
        }
```

3.46 Fragile Type Enforcement

- Severity Moderate
- Source OracleV1.sol

Description: Currently `uploadEventResult` is only able to enforce `resultBytes` to be possibly an `eventResult` by calling `result.eventId` inside `getEventId()`. Otherwise, it's possible to pass in an `outrightResult` instead.
Recommendation: Either have an explicit check, or at the very least, comment the code well so future maintainers know the importance.

Client: We add comments to notify the maintainer / developer not to mix it with a function of outright result, we'll also find some other way to prevent using these functions with incorrect data / data type, this will be covered in the future version.

Lines 48-50

```
        if (version == 1) {
            eventId = EventHandlerV1.getEventId(resultBytes);
        }
```

3.47 Fragile Type Enforcement

- Severity Moderate
- Source OracleV1.sol

Description: Currently `uploadOutrightResult` is only able to enforce `resultBytes` to be possibly an `outrightResult` by calling `result.leagueId` inside `getLeagueId()`. Otherwise, it's possible to pass in an `eventResult` instead.
Recommendation: Either have an explicit check, or at the very least, comment the code well so future maintainers know the importance.

Client: We add comments to notify the maintainer / developer not to mix it with a function of outright result, we'll also find some other way to prevent using these functions with incorrect data / data type, this will be covered in the future version.

Lines 77-79

```
        if (version == 1) {
            leagueId = OutrightHandlerV1.getLeagueId(resultBytes);
        }
```

3.48 Unnecessary Parentheses

- Severity Minor
- Source OracleV1.sol

Description: It is unnecessary to add parentheses to modifiers that take no arguments. Specifically, `OnlyOwner` does not take any arguments.
Recommendation: Remove the parentheses.

Client: Have fixed according to the recommendations.

Line 98

3.49 Unnecessary Parentheses

- Severity Minor
- Source OracleV1.sol

Description: It is unnecessary to add parentheses to modifiers that take no arguments. Specifically, `OnlyOwner` does not take any arguments.
Recommendation: Remove the parentheses.

Client: Have fixed according to the recommendations.

Line 109

3.50 Unnecessary Parentheses

- Severity Minor
- Source OracleV1.sol

Description: It is unnecessary to add parentheses to modifiers that take no arguments. Specifically, `OnlyOwner` does not take any arguments.
Recommendation: Remove the parentheses.

Client: Have fixed according to the recommendations.

Line 119

3.51 Redundant Interfaces

- **Severity** Minor
- **Source** engine/outright/IOutrightBetEngineV1.sol engine/single/ISingleBetEngineV1.sol

Description: Besides having different id names (`leagueId` vs `eventId`), the two interfaces have the same functions, parameters, and return types.
Recommendation: Consider merging the interfaces as a betEngine. Since we are directly calling the engine using abi encode, the interface isn't actually being used.
Client: There might be some chance that SingleBetEngine and OutrightBetEngine will have some different function, so we'll still keep this with two files.

3.52 Redundant Structs

- **Severity** Minor
- **Source** lib/oracle/BytesResult.sol

Description: `EventResultV1` and `OutrightResultV1` structs are exactly the same with the same member name and types.
Recommendation: Since these are only storing the bytes and a version number, and they are not getting type checked, they can simply share the same struct.

Client: We want to keep it as two files for a few versions to guarantee there won't be any difference between the event and outright result struct, we may merge them after a few versions update.

3.53 Hardcoded Constants

- **Severity** Minor
- **Source**

lib/oracle/EventHandlerV1.sol
lib/ResultHandler.sol

Description: Currently the handling of `betType` is hardcoded despite it appearing in multiple files. Constants like these are recommended to be referenced from a shared config file.
Recommendation: Put shared constants into a file that can be referenced by others.

Client: Have fixed according to the recommendations.

3.54 Potential Dead Code

- **Severity** Minor
- **Source** lib/EvolutionToken.sol

Description: This doesn't seem to be referenced by any other contract. If this is still being used, it probably shouldn't be in the lib folder since it is not a library.
Recommendation: Delete if it is indeed dead code.

Client: After the discussion, we'll move these files into another repository.

3.55 Potential Dead Code

- **Severity** Minor
- **Source** lib/TestNextfortuneToken.sol

Description: This doesn't seem to be referenced by any other contract. If this is still being used, it probably shouldn't be in the lib folder since it is not a library.
Recommendation: Delete if it is indeed dead code.

Client: This file is required for the test purpose, and it has been moved to the test folder.

3.56 Inheritance

- **Severity** Minor
- **Source** utils/BaseOwnablePausable.sol

Description: `BaseOwnablePausable` is currently just `BaseOwnable` with `PausableUpgradeable`.
Recommendation: `BaseOwnablePausable` should just inherit `BaseOwnable` and `PausableUpgradeable` instead of reimplementing the functions.

Client: We inherited both ownable and pausable from the OpenZeppelin library without any modification now, we'll try to implement this recommendation in next few versions.

3.57 Redundant Map Implementations

- **Severity** Moderate
- **Source**
lib/oracle/BytesEventMap.sol
lib/oracle/BytesOutrightMap.sol
lib/outright/BytesTicketMap.sol
lib/single/BytesTicketMap.sol

Description: There are currently multiple implementations of an EnumerableMap for different types.
Recommendation: To reduce code repetition as much as possible, the recommendation is to implement it as a generic Map type with private functions, and wrapper functions as wrappers to the underlying map.
Client: We want to keep these files for a few versions to guarantee there won't be any difference between them, we may merge into a single file after a few versions update.

3.58 Unsupported Versions

- Severity Moderate
- Source OutrightBetV1.sol

Description: It's currently possible to set the ticket engine for unsupported versions in `setTicketEngine()`, even though this contract only supports version 1.
Recommendation: To avoid unexpected behaviors, modify `setTicketEngine()` to not have a `version` parameter, and in future versions of this contract, require that the version can't be higher than the newest version.

Client: We'll limit the maximum and minimum version to prevent us setting the engine to an unsupported version, this will be fixed in the next few versions.
Lines: 206-215

3.59 Unsupported Versions

- Severity Moderate
- Source SingleBetV1.sol

Description: It's currently possible to set the ticket engine for unsupported versions in `setTicketEngine()`, even though this contract only supports version 1.
Recommendation: To avoid unexpected behaviors, modify `setTicketEngine()` to not have a `version` parameter, and in future versions of this contract, require that the version can't be higher than the newest version.

Client: We'll limit the maximum and minimum version to prevent us setting the engine to an unsupported version, this will be fixed in the next few versions.

Lines: 175-184