Security Assessment

Predy Protocol

Dec 13th, 2021

Table of Contents

Summary

This report has been prepared for Predy Protocol to discover issues and vulnerabilities in the source code of the Predy Protocol project 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.

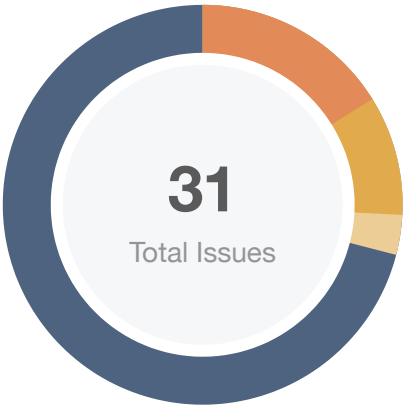
The security assessment resulted in findings that ranged from critical to informational. We recommend addressing these findings to ensure a high level of security standards and industry practices. We suggest recommendations that could better serve the project from the security perspective:

- Enhance general coding practices for better structures of source codes;
- Add enough unit tests to cover the possible use cases;
- Provide more comments per each function for readability, especially contracts that are verified in public;
- Provide more transparency on privileged activities once the protocol is live.

Audit Scope

ID	File	SHA256 Checksum
AMM	lib/AMMLib.sol	64cc6ccab371a2a9d74c0071a7c65319ba0809e40d9fefe5b1f6bba75812cb97
AMC	lib/AdvancedMath.sol	c5aa08f4dfcbd4f78b5dd1270e4004fbe9a52c1cf4f19db870c73df44a56f811
OLC	lib/OptionLib.sol	7368a636e65da7768770b1e56c7ab9c4ebc308b8cb5c0f3359850d8aaad52027
PMC	lib/PredyMath.sol	6857311a0c7157df39f2c87961b6e64b44195ab41f6d8005ad6ea60912f71f9e
PCC	lib/PriceCalculator.sol	d8154b20cdb928eb1c235616e28a8b3464c61885b94cea58649d116321f8b9b0
AMK	AMM.sol	2ece5d66c75e0769bc689ca5cb22440f9bb723145cedd067f699aa23c52f4d76
AMF	AMMFactory.sol	79465eb15cadf9c26289c6cb77336c76ad2e592def45b9f0b56aa1b6d66c780e
FPC	FeePool.sol	ec6450047228abf12420569f2973e09d06a2685dca483fcaecf7bf4275089540
OVC	OptionVault.sol	e3df7aeca69c79a64943aec209e918d4e1e1346aa93042767a817e6c13bb4e42
OVF	OptionVaultFactory.sol	410fbd3d3af0ea3ed444d4dfb417c71ec23d0e0d93538ab1d8fd24eabf32c435
POC	PriceOracle.sol	023bac08cb5d594ffcc038fa1f11a2f11abac80315c7bc0c5dcc1bb541e68c4f
AML	lib/AMMLib.sol	6249294f13cadd9f89c15ebd12ac88d13bbeb149b15e5d36d2de53607b5c9eef
AMP	lib/AdvancedMath.sol	c5aa08f4dfcbd4f78b5dd1270e4004fbe9a52c1cf4f19db870c73df44a56f811
OLK	lib/OptionLib.sol	92c3070c41991e4172f77acc1ff7a7319d6c3d22edbbdaa24e96c19ac20a6881
PMK	lib/PredyMath.sol	6857311a0c7157df39f2c87961b6e64b44195ab41f6d8005ad6ea60912f71f9e
PCK	lib/PriceCalculator.sol	d2b20cba95cd61ce39e7def9660d78b540cfdd4112acc68653be08c0dd15e7d5
ACK	AMM.sol	47f271641c74234cab5d99a0d04cb49f3902db53c6a7675e3ebabfd0db5f0b0b
AFC	AMMFactory.sol	4fd56556cfbedd5385d000d3a70165607626af9efcffd3dee8118392a992b500
FPK	FeePool.sol	ec6450047228abf12420569f2973e09d06a2685dca483fcaecf7bf4275089540
OVK	OptionVault.sol	3b86bc217a789f7d5f751afd40ce679c9bb2ffdea3227bf22787d0d73e300591
OVP	OptionVaultFactory.sol	cc4ccb9a26cb6a286d697092b6a3be99ff49afa49411ba2ef9240a835ce81a58
POK	PriceOracle.sol	acb47968459d7b9a957e7e1d9b7f65db5f00c54af166ad4cddee046616c67fc4

Findings



Critical	**0** (0.00%)
Major	**5** (16.13%)
Medium	**3** (9.68%)
Minor	**1** (3.23%)
Informational	**22** (70.97%)
Discussion	**0** (0.00%)

ID	Title	Category	Severity	Status
Predy Protocol-01	Typos in Codes and Comments	Coding Style	Informational	Partially Resolved
Predy Protocol-02	Gas Optimization on `uint` Type	Gas Optimization, Language Specific	Informational	Acknowledged
ACK-01	Centralization Risk in contract `AMM`	Control Flow, Volatile Code	Major	Acknowledged
ACK-02	Severe Centralization Risk in Contract `AMM`	**Centralization / Privilege**	**Major**	Acknowledged
ACK-03	Missing Emit Events	Coding Style	Informational	Resolved
AMK-01	Function Visibility Optimization	Gas Optimization	Informational	Acknowledged
AMK-02	Check Effect Interaction Pattern Violated	Logical Issue	Informational	Resolved
AMK-03	Incompatibility With Deflationary Tokens	Volatile Code	Informational	Acknowledged
AML-01	Third Party Dependencies on Collateral Token's `approve()` Function	Volatile Code	Informational	Acknowledged
AML-02	Rounding Up in Calculations Using `PredyMath.mulDiv()`	Mathematical Operations	Informational	Resolved
AML-03	Unsafe Casting from `uint128` to `int128`	Mathematical Operations	Minor	Resolved

ID	Title	Category	Severity	Status
AML-04	Potential Sandwich Attack	Volatile Code	Medium	Resolved
AMM-01	Unused Input Parameter in `AMMLib.settleInternal()`	Coding Style	Informational	Resolved
AMP-01	Third Party Dependencies on `AdvancedMath` Library	Volatile Code	Informational	Acknowledged
FPK-01	Centralization Risk in `PriceOracle` and `FeePool`	**Centralization / Privilege**	**Major**	Acknowledged
OLC-01	Third Party Dependencies on Aave Lending Pool	Volatile Code	Informational	Acknowledged
OLC-02	Confusing Logic in Function `settle()`	Logical Issue	Informational	Resolved
OLC-03	`shortLiquidity` Arbitrarily Set to 0	Logical Issue	Informational	Resolved
OLC-04	Redundant Internal Function `getPoolInitialMarginForASeries()`	Coding Style	Informational	Resolved
OLC-05	Check Effect Interaction Pattern Violated	Logical Issue	Informational	Resolved
OLK-01	Inconsistent Comments and Codes	Inconsistency	Informational	Resolved
OLK-02	Rounding Up in Calculations Using `PredyMath.mulDiv()`	Mathematical Operations	Informational	Resolved
OVC-01	Lack of Restriction for `setAMMAddress()` Function	Control Flow, Volatile Code	Medium	Resolved
OVC-02	Incompatibility With Deflationary Tokens	Volatile Code	Informational	Acknowledged
OVK-01	Correctness of Expiry and Series is not Guaranteed in `OptionVault`	Control Flow	Informational	Acknowledged

ID	Title	Category	Severity	Status
OVK-02	Centralization Risk in `OptionVault`	**Centralization / Privilege**	**Major**	Acknowledged
OVK-03	Missing Error Message	Coding Style	Informational	Acknowledged
OVK-04	Lack of Debts Repayment Guarantees	Volatile Code	Medium	Resolved
OVK-05	Missing Emit Events	Coding Style	Informational	Resolved
POC-01	Third Party Dependencies on Chainlink Aggregator	Volatile Code	Informational	Acknowledged
POK-01	Centralization Risk in `PriceOracle` and `FeePool`	**Centralization / Privilege**	**Major**	Acknowledged

Predy Protocol-01 | Typos in Codes and Comments

Category	Severity	Location	Status
Coding Style	Informational	Global	Partially Resolved

Description

There are several typos in the codes and comments:

1. AMM.sol:

- `maneges`
- `recepient`
- `amout`

2. AdvancedMath.sol:

- `Calcurate`
- `Tayler`

3. AMMLib.sol:

- `muptiples`
- `initilize`
- `recepient`
- `avaiable`

4. OptionLib.sol:

- `tractthe`
- `amout`
- `exactlly`

5. OptionVault.sol:

- `recepient`

6. AMMFactory.sol:

- `recepient`

7. PredyMath:

- `remainer`

Recommendation

Recommend correcting all typos in the contract.

Alleviation

[Predy Team]: The team addressed the issue and reflected in the commit hash 5dc1b447b10bbc3bfedd8a4da86681383c6446e1.

Predy Protocol-02 | Gas Optimization on `uint` Type

Category	Severity	Location	Status
Gas Optimization, Language Specific	Informational	Global	Acknowledged

Description

This finding is not a security issue. We noticed that `uint8`, `uint16`, ..., `uint128`, `uint256` are used for variable declaration. We would like to mention that all other `uint` types except `uint256` would be implicitly converted to `uint256` during code execution, since EVM is designed and works with `256bit/32byte`. The implicit conversion would lead to extra gas cost.

Alleviation

[Predy Team]: The team acknowledge the issue and decided to continue using uint128 and others as those variable types have the effect of compressing the state.

ACK-01 | Centralization Risk in contract `AMM`

Category	Severity	Location	Status
Control Flow, Volatile Code	Major	projects/contracts/AMM.sol (0564c2d): 426, 418, 410, 402, 395, 391, 387, 368	Acknowledged

Description

In contract `AMM`, the role `operator` has the authority over the following function:

- `rebalanceCollateral()`
- `setDepositAllowedUntil()`
- `setLockupPeriod()`
- `setAddressAllowedSkippingLockup()`
- `setConfig()`
- `setBot()`
- `setFeeRecipient()`
- `setNewOperator()`

Any compromise to the `operator` account may allow the hacker to take advantage of this and manipulate configuration parameters and `onlyBot` functions.

Recommendation

We advise the client to carefully manage the `operator` account's private key to avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol to be improved via a decentralized mechanism or smart-contract-based accounts with enhanced security practices, e.g., Multisignature wallets.

Indicatively, here is some feasible suggestions that would also mitigate the potential risk at the different level in term of short-term and long-term:

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key;
- Introduction of a DAO/governance/voting module to increase transparency and user involvement.

Alleviation

[Predy Team]: The team acknowledged the issue and adopted to use the multi-signature solution. The operator role will require multiple signers to co-sign the transaction before any funds enter the contract.

ACK-02 | Severe Centralization Risk in Contract `AMM`

Category	Severity	Location	Status
Centralization / Privilege	**Major**	projects/contracts/AMM.sol (0564c2d): 377	Acknowledged

Description

In the contract `AMM`, the role `operator` has the authority over function `changeState()`, which sets the value of `isEmergencyMode`. When `isEmergencyMode` is set to true, functions with `notEmergencyMode` modifier cannot be executed. User will not be able to call function `deposit()`, `withdraw()`, `buy()` and `sell()` in emergency mode.

Any compromise to the `operator` account may allow the hacker to take advantage of this and block user interaction with these four important functions.

Recommendation

We advise the client to carefully manage the `operator` account's private key to avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol to be improved via a decentralized mechanism or smart-contract-based accounts with enhanced security practices, e.g., Multisignature wallets.

Indicatively, here is some feasible suggestions that would also mitigate the potential risk at the different level in term of short-term and long-term:

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key;
- Introduction of a DAO/governance/voting module to increase transparency and user involvement.

Alleviation

[Predy Team]: The team acknowledged the issue and adopted to use the multi-signature solution. The operator role will require multiple signers to co-sign the transaction before any funds enter the contract.

ACK-03 | Missing Emit Events

Category	Severity	Location	Status
Coding Style	Informational	projects/contracts/AMM.sol (0564c2d): 402	Resolved

Description

The function that affects the status of sensitive variables should be able to emit events as notifications to traders and LP providers.

- `setConfig()` in contract `OptionVault`
- `setConfig()` in contract `AMM`
- `setDepositAllowedUntil()` in `AMM`
- `setLockupPeriod()` in `AMM`

Recommendation

Consider adding events for sensitive actions, and emit them in the function.

Alleviation

[Predy Team]: The team addressed the issue and reflected the change in the commit hash bcbfea1007d6ce67c1ae226121f1a2943ac2b184.

AMK-01 | Function Visibility Optimization

Category	Severity	Location	Status
Gas Optimization	Informational	projects/contracts/AMM.sol (b24af21): 391, 400~404	Acknowledged

Description

The following functions are declared as `public`, contain array function arguments, and are not invoked in any of the contracts contained within the project's scope. The functions that are never called internally within the contract should have external visibility.

- `getSeriesState()`
- `getTicks()`

Recommendation

We advise that the functions' visibility specifiers are set to `external` and the array-based arguments change their data location from `memory` to `calldata`, optimizing the gas cost of the function.

Alleviation

[Predy Team]: The team acknowledged the issue and decided no make any further change in the current version.

AMK-02 | Check Effect Interaction Pattern Violated

Category	Severity	Location	Status
Logical Issue	Informational	projects/contracts/AMM.sol (b24af21): 281~286	Resolved

Description

The order of external call/transfer and storage manipulation must follow the check-effect-interaction pattern.

Recommendation

We advise the client to check if storage manipulation is before the external call/transfer operation.LINK

Alleviation

[Predy Team]: The team address the issue and reflected the changes fixed for OLC-06, OVC-01, and POC-01 in the commit hash d294a392156023607ec1f7116234c59b24f5ab3f.

AMK-03 | Incompatibility With Deflationary Tokens

Category	Severity	Location	Status
Volatile Code	Informational	projects/contracts/AMM.sol (b24af21): 1	Acknowledged

Description

When transferring standard ERC20 deflationary tokens, the input amount may not be equal to the received amount due to the charged transaction fee. For example, if a user deposits 100 deflationary tokens (with a 10% transaction fee) in a vault, only 90 tokens actually arrived in the contract. However, the user can still withdraw 100 tokens from the contract, which causes the contract to lose 10 tokens in such a transaction.

Recommendation

We advise the client to regulate the set of collateral and underlying tokens supported and add necessary mitigation mechanisms to keep track of accurate balances if there is a need to support deflationary tokens.

Alleviation

[Predy Team]: The team agreed that this is an acceptable risk, as this version of contracts is only compatible with WETH and USDC which are not deflationary.

AML-01 | Third Party Dependencies on Collateral Token's `approve()` Function

Category	Severity	Location	Status
Volatile Code	Informational	projects/contracts/lib/AMMLib.sol (0564c2d): 138	Acknowledged

Description

In line 133 of contract `AMMLib`, `init()` function approves maximum amount of collateral tokens to `OptionVault`. Presumably, this is to allow `OptionVault` contract to manage the collateral tokens in the pool. While standard ERC20 token allow this design, there is no guarantee that the `approve()` function won't be overridden as customization of collateral token. It is possible that the `approve()` method of the collateral token applies restriction based on the approver's balance.

```
1 IERC20(_pool.collateral).approve(address(_pool.optionVault), MAX_UINT256);
```

Recommendation

We understand that the business logic of `AMMLib` requires that `OptionVault` be able to transfer collateral token to/from PoolInfo. We encourage the team to double check if the design of collateral token is compatible with this line of code.

Alleviation

[Predy Team]: The team acknowledged the issue and decided to use USDC as collateral token. The team believes the trust USDC doesn't override `approve` method.

AML-02 | Rounding Up in Calculations Using `PredyMath.mulDiv()`

Category	Severity	Location	Status
Mathematical Operations	Informational	projects/contracts/lib/AMMLib.sol (0564c2d): 669~670, 775, 1207, 157	Resolved

Description

In functions `calculateTradeStateToBuy()`, `calculateTradeStateToSell()`, `calAvailableSizeForSelling()`, `addBalance()` in contract `AMMLib` and `settle()` in contract `OptionLib`, the lower level call of `PredyMath.mulDiv(uint128 _x, uint128 _y, uint128 _d, bool _roundUp)` has the input parameter `_roundUp == true`, which is $\lceil x \times y/z \rceil$.

In this case, some value calculation like `iv` in `calculateTradeStateToXXX()` and `payout` in `settle()` would be accumulatively inaccurate and would potentially lead to assets loss.

It seems always disable the rounding up would be much safer, and we would like to learn are there special reasons/concerns on this rounding up behaviors?

Alleviation

[Predy Team]:

AML-02: The roundUp is applied here to make sure that the money going into the contract is always equal or greater than the money going out.

For example:

The variable `addBalance`, the amount deposited by LP must be equal or greater than the amount withdrawn if the contract status is the same. This is the reason why has to be roundUp in the deposit function.

However, there was one place where we did not need to use roundUp, we fixed the issue and the change is reflected in the commit hash ae7a6367f4fd38a4be976933ce31f72e0550be4c

OLK-02: The roundUp is applied here to make sure that the funds going out of the contract do not exceed the funds coming in, depending on the result of the division. In this case, the funds paid from the contract to the option holder are "payout" and the funds paid from the contract to the vault owner are "collateral - payout". So the redeem amount is calculated by rounding up "payout" so that the amount paid to the vault owner is equal to or smaller than the amount held by the contract.

AML-03 | Unsafe Casting from `uint128` to `int128`

Category	Severity	Location	Status
Mathematical Operations	Minor	projects/contracts/lib/AMMLib.sol (0564c2d): 815~817, 860~863, 936~939, 994~997	Resolved

Description

We noticed that there are operations casting `uint128` type to `int128` type without evaluating the bounds. Since `uint128` type could hold more numbers than `int128` type, consider adding checks to make sure there are no overflows when casting from `uint` types to `int` types.

Recommendation

Recommend always ensuring the result is still positive as high numbers will cause an overflow to occur here, thereby causing the system to misbehave.

Alleviation

[Predy Team]: The team addressed the issue. The SafeCast library to cast uint128 to int128 is applied and the change is reflected in the commit hash f2f18905732dedb2d723234cede18ba488cf83f2.

Meanwhile, the team found another problem when depositing a large value of 2^18 USDC to test the SafeCast issue. The problem caused ivMove to go to 0.

The solution is to change decimal of ivMove to 12, and relfected in the commit hash 8111adb2ce768b39035437ba1e618b0a6e482869.

AML-04 | Potential Sandwich Attack

Category	Severity	Location	Status
Volatile Code	Medium	projects/contracts/lib/AMMLib.sol (0564c2d)	Resolved

Description

In Predy Protocol, spot price is taken from Chainlink aggregator, and used in calculating the premium. The final value of premium is calculated differently when buying and selling an option, as a spread is added when a trader buys an option. The spread is calculated in the following function.

```
1095      function calculateSpread(
1096          PoolInfo storage _pool,
1097          uint128 _amount,
1098          uint128 _spot,
1099          uint128 _premium
1100      ) internal view returns (uint128) {
1101          return (_amount * (_spot / _pool.configs[BASE_SPREAD] + _premium / 100)) /
(1e10);
1102      }
```

`_pool.configs[BASE_SPREAD]` is set to 250 on line 136, thus 1 / _pool.configs[BASE_SPREAD] = 0.4%. Essentially, the calculation is (0.4% * spot price + 1% * premium) * (_step.stepAmount / 1e10).

The problem lies with Chainlink's deviation threshold. In Chainlink, a new aggregation round starts when a node identifies that the off-chain values deviate by more than the defined deviation threshold from the on-chain value. This value is set to 0.5% at the time of this report, which is not necessarily covered by the spread. This may provide a opportunity for a front-runner to pick up the unused Chainlink transactions for the spot price of the next aggregation, and profit from the difference.

Recommendation

We recommend that the code be changed to make sure the spread covers the deviation threshold so that front-runners cannot profit from this.

Alleviation

[PredyTeam]: The solution the team concluded is to introduce lastPricePerSize to avoid the sandwich attack. The checkLastPrice function checks lastPricePerSize and current pricePerSize, and re-calculate the premium.

Key parameters: lastPricePerSize: the price per size in the last trade recorded per series. lastTradeTime: the timestamp of last trade recorded per series. SafetyPeriod: the maximum interval of trading at which the protocol recalculates the premium.

In the following cases, the protocol uses lastPricePerSize to calculate the premium.

Case Buying

- If lastPricePerSize > the current pricePerSize then the premium must be greater than lastPricePerSize

Case Selling if lastPricePerSize < the current pricePerSize then the premium must be less than lastPricePerSize

Case Special There is one more condition for recalculating the premium. The Protocol checks the block.timestamp is less than 'lastTradeTime + SafetyPeriod'. After safetyPeriod, premiums are not subject to this restriction.

If the oracle price changes significantly and the trades are made within the SafetyPeriod interval, traders will be trading with a larger spread. So, this is a way of replicating this kind of volatility-based spread increase.

Please check the change in the below commit hashs

- 15841882e6c440364668874504c3f5854dd426db
- b45d6e6f703a18e044d9203fc76297b97b2c272a

AMM-01 | Unused Input Parameter in `AMMLib.settleInternal()`

Category	Severity	Location	Status
Coding Style	Informational	projects/contracts/lib/AMMLib.sol (b24af21): 371	Resolved

Description

In function `settleInternal()` of contract AMMLib, the input parameter `_expiryId` is never used.

Alleviation

[Predy Team]: The team addressed the issue and reflected the changes in the commit hash ea486e0fb85a83adb53443b2073da25a90f589b5.

AMP-01 | Third Party Dependencies on `AdvancedMath` Library

Category	Severity	Location	Status
Volatile Code	Informational	projects/contracts/lib/AdvancedMath.sol (0564c2d): 1~4	Acknowledged

Description

The contract is serving as the underlying entity to interact with third party `AdvancedMath` library. The scope of the audit treats 3rd party entities as black boxes and assume their functional correctness. Note that in the real world, code from a 3rd party can be compromised and this may lead to issues in the dependending contracts.

Recommendation

We understand that the business logic requires the usage of `AdvancedMath` library. We encourage the team to constantly monitor the statuses of 3rd parties to mitigate the side effects when unexpected activities are observed.

Alleviation

[Predy Team]: The team acknowledged the issue. However, the Library Contract has no state, the state of this contract cannot be monitored. However, the team will continuously monitor the option price and delta for abnormal values.

FPK-01 | Centralization Risk in `PriceOracle` and `FeePool`

Category	Severity	Location	Status
Centralization / Privilege	**Major**	projects/contracts/FeePool.sol (0564c2d): 15	ⓘ Acknowledged

Description

In the contract `PriceOracle`, the role `owner` has the authority over the following function:

- `setAggregator()`

In the contract `FeePool`, the role `owner` has the authority over the following function:

- `withdraw()`

Any compromise to the `owner` account(s) may allow the hacker to take advantage of this and cause issues.

Recommendation

We advise the client to carefully manage the `owner` account's private key to avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol to be improved via a decentralized mechanism or smart-contract-based accounts with enhanced security practices, e.g., Multisignature wallets.

Indicatively, here is some feasible suggestions that would also mitigate the potential risk at the different level in term of short-term and long-term:

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key;
- Introduction of a DAO/governance/voting module to increase transparency and user involvement.

Alleviation

[Predy Team]: The team acknowledged the issue and adopted to use the multi-signature solution. The operator role will require multiple signers to co-sign the transaction before any funds enter the contract.

OLC-01 | Third Party Dependencies on Aave Lending Pool

Category	Severity	Location	Status
Volatile Code	Informational	projects/contracts/lib/OptionLib.sol (b24af21): 6	Acknowledged

Description

The contract is serving as the underlying entity to interact with Aave lending pool, which is a 3rd party protocol. The scope of the audit treats 3rd party entities as black boxes and assume their functional correctness. However, in the real world, 3rd parties can be compromised and this may lead to lost or stolen assets. In addition, upgrades of 3rd parties can possibly create severe impacts, such as increasing fees of 3rd parties, migrating to new LP pools, etc.

Recommendation

We understand that the business logic of OptionLib requires interaction with Aave lending pool. We encourage the team to constantly monitor the statuses of 3rd parties to mitigate the side effects when unexpected activities are observed.

Alleviation

[Predy Team]: The team acknowledged the issue and will monitor the status of Aave protocol constantly.

OLC-02 | Confusing Logic in Function `settle()`

Category	Severity	Location	Status
Logical Issue	Informational	projects/contracts/lib/OptionLib.sol (b24af21): 447~450	Resolved

Description

The following **require** statement requests that, in order to settle a vault, either the vault is solvent, or the current time is beyond extension period. Essentially, if a vault is solvent, it can be settled at any time. If the requirement is not met, the error message will pop up suggesting that "vault can be settled before extension period".

This poses 2 questions:

- Is it by design that the vault can be settled at any moment, as long as it is solvent?
- Does the error message reflect the reason of mismatch?

```
require(
    !_optionInfo.isInsolvency || _expiration.expiry +
_optionInfo.configs[EXTENSION_PERIOD] >= block.timestamp,
    "OptionLib: vault can be settled before extension period"
);
```

Alleviation

[Predy Team]: The team decided to remove the `require` statement and Insolvency flag in the next version v1.1.

OLC-03 | `shortLiquidity` Arbitrarily Set to 0

Category	Severity	Location	Status
Logical Issue	Informational	projects/contracts/lib/OptionLib.sol (b24af21): 445	Resolved

Description

In line 444 of contract `OptionLib`, the comment suggests that Aave pool should be redeemed before settlement; however, function `settle()` does not directly or indirectly redeem the Aave lending pool, or check if the short liquidity is properly decreased cleared out. In stead, the value of `vault.shortLiquidity` is arbitrarily set to 0.

```
1 // all collaterals in Aave must be redeemed before settlement
2     vault.shortLiquidity = 0;
```

Alleviation

[Predy Team]: The team acknowledged the issue and addressed the item in the next release version v1.1.

v1.1 Change Log In line 412, shortLiquidity will be updated to require the zero check such as "validating shortLiquidity must be 0" at v1.1.

- https://github.com/predyprotocol/predy-v1-contracts/blob/v1.1/contracts/lib/OptionLib.sol#L412

If the protocol can not withdraw full collateral, the team will repay the WETH from out of protocol.

OLC-04 | Redundant Internal Function `getPoolInitialMarginForASeries()`

Category	Severity	Location	Status
Coding Style	Informational	projects/contracts/lib/OptionLib.sol (b24af21): 917	Resolved

Description

Internal function `getPoolInitialMarginForASeries()` is created but not called in any library or function. It does not affect the functionality of the codebase and appear to be either leftovers from test code or older functionality.

Recommendation

We advise that they are removed to better prepare the code for production environments.

Alleviation

[Predy Team]:This function is removed in v1.1

OLC-05 | Check Effect Interaction Pattern Violated

Category	Severity	Location	Status
Logical Issue	Informational	projects/contracts/lib/OptionLib.sol (b24af21): 578~580, 518~523, 534~537	Resolved

Description

The order of external call/transfer and storage manipulation must follow the check-effect-interaction pattern.

Recommendation

We advise the client to check if storage manipulation is before the external call/transfer operation.LINK

Alleviation

[Predy Team]: The team address the issue and reflected the changes fixed for OLC-06, OVC-01, and POC-01 in the commit hash d294a392156023607ec1f7116234c59b24f5ab3f.

OLK-01 | Inconsistent Comments and Codes

Category	Severity	Location	Status
Inconsistency	Informational	projects/contracts/lib/OptionLib.sol (0564c2d): 91~92	Resolved

Description

In library `OptionLib`, `_optionInfo.configs[IM_RATIO]` is set to 200 (or 20%), whereas the comment above indicates a 15%.

Alleviation

[Predy Team]: The team addressed the issue and reflected in the commit hash c963e59cc00978bfb96e01761d72944361e361d9

OLK-02 | Rounding Up in Calculations Using `PredyMath.mulDiv()`

Category	Severity	Location	Status
Mathematical Operations	Informational	projects/contracts/lib/OptionLib.sol (0564c2d): 404	Resolved

Description

In functions `calculateTradeStateToBuy()`, `calculateTradeStateToSell()`, `calAvailableSizeForSelling()`, `addBalance()` in contract `AMMLib` and `settle()` in contract `OptionLib`, the lower level call of `PredyMath.mulDiv(uint128 _x, uint128 _y, uint128 _d, bool _roundUp)` has the input parameter `_roundUp == true`, which is $\lceil x \times y/z \rceil$.

In this case, some value calculation like `iv` in `calculateTradeStateToXXX()` and `payout` in `settle()` would be accumulatively inaccurate and would potentially lead to assets loss.

It seems always disable the rounding up would be much safer, and we would like to learn are there special reasons/concerns on this rounding up behaviors?

Alleviation

[Predy Team]:

AML-02: The roundUp is applied here to make sure that the money going into the contract is always equal or greater than the money going out.

For example:

The variable `addBalance`, the amount deposited by LP must be equal or greater than the amount withdrawn if the contract status is the same. This is the reason why has to be roundUp in the deposit function.

However, there was one place where we did not need to use roundUp, we fixed the issue and the change is reflected in the commit hash ae7a6367f4fd38a4be976933ce31f72e0550be4c

OLK-02: The roundUp is applied here to make sure that the funds going out of the contract do not exceed the funds coming in, depending on the result of the division. In this case, the funds paid from the contract to the option holder are "payout" and the funds paid from the contract to the vault owner are "collateral - payout". So the redeem amount is calculated by rounding up "payout" so that the amount paid to the vault owner is equal to or smaller than the amount held by the contract.

OVC-01 | Lack of Restriction for `setAMMAddress()` Function

Category	Severity	Location	Status
Control Flow, Volatile Code	Medium	projects/contracts/OptionVault.sol (b24af21): 99	Resolved

Description

Function `setAMMAddress()` sets the address of AMM contract after the deployment of `OptionVault` contract, only allowed to be called once. It is an external function without any restriction on the role or address of the caller; and it is not called in contract `OptionVaultFactory`.

```
1 function setAMMAddress(address _ammAddress) external {
2     require(ammAddress == address(0));
3     ammAddress = _ammAddress;
4     setApprovalForAll(ammAddress, true);
5 }
```

Recommendation

We advise that the AMM contract address be set in constructor or factory contract; or the permission of `setAMMAddress()` be restricted to a role (e.g. operator). If the dev team choose the second solution, it is further recommended to also adopt proper decentralized mechanism or smart-contract-based accounts with enhanced security practices to mitigate centralization risk.

Alleviation

[Predy Team]: The team acknowledged the issue and addressed in the commit hash d294a392156023607ec1f7116234c59b24f5ab3f, the fixed make the variable `setAMMAddress` is called by AMMFactory.

OVC-02 | Incompatibility With Deflationary Tokens

Category	Severity	Location	Status
Volatile Code	Informational	projects/contracts/OptionVault.sol (b24af21): 1	Acknowledged

Description

When transferring standard ERC20 deflationary tokens, the input amount may not be equal to the received amount due to the charged transaction fee. For example, if a user deposits 100 deflationary tokens (with a 10% transaction fee) in a vault, only 90 tokens actually arrived in the contract. However, the user can still withdraw 100 tokens from the contract, which causes the contract to lose 10 tokens in such a transaction.

Recommendation

We advise the client to regulate the set of collateral and underlying tokens supported and add necessary mitigation mechanisms to keep track of accurate balances if there is a need to support deflationary tokens.

Alleviation

[Predy Team]: The team agreed that this is an acceptable risk, as this version of contracts is only compatible with WETH and USDC which are not deflationary.

OVK-01 | Correctness of Expiry and Series is not Guaranteed in `OptionVault`

Category	Severity	Location	Status
Control Flow	Informational	projects/contracts/OptionVault.sol (0564c2d): 526	Acknowledged

Description

In contract `OptionVault`, series is used as a form of identifier for options, and expiry indicates the expiration time. The overall functionality of the contract depends on their correctness. The valuess of expiry and series are directy created from input arguments through function `createExpiry()`.

The correctness of expiry and series relies on the proper governance, which is out of scope of this audit.

Alleviation

[Predy Team]: The team acknowledged the issue. The team is willing to take this as an acceptable risk. The operator will carefully create expiry and series in the current version.

OVK-02 | Centralization Risk in `OptionVault`

Category	Severity	Location	Status
Centralization / Privilege	**Major**	projects/contracts/OptionVault.sol (0564c2d): 553, 544, 526, 395	Acknowledged

Description

In the contract `OptionVault`, the role `operator` has the authority over the following function:

- `redeemCollateralFromLendingPool()`
- `createExpiry()`
- `setConfig()`
- `setNewOperator()`

Any compromise to the `operator` account may allow the hacker to take advantage of this and cause issues.

Recommendation

We advise the client to carefully manage the `operator` account's private key to avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol to be improved via a decentralized mechanism or smart-contract-based accounts with enhanced security practices, e.g., Multisignature wallets.

Indicatively, here is some feasible suggestions that would also mitigate the potential risk at the different level in term of short-term and long-term:

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key;
- Introduction of a DAO/governance/voting module to increase transparency and user involvement.

Alleviation

[Predy Team]: The team acknowledged the issue and adopted to use the multi-signature solution. The operator role will require multiple signers to co-sign the transaction before any funds enter the contract.

OVK-03 | Missing Error Message

Category	Severity	Location	Status
Coding Style	Informational	projects/contracts/OptionVault.sol (0564c2d): 53	Acknowledged

Description

In the current version of contract `OptionVault`, all the error messages in the **require** statements are replaced with code.

Recommendation

We recommend putting proper error messages in the **require** statements.

Alleviation

[Predy Team]: The team acknowledged the issue. A correspondence table between the error number and the error message will be provided where appropriate.

OVK-04 | Lack of Debts Repayment Guarantees

Category	Severity	Location	Status
Volatile Code	Medium	projects/contracts/OptionVault.sol (0564c2d): 431~435	Resolved

Description

In contracts OptionLib and OptionVault, the concept of debts is represented by variable `optionInfo.excessDebt`, which would be increased when calling function `settle()` of the contract OptionLib. Debts would be decreased when calling function `decreaseDebt()` of the contract OptionVault.

The debt of a vault can be repaid by anyone, and there are no time limits nor penalties on repaying. It seems currently there are no guarantees that a vault owner will repay the debt. Or, is it designed in such a way that there is no need to guarantee anyone's repayment? Unless they have no more collateral or have taken out all of their collateral?

Alleviation

[Predy Team]: The team agreed about the issue. First, the liquidation function works before the vault becomes insolvent. If the OptionVault contract still becomes insolvent, a portion of the FeePool is considered to be the LiquidationPool and the debt is paid from the FeePool. In the future, this process will be automated in the contract.

The team also decided that we don't need `decreaseDebt` and `excessDebt` to track the debt in the current version, so we decided to transfer the debt directly from LiquidationPool to OptionVault instead of using `decreaseDebt`.

The variables `decreaseDebt` and `excessDebt` are removed from the contract and its reflected in the commit hash ca86b089160688535d12b27b85f30c8d2ee87685.

OVK-05 | Missing Emit Events

Category	Severity	Location	Status
Coding Style	Informational	projects/contracts/OptionVault.sol (0564c2d): 544	Resolved

Description

The function that affects the status of sensitive variables should be able to emit events as notifications to traders and LP providers.

- `setConfig()` in contract `OptionVault`
- `setConfig()` in contract `AMM`
- `setDepositAllowedUntil()` in `AMM`
- `setLockupPeriod()` in `AMM`

Recommendation

Consider adding events for sensitive actions, and emit them in the function.

Alleviation

[Predy Team]: The team addressed the issue and reflected the change in the commit hash bcbfea1007d6ce67c1ae226121f1a2943ac2b184.

POC-01 | Third Party Dependencies on Chainlink Aggregator

Category	Severity	Location	Status
Volatile Code	Informational	projects/contracts/PriceOracle.sol (b24af21): 5	Acknowledged

Description

In contract `OptionVault` and `AMM`, spot price and expiry price are used to calculate payout, premium, delta, and many other important values. The value of spot price and expiry price are acquired from chainlink aggregator in contract `PriceOracle`, which is a 3rd party protocol. The scope of the audit treats 3rd party entities as black boxes and assume their functional correctness. However, in the real world, 3rd parties can be compromised and this may lead to lost or stolen assets. In addition, upgrades of 3rd parties can possibly lead to severe impacts.

Recommendation

We understand that the business logic of `PriceOracle` requires interaction with chainlink aggregator. We encourage the team to constantly monitor the statuses of 3rd parties to mitigate the side effects when unexpected activities are observed.

Alleviation

[Predy Team]: The team acknowledged the issue and reflected the chainlink's answer always greater than 0, in the commit hash b335c332bf34a5a91534db3477f4d443ef6683d2.

POK-01 | Centralization Risk in `PriceOracle` and `FeePool`

Category	Severity	Location	Status
Centralization / Privilege	**Major**	projects/contracts/PriceOracle.sol (0564c2d): 32	Acknowledged

Description

In the contract `PriceOracle`, the role `owner` has the authority over the following function:

- `setAggregator()`

In the contract `FeePool`, the role `owner` has the authority over the following function:

- `withdraw()`

Any compromise to the `owner` account(s) may allow the hacker to take advantage of this and cause issues.

Recommendation

We advise the client to carefully manage the `owner` account's private key to avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol to be improved via a decentralized mechanism or smart-contract-based accounts with enhanced security practices, e.g., Multisignature wallets.

Indicatively, here is some feasible suggestions that would also mitigate the potential risk at the different level in term of short-term and long-term:

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key;
- Introduction of a DAO/governance/voting module to increase transparency and user involvement.

Alleviation

[Predy Team]: The team acknowledged the issue and adopted to use the multi-signature solution. The operator role will require multiple signers to co-sign the transaction before any funds enter the contract.

Appendix

Finding Categories

Centralization / Privilege

Centralization / Privilege findings refer to either feature logic or implementation of components that act against the nature of decentralization, such as explicit ownership or specialized access roles in combination with a mechanism to relocate funds.

Gas Optimization

Gas Optimization findings do not affect the functionality of the code but generate different, more optimal EVM opcodes resulting in a reduction on the total gas cost of a transaction.

Mathematical Operations

Mathematical Operation findings relate to mishandling of math formulas, such as overflows, incorrect operations etc.

Logical Issue

Logical Issue findings detail a fault in the logic of the linked code, such as an incorrect notion on how block.timestamp works.

Control Flow

Control Flow findings concern the access control imposed on functions, such as owner-only functions being invoke-able by anyone under certain circumstances.

Volatile Code

Volatile Code findings refer to segments of code that behave unexpectedly on certain edge cases that may result in a vulnerability.

Language Specific

Language Specific findings are issues that would only arise within Solidity, i.e. incorrect usage of private or delete.

Coding Style

Coding Style findings usually do not affect the generated byte-code but rather comment on how to make the codebase more legible and, as a result, easily maintainable.

Inconsistency

Inconsistency findings refer to functions that should seemingly behave similarly yet contain different code, such as a constructor assignment imposing different require statements on the input variables than a setter function.

Checksum Calculation Method

The "Checksum" field in the "Audit Scope" section is calculated as the SHA-256 (Secure Hash Algorithm 2 with digest size of 256 bits) digest of the content of each file hosted in the listed source repository under the specified commit.

The result is hexadecimal encoded and is the same as the output of the Linux "sha256sum" command against the target file.

Disclaimer

This report is subject to the terms and conditions (including without limitation, description of services, confidentiality, disclaimer and limitation of liability) set forth in the Services Agreement, or the scope of services, and terms and conditions provided to you ("Customer" or the "Company") in connection with the Agreement. This report provided in connection with the Services set forth in the Agreement shall be used by the Company only to the extent permitted under the terms and conditions set forth in the Agreement. This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes, nor may copies be delivered to any other person other than the Company, without CertiK's prior written consent in each instance.

This report is not, nor should be considered, an "endorsement" or "disapproval" of any particular project or team. This report is not, nor should be considered, an indication of the economics or value of any "product" or "asset" created by any team or project that contracts CertiK to perform a security assessment. This report does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors, business, business model or legal compliance.

This report should not be used in any way to make decisions around investment or involvement with any particular project. This report in no way provides investment advice, nor should be leveraged as investment advice of any sort. This report represents an extensive assessing process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.

Blockchain technology and cryptographic assets present a high level of ongoing risk. CertiK's position is that each company and individual are responsible for their own due diligence and continuous security. CertiK's goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies, and in no way claims any guarantee of security or functionality of the technology we agree to analyze.

The assessment services provided by CertiK is subject to dependencies and under continuing development. You agree that your access and/or use, including but not limited to any services, reports, and materials, will be at your sole risk on an as-is, where-is, and as-available basis. Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty. The assessment reports could include false positives, false negatives, and other unpredictable results. The services may access, and depend upon, multiple layers of third-parties.

ALL SERVICES, THE LABELS, THE ASSESSMENT REPORT, WORK PRODUCT, OR OTHER MATERIALS, OR ANY PRODUCTS OR RESULTS OF THE USE THEREOF ARE PROVIDED "AS IS" AND "AS

AVAILABLE" AND WITH ALL FAULTS AND DEFECTS WITHOUT WARRANTY OF ANY KIND. TO THE MAXIMUM EXTENT PERMITTED UNDER APPLICABLE LAW, CERTIK HEREBY DISCLAIMS ALL WARRANTIES, WHETHER EXPRESS, IMPLIED, STATUTORY, OR OTHERWISE WITH RESPECT TO THE SERVICES, ASSESSMENT REPORT, OR OTHER MATERIALS. WITHOUT LIMITING THE FOREGOING, CERTIK SPECIFICALLY DISCLAIMS ALL IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT, AND ALL WARRANTIES ARISING FROM COURSE OF DEALING, USAGE, OR TRADE PRACTICE. WITHOUT LIMITING THE FOREGOING, CERTIK MAKES NO WARRANTY OF ANY KIND THAT THE SERVICES, THE LABELS, THE ASSESSMENT REPORT, WORK PRODUCT, OR OTHER MATERIALS, OR ANY PRODUCTS OR RESULTS OF THE USE THEREOF, WILL MEET CUSTOMER'S OR ANY OTHER PERSON'S REQUIREMENTS, ACHIEVE ANY INTENDED RESULT, BE COMPATIBLE OR WORK WITH ANY SOFTWARE, SYSTEM, OR OTHER SERVICES, OR BE SECURE, ACCURATE, COMPLETE, FREE OF HARMFUL CODE, OR ERROR-FREE. WITHOUT LIMITATION TO THE FOREGOING, CERTIK PROVIDES NO WARRANTY OR UNDERTAKING, AND MAKES NO REPRESENTATION OF ANY KIND THAT THE SERVICE WILL MEET CUSTOMER'S REQUIREMENTS, ACHIEVE ANY INTENDED RESULTS, BE COMPATIBLE OR WORK WITH ANY OTHER SOFTWARE, APPLICATIONS, SYSTEMS OR SERVICES, OPERATE WITHOUT INTERRUPTION, MEET ANY PERFORMANCE OR RELIABILITY STANDARDS OR BE ERROR FREE OR THAT ANY ERRORS OR DEFECTS CAN OR WILL BE CORRECTED.

WITHOUT LIMITING THE FOREGOING, NEITHER CERTIK NOR ANY OF CERTIK'S AGENTS MAKES ANY REPRESENTATION OR WARRANTY OF ANY KIND, EXPRESS OR IMPLIED AS TO THE ACCURACY, RELIABILITY, OR CURRENCY OF ANY INFORMATION OR CONTENT PROVIDED THROUGH THE SERVICE. CERTIK WILL ASSUME NO LIABILITY OR RESPONSIBILITY FOR (I) ANY ERRORS, MISTAKES, OR INACCURACIES OF CONTENT AND MATERIALS OR FOR ANY LOSS OR DAMAGE OF ANY KIND INCURRED AS A RESULT OF THE USE OF ANY CONTENT, OR (II) ANY PERSONAL INJURY OR PROPERTY DAMAGE, OF ANY NATURE WHATSOEVER, RESULTING FROM CUSTOMER'S ACCESS TO OR USE OF THE SERVICES, ASSESSMENT REPORT, OR OTHER MATERIALS.

ALL THIRD-PARTY MATERIALS ARE PROVIDED "AS IS" AND ANY REPRESENTATION OR WARRANTY OF OR CONCERNING ANY THIRD-PARTY MATERIALS IS STRICTLY BETWEEN CUSTOMER AND THE THIRD-PARTY OWNER OR DISTRIBUTOR OF THE THIRD-PARTY MATERIALS.

THE SERVICES, ASSESSMENT REPORT, AND ANY OTHER MATERIALS HEREUNDER ARE SOLELY PROVIDED TO CUSTOMER AND MAY NOT BE RELIED ON BY ANY OTHER PERSON OR FOR ANY PURPOSE NOT SPECIFICALLY IDENTIFIED IN THIS AGREEMENT, NOR MAY COPIES BE DELIVERED TO, ANY OTHER PERSON WITHOUT CERTIK'S PRIOR WRITTEN CONSENT IN EACH INSTANCE.

NO THIRD PARTY OR ANYONE ACTING ON BEHALF OF ANY THEREOF, SHALL BE A THIRD PARTY OR OTHER BENEFICIARY OF SUCH SERVICES, ASSESSMENT REPORT, AND ANY ACCOMPANYING

MATERIALS AND NO SUCH THIRD PARTY SHALL HAVE ANY RIGHTS OF CONTRIBUTION AGAINST CERTIK WITH RESPECT TO SUCH SERVICES, ASSESSMENT REPORT, AND ANY ACCOMPANYING MATERIALS.

THE REPRESENTATIONS AND WARRANTIES OF CERTIK CONTAINED IN THIS AGREEMENT ARE SOLELY FOR THE BENEFIT OF CUSTOMER. ACCORDINGLY, NO THIRD PARTY OR ANYONE ACTING ON BEHALF OF ANY THEREOF, SHALL BE A THIRD PARTY OR OTHER BENEFICIARY OF SUCH REPRESENTATIONS AND WARRANTIES AND NO SUCH THIRD PARTY SHALL HAVE ANY RIGHTS OF CONTRIBUTION AGAINST CERTIK WITH RESPECT TO SUCH REPRESENTATIONS OR WARRANTIES OR ANY MATTER SUBJECT TO OR RESULTING IN INDEMNIFICATION UNDER THIS AGREEMENT OR OTHERWISE.

FOR AVOIDANCE OF DOUBT, THE SERVICES, INCLUDING ANY ASSOCIATED ASSESSMENT REPORTS OR MATERIALS, SHALL NOT BE CONSIDERED OR RELIED UPON AS ANY FORM OF FINANCIAL, TAX, LEGAL, REGULATORY, OR OTHER ADVICE.

About

Founded in 2017 by leading academics in the field of Computer Science from both Yale and Columbia University, CertiK is a leading blockchain security company that serves to verify the security and correctness of smart contracts and blockchain-based protocols. Through the utilization of our world-class technical expertise, alongside our proprietary, innovative tech, we're able to support the success of our clients with best-in-class security, all whilst realizing our overarching vision; provable trust for all throughout all facets of blockchain.