SolidityScan | Powered by CRED SHIELDS

Security Assessment

amet-contracts

14 Nov 2023

This security assessment report was prepared by SolidityScan.com, a cloud-based Smart Contract Scanner.

Self-published

This audit report was Self-published by the user. To learn more about our published reports click here.

Table of Contents.

- VARIABLES SHOULD BE IMMUTABLE

Scan History

Disclaimer

Project Summary

This report has been prepared for amet-contracts using SolidityScan to scan and discover vulnerabilities and safe coding practices in their smart contract including the libraries used by the contract that are not officially recognized. The SolidityScan tool runs a comprehensive static analysis on the Solidity code and finds vulnerabilities ranging from minor gas optimizations to major vulnerabilities leading to the loss of funds. The coverage scope pays attention to all the informational and critical vulnerabilities with over (140+) modules. The scanning and auditing process covers the following areas:

Various common and uncommon attack vectors will be investigated to ensure that the smart contracts are secure from malicious actors. The scanner modules find and flag issues related to Gas optimizations that help in reducing the overall Gas cost It scans and evaluates the codebase against industry best practices and standards to ensure compliance It makes sure that the officially recognized libraries used in the code are secure and up to date

The SolidityScan Team recommends running regular audit scans to identify any vulnerabilities that are introduced after amet-contracts introduces new features or refactors the code.

Audit Summary

Project Name

amet-contracts

Contract Type

Smart Contract

Language

Solidity

Codebase

https://github.com/Amet-Finance/contracts

Date Published

14 Nov 2023

Organization

Amet Finance

Publishers/Owners Name

The Unconstrained Mind

Audit Methodology

Static Scanning

Findings Summary

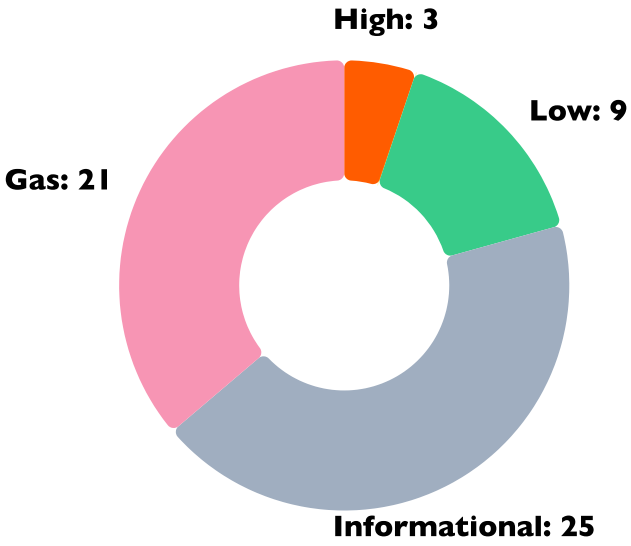
amet-contracts

https://github.com/Amet-Finance/contracts

Lines of Code **334**

100.00

Security Score



Critical	0	Low	9
High	3	Informational	25
Medium	0	Gas	21

ACTION TAKEN

Fixed	False Positive
75	29
Won't Fix	**Pending Fix**
0	0

Bug ID	Severity	Bug Type	Status
SSP_3497_82	Informational	BLOCK VALUES AS A PROXY FOR TIME	*False Positive*
SSP_3497_39	Informational	BLOCK VALUES AS A PROXY FOR TIME	*False Positive*
SSP_3497_83	Informational	BLOCK VALUES AS A PROXY FOR TIME	*False Positive*
SSP_3497_38	Informational	BLOCK VALUES AS A PROXY FOR TIME	*Fixed*
SSP_3497_40	Informational	BLOCK VALUES AS A PROXY FOR TIME	*Fixed*
SSP_3497_10	Informational	BOOLEAN EQUALITY	*Fixed*

SSP_3497_72	Gas	CHEAPER INEQUALITIES IN IF()	*False Positive*
SSP_3497_85	Gas	CHEAPER INEQUALITIES IN IF()	*False Positive*
SSP_3497_101	Gas	CHEAPER INEQUALITIES IN IF()	*False Positive*
SSP_3497_47	Gas	CHEAPER INEQUALITIES IN IF()	*False Positive*
SSP_3497_87	Gas	CHEAPER INEQUALITIES IN IF()	*Fixed*
SSP_3497_86	Gas	CHEAPER INEQUALITIES IN IF()	*Fixed*
SSP_3497_71	Gas	CHEAPER INEQUALITIES IN IF()	*Fixed*
SSP_3497_45	Gas	CHEAPER INEQUALITIES IN IF()	*Fixed*
SSP_3497_46	Gas	CHEAPER INEQUALITIES IN IF()	*Fixed*
SSP_3497_41	Gas	CHEAPER INEQUALITIES IN IF()	*Fixed*
SSP_3497_42	Gas	CHEAPER INEQUALITIES IN IF()	*Fixed*
SSP_3497_43	Gas	CHEAPER INEQUALITIES IN IF()	*Fixed*

SSP_3497_44	Gas	CHEAPER INEQUALITIES IN IF()	*Fixed*
SSP_3497_69	Low	COMPILER VERSION TOO RECENT	*False Positive*
SSP_3497_68	Low	COMPILER VERSION TOO RECENT	*False Positive*
SSP_3497_6	Gas	DEFINE CONSTRUCTOR AS PAYABLE	*False Positive*
SSP_3497_76	Gas	DEFINE CONSTRUCTOR AS PAYABLE	*False Positive*
SSP_3497_7	Gas	DEFINE CONSTRUCTOR AS PAYABLE	*Fixed*
SSP_3497_102	Low	EVENT BASED REENTRANCY	*False Positive*
SSP_3497_97	Low	EVENT BASED REENTRANCY	*Fixed*
SSP_3497_32	Low	USE OF FLOATING PRAGMA	*Fixed*
SSP_3497_33	Low	USE OF FLOATING PRAGMA	*Fixed*
SSP_3497_88	Gas	FUNCTION SHOULD RETURN STRUCT	*False Positive*
SSP_3497_31	Gas	FUNCTION SHOULD RETURN STRUCT	*Fixed*

SSP_3497_8	High	UNCHECKED ARRAY LENGTH	*False Positive*
SSP_3497_67	Critical	INCORRECT ACCESS CONTROL	*False Positive*
SSP_3497_13	Critical	INCORRECT ACCESS CONTROL	*Fixed*
SSP_3497_70	Low	MISSING EVENTS	*False Positive*
SSP_3497_99	Low	MISSING EVENTS	*False Positive*
SSP_3497_81	Low	MISSING EVENTS	*False Positive*
SSP_3497_80	Low	MISSING EVENTS	*Fixed*
SSP_3497_26	Low	MISSING EVENTS	*Fixed*
SSP_3497_27	Low	MISSING EVENTS	*Fixed*
SSP_3497_79	Low	MISSING EVENTS	*Fixed*
SSP_3497_28	Low	MISSING EVENTS	*Fixed*
SSP_3497_29	Low	MISSING EVENTS	*Fixed*

SSP_3497_30	Low	MISSING EVENTS	*Fixed*
SSP_3497_25	Low	MISSING EVENTS	*Fixed*
SSP_3497_104	Informational	MISSING INDEXED KEYWORDS IN EVENTS	*False Positive*
SSP_3497_34	Informational	MISSING INDEXED KEYWORDS IN EVENTS	*Fixed*
SSP_3497_35	Informational	MISSING INDEXED KEYWORDS IN EVENTS	*Fixed*
SSP_3497_36	Informational	MISSING INDEXED KEYWORDS IN EVENTS	*Fixed*
SSP_3497_37	Informational	MISSING INDEXED KEYWORDS IN EVENTS	*Fixed*
SSP_3497_59	Informational	MISSING PAYABLE IN CALL FUNCTION	*False Positive*
SSP_3497_14	Informational	MISSING UNDERSCORE IN NAMING VARIABLES	*Fixed*
SSP_3497_15	Informational	MISSING UNDERSCORE IN NAMING VARIABLES	*Fixed*
SSP_3497_16	Informational	MISSING UNDERSCORE IN NAMING VARIABLES	*Fixed*
SSP_3497_17	Informational	MISSING UNDERSCORE IN NAMING VARIABLES	*Fixed*

SSP_3497_18	Informational	MISSING UNDERSCORE IN NAMING VARIABLES	*Fixed*
SSP_3497_61	Informational	MISSING UNDERSCORE IN NAMING VARIABLES	*Fixed*
SSP_3497_62	Informational	MISSING UNDERSCORE IN NAMING VARIABLES	*Fixed*
SSP_3497_63	Informational	MISSING UNDERSCORE IN NAMING VARIABLES	*Fixed*
SSP_3497_64	Informational	MISSING UNDERSCORE IN NAMING VARIABLES	*Fixed*
SSP_3497_65	Informational	MISSING UNDERSCORE IN NAMING VARIABLES	*Fixed*
SSP_3497_66	Informational	MISSING UNDERSCORE IN NAMING VARIABLES	*Fixed*
SSP_3497_19	Informational	MISSING UNDERSCORE IN NAMING VARIABLES	*Fixed*
SSP_3497_20	Informational	MISSING UNDERSCORE IN NAMING VARIABLES	*Fixed*
SSP_3497_21	Informational	MISSING UNDERSCORE IN NAMING VARIABLES	*Fixed*
SSP_3497_22	Informational	MISSING UNDERSCORE IN NAMING VARIABLES	*Fixed*
SSP_3497_23	Informational	MISSING UNDERSCORE IN NAMING VARIABLES	*Fixed*

SSP_3497_24	Informational	MISSING UNDERSCORE IN NAMING VARIABLES	*Fixed*
SSP_3497_100	High	REENTRANCY	*False Positive*
SSP_3497_78	High	REENTRANCY	*False Positive*
SSP_3497_96	High	REENTRANCY	*Fixed*
SSP_3497_77	High	REENTRANCY	*Fixed*
SSP_3497_11	High	REENTRANCY	*Fixed*
SSP_3497_12	High	REENTRANCY	*Fixed*
SSP_3497_89	Gas	STORAGE VARIABLE CACHING IN MEMORY	*False Positive*
SSP_3497_90	Gas	STORAGE VARIABLE CACHING IN MEMORY	*False Positive*
SSP_3497_91	Gas	STORAGE VARIABLE CACHING IN MEMORY	*False Positive*
SSP_3497_91	Gas	STORAGE VARIABLE CACHING IN MEMORY	*False Positive*
SSP_3497_93	Gas	STORAGE VARIABLE CACHING IN MEMORY	*False Positive*

SSP_3497_94	Gas	STORAGE VARIABLE CACHING IN MEMORY	*False Positive*
SSP_3497_58	Gas	STORAGE VARIABLE CACHING IN MEMORY	*False Positive*
SSP_3497_49	Gas	STORAGE VARIABLE CACHING IN MEMORY	*Fixed*
SSP_3497_50	Gas	STORAGE VARIABLE CACHING IN MEMORY	*Fixed*
SSP_3497_51	Gas	STORAGE VARIABLE CACHING IN MEMORY	*Fixed*
SSP_3497_92	Gas	STORAGE VARIABLE CACHING IN MEMORY	*Fixed*
SSP_3497_52	Gas	STORAGE VARIABLE CACHING IN MEMORY	*Fixed*
SSP_3497_73	Gas	STORAGE VARIABLE CACHING IN MEMORY	*Fixed*
SSP_3497_54	Gas	STORAGE VARIABLE CACHING IN MEMORY	*Fixed*
SSP_3497_54	Gas	STORAGE VARIABLE CACHING IN MEMORY	*Fixed*
SSP_3497_74	Gas	STORAGE VARIABLE CACHING IN MEMORY	*Fixed*
SSP_3497_56	Gas	STORAGE VARIABLE CACHING IN MEMORY	*Fixed*

SSP_3497_57	Gas	STORAGE VARIABLE CACHING IN MEMORY	*Fixed*
SSP_3497_53	Gas	STORAGE VARIABLE CACHING IN MEMORY	*Fixed*
SSP_3497_55	Gas	STORAGE VARIABLE CACHING IN MEMORY	*Fixed*
SSP_3497_84	Informational	UNUSED RECEIVE FALLBACK	*Fixed*
SSP_3497_48	Informational	USE CALL INSTEAD OF TRANSFER OR SEND	*Fixed*
SSP_3497_103	Low	USE OWNABLE2STEP	*False Positive*
SSP_3497_98	Low	USE OWNABLE2STEP	*Fixed*
SSP_3497_95	Low	USE OWNABLE2STEP	*Fixed*
SSP_3497_75	Low	USE OWNABLE2STEP	*Fixed*
SSP_3497_60	Low	USE OWNABLE2STEP	*Fixed*
SSP_3497_9	Low	USE OWNABLE2STEP	*Fixed*
SSP_3497_1	Informational	VARIABLES SHOULD BE IMMUTABLE	*Fixed*

SSP_3497_2	Informational	VARIABLES SHOULD BE IMMUTABLE	*Fixed*
SSP_3497_3	Informational	VARIABLES SHOULD BE IMMUTABLE	*Fixed*
SSP_3497_4	Informational	VARIABLES SHOULD BE IMMUTABLE	*Fixed*
SSP_3497_5	Informational	VARIABLES SHOULD BE IMMUTABLE	*Fixed*

Vulnerability Details

Bug ID

SSP_3497_82

Severity

Informational

Confidence

Firm

Line nos

86-86

Action Taken

False Positive

Bug Type

BLOCK VALUES AS A PROXY FOR TIME

File Location

/contracts/zcb-v2/ZeroCouponBondsV1_AmetFinance.sol

Issue Description

Contracts often need access to time values to perform certain types of functionality. Values such as `block.timestamp` and `block.number` can be used to determine the current time or the time delta. However, they are not recommended for most use cases.

For `block.number`, as Ethereum block times are generally around 14 seconds, the delta between blocks can be predicted. The block times, on the other hand, do not remain constant and are subject to change for a number of reasons, e.g., fork reorganizations and the difficulty bomb.

Due to variable block times, `block.number` should not be relied on for precise calculations of time.

Issue Remediation

It is recommended to use trusted external time sources, block numbers instead of timestamps, and/or utilizing multiple time sources to increase reliability. These practices can help mitigate risks of timestamp manipulation and inaccurate timing, increasing the reliability and security of the smart contract.

Bug ID

SSP_3497_39

Severity

Informational

Confidence

Firm

Line nos

157-157

Action Taken

False Positive

Bug Type

BLOCK VALUES AS A PROXY FOR TIME

File Location

/contracts/zcb-v2/ZeroCouponBondsV1_AmetFinance.sol

Issue Description

Contracts often need access to time values to perform certain types of functionality. Values such as `block.timestamp` and `block.number` can be used to determine the current time or the time delta. However, they are not recommended for most use cases.

For `block.number`, as Ethereum block times are generally around 14 seconds, the delta between blocks can be predicted. The block times, on the other hand, do not remain constant and are subject to change for a number of reasons, e.g., fork reorganizations and the difficulty bomb.

Due to variable block times, `block.number` should not be relied on for precise calculations of time.

Issue Remediation

It is recommended to use trusted external time sources, block numbers instead of timestamps, and/or utilizing multiple time sources to increase reliability. These practices can help mitigate risks of timestamp manipulation and inaccurate timing, increasing the reliability and security of the smart contract.

Bug ID

SSP_3497_83

Severity

Informational

Confidence

Firm

Line nos

172-172

Action Taken

False Positive

Bug Type

BLOCK VALUES AS A PROXY FOR TIME

File Location

/contracts/zcb-v2/ZeroCouponBondsV1_AmetFinance.sol

Issue Description

Contracts often need access to time values to perform certain types of functionality. Values such as `block.timestamp` and `block.number` can be used to determine the current time or the time delta. However, they are not recommended for most use cases.

For `block.number`, as Ethereum block times are generally around 14 seconds, the delta between blocks can be predicted. The block times, on the other hand, do not remain constant and are subject to change for a number of reasons, e.g., fork reorganizations and the difficulty bomb.

Due to variable block times, `block.number` should not be relied on for precise calculations of time.

Issue Remediation

It is recommended to use trusted external time sources, block numbers instead of timestamps, and/or utilizing multiple time sources to increase reliability. These practices can help mitigate risks of timestamp manipulation and inaccurate timing, increasing the reliability and security of the smart contract.

Bug ID

SSP_3497_38

Severity

Informational

Confidence

Firm

Line nos

83-83

Action Taken

Fixed

Bug Type

BLOCK VALUES AS A PROXY FOR TIME

File Location

/contracts/zcb-v2/ZeroCouponBondsV1_AmetFinance.sol

Issue Description

Contracts often need access to time values to perform certain types of functionality. Values such as `block.timestamp` and `block.number` can be used to determine the current time or the time delta. However, they are not recommended for most use cases.

For `block.number`, as Ethereum block times are generally around 14 seconds, the delta between blocks can be predicted. The block times, on the other hand, do not remain constant and are subject to change for a number of reasons, e.g., fork reorganizations and the difficulty bomb.

Due to variable block times, `block.number` should not be relied on for precise calculations of time.

Issue Remediation

It is recommended to use trusted external time sources, block numbers instead of timestamps, and/or utilizing multiple time sources to increase reliability. These practices can help mitigate risks of timestamp manipulation and inaccurate timing, increasing the reliability and security of the smart contract.

Bug ID

SSP_3497_40

Severity

Informational

Confidence

Firm

Line nos

163-163

Action Taken

Fixed

Bug Type

BLOCK VALUES AS A PROXY FOR TIME

File Location

/contracts/zcb-v2/ZeroCouponBondsV1_AmetFinance.sol

Issue Description

Contracts often need access to time values to perform certain types of functionality. Values such as `block.timestamp` and `block.number` can be used to determine the current time or the time delta. However, they are not recommended for most use cases.

For `block.number`, as Ethereum block times are generally around 14 seconds, the delta between blocks can be predicted. The block times, on the other hand, do not remain constant and are subject to change for a number of reasons, e.g., fork reorganizations and the difficulty bomb.

Due to variable block times, `block.number` should not be relied on for precise calculations of time.

Issue Remediation

It is recommended to use trusted external time sources, block numbers instead of timestamps, and/or utilizing multiple time sources to increase reliability. These practices can help mitigate risks of timestamp manipulation and inaccurate timing, increasing the reliability and security of the smart contract.

Bug ID

SSP_3497_10

Severity

- **Informational**

Confidence

Certain

Line nos

39-39

Action Taken

Fixed

Bug Type

BOOLEAN EQUALITY

File Location

/contracts/zcb-v2/ZeroCouponBondsIssuerV1_AmetFinance.sol

Issue Description

In Solidity, and many other languages, boolean constants can be used directly in conditionals like `if` and `else` statements.
The contract was found to be equating constants in conditionals which is unnecessary.

Issue Remediation

It is recommended to directly use boolean constants. It is not required to equate them to `true` or `false.`

Bug ID

SSP_3497_72

Severity

Gas

Confidence

Firm

Line nos

39-39

Action Taken

False Positive

Bug Type

CHEAPER INEQUALITIES IN IF()

File Location

/contracts/zcb-v2/ZeroCouponBondsIssuerV1_AmetFinance.sol

Issue Description

The contract was found to be doing comparisons using inequalities inside the if statement.
When inside the `if` statements, non-strict inequalities `(>=, <=)` are usually cheaper than the strict equalities `(>, <).`

Issue Remediation

It is recommended to go through the code logic, and, if possible, modify the strict inequalities with the non-strict ones to save `~3` gas as long as the logic of the code is not affected.

Bug ID

SSP_3497_85

Severity

Gas

Confidence

Firm

Line nos

128-128

Action Taken

False Positive

Bug Type

CHEAPER INEQUALITIES IN IF()

File Location

/contracts/zcb-v2/ZeroCouponBondsV1_AmetFinance.sol

Issue Description

The contract was found to be doing comparisons using inequalities inside the if statement.
When inside the `if` statements, non-strict inequalities `(>=, <=)` are usually cheaper than the strict equalities `(>, <).`

Issue Remediation

It is recommended to go through the code logic, and, if possible, modify the strict inequalities with the non-strict ones to save `~3` gas as long as the logic of the code is not affected.

Bug ID

SSP_3497_101

Severity

Gas

Confidence

Firm

Line nos

151-151

Action Taken

False Positive

Bug Type

CHEAPER INEQUALITIES IN IF()

File Location

/contracts/zcb-v2/ZeroCouponBondsV1_AmetFinance.sol

Issue Description

The contract was found to be doing comparisons using inequalities inside the if statement.
When inside the `if` statements, non-strict inequalities `(>=, <=)` are usually cheaper than the strict equalities `(>, <).`

Issue Remediation

It is recommended to go through the code logic, and, if possible, modify the strict inequalities with the non-strict ones to save `~3` gas as long as the logic of the code is not affected.

Bug ID

SSP_3497_47

Severity

Gas

Confidence

Firm

Line nos

177-177

Action Taken

False Positive

Bug Type

CHEAPER INEQUALITIES IN IF()

File Location

/contracts/zcb-v2/ZeroCouponBondsV1_AmetFinance.sol

Issue Description

The contract was found to be doing comparisons using inequalities inside the if statement.
When inside the `if` statements, non-strict inequalities `(>=, <=)` are usually cheaper than the strict equalities `(>, <).`

Issue Remediation

It is recommended to go through the code logic, and, if possible, modify the strict inequalities with the non-strict ones to save `~3` gas as long as the logic of the code is not affected.

Bug ID

SSP_3497_87

Severity

Gas

Confidence

Firm

Line nos

148-148

Action Taken

Fixed

Bug Type

CHEAPER INEQUALITIES IN IF()

File Location

/contracts/zcb-v2/ZeroCouponBondsV1_AmetFinance.sol

Issue Description

The contract was found to be doing comparisons using inequalities inside the if statement.
When inside the `if` statements, non-strict inequalities `(>=, <=)` are usually cheaper than the strict equalities `(>, <).`

Issue Remediation

It is recommended to go through the code logic, and, if possible, modify the strict inequalities with the non-strict ones to save `~3` gas as long as the logic of the code is not affected.

Bug ID

SSP_3497_86

Severity

Gas

Confidence

Firm

Line nos

134-134

Action Taken

Fixed

Bug Type

CHEAPER INEQUALITIES IN IF()

File Location

/contracts/zcb-v2/ZeroCouponBondsV1_AmetFinance.sol

Issue Description

The contract was found to be doing comparisons using inequalities inside the if statement.
When inside the `if` statements, non-strict inequalities `(>=, <=)` are usually cheaper than the strict equalities `(>, <).`

Issue Remediation

It is recommended to go through the code logic, and, if possible, modify the strict inequalities with the non-strict ones to save `~3` gas as long as the logic of the code is not affected.

Bug ID

SSP_3497_71

Severity

Gas

Confidence

Firm

Line nos

124-124

Action Taken

Fixed

Bug Type

CHEAPER INEQUALITIES IN IF()

File Location

/contracts/zcb-v2/ZeroCouponBondsV1_AmetFinance.sol

Issue Description

The contract was found to be doing comparisons using inequalities inside the if statement.
When inside the `if` statements, non-strict inequalities `(>=, <=)` are usually cheaper than the strict equalities `(>, <).`

Issue Remediation

It is recommended to go through the code logic, and, if possible, modify the strict inequalities with the non-strict ones to save `~3` gas as long as the logic of the code is not affected.

Bug ID

SSP_3497_45

Severity

Gas

Confidence

Firm

Line nos

133-133

Action Taken

Fixed

Bug Type

CHEAPER INEQUALITIES IN IF()

File Location

/contracts/zcb-v2/ZeroCouponBondsV1_AmetFinance.sol

Issue Description

The contract was found to be doing comparisons using inequalities inside the if statement.
When inside the `if` statements, non-strict inequalities `(>=, <=)` are usually cheaper than the strict equalities `(>, <).`

Issue Remediation

It is recommended to go through the code logic, and, if possible, modify the strict inequalities with the non-strict ones to save `~3` gas as long as the logic of the code is not affected.

Bug ID

SSP_3497_46

Severity

Gas

Confidence

Firm

Line nos

142-142

Action Taken

Fixed

Bug Type

CHEAPER INEQUALITIES IN IF()

File Location

/contracts/zcb-v2/ZeroCouponBondsV1_AmetFinance.sol

Issue Description

The contract was found to be doing comparisons using inequalities inside the if statement.
When inside the `if` statements, non-strict inequalities `(>=, <=)` are usually cheaper than the strict equalities `(>, <).`

Issue Remediation

It is recommended to go through the code logic, and, if possible, modify the strict inequalities with the non-strict ones to save `~3` gas as long as the logic of the code is not affected.

Bug ID

SSP_3497_41

Severity

Gas

Confidence

Firm

Line nos

38-38

Action Taken

Fixed

Bug Type

CHEAPER INEQUALITIES IN IF()

File Location

/contracts/zcb-v2/ZeroCouponBondsIssuerV1_AmetFinance.sol

Issue Description

The contract was found to be doing comparisons using inequalities inside the if statement.
When inside the `if` statements, non-strict inequalities `(>=, <=)` are usually cheaper than the strict equalities `(>, <).`

Issue Remediation

It is recommended to go through the code logic, and, if possible, modify the strict inequalities with the non-strict ones to save `~3` gas as long as the logic of the code is not affected.

Bug ID

SSP_3497_42

Severity

Gas

Confidence

Firm

Line nos

94-94

Action Taken

Fixed

Bug Type

CHEAPER INEQUALITIES IN IF()

File Location

/contracts/zcb-v2/ZeroCouponBondsV1_AmetFinance.sol

Issue Description

The contract was found to be doing comparisons using inequalities inside the if statement.
When inside the `if` statements, non-strict inequalities `(>=, <=)` are usually cheaper than the strict equalities `(>, <).`

Issue Remediation

It is recommended to go through the code logic, and, if possible, modify the strict inequalities with the non-strict ones to save `~3` gas as long as the logic of the code is not affected.

Bug ID

SSP_3497_43

Severity

Gas

Confidence

Firm

Line nos

107-107

Action Taken

Fixed

Bug Type

CHEAPER INEQUALITIES IN IF()

File Location

/contracts/zcb-v2/ZeroCouponBondsV1_AmetFinance.sol

Issue Description

The contract was found to be doing comparisons using inequalities inside the if statement.
When inside the `if` statements, non-strict inequalities `(>=, <=)` are usually cheaper than the strict equalities `(>, <).`

Issue Remediation

It is recommended to go through the code logic, and, if possible, modify the strict inequalities with the non-strict ones to save `~3` gas as long as the logic of the code is not affected.

Bug ID

SSP_3497_44

Severity

Gas

Confidence

Firm

Line nos

123-123

Action Taken

Fixed

Bug Type

CHEAPER INEQUALITIES IN IF()

File Location

/contracts/zcb-v2/ZeroCouponBondsV1_AmetFinance.sol

Issue Description

The contract was found to be doing comparisons using inequalities inside the if statement.
When inside the `if` statements, non-strict inequalities `(>=, <=)` are usually cheaper than the strict equalities `(>, <).`

Issue Remediation

It is recommended to go through the code logic, and, if possible, modify the strict inequalities with the non-strict ones to save `~3` gas as long as the logic of the code is not affected.

Bug ID

SSP_3497_69

Severity

Low

Confidence

Certain

Line nos

2-2

Action Taken

False Positive

Bug Type

COMPILER VERSION TOO RECENT

File Location

/contracts/zcb-v2/ZeroCouponBondsIssuerV1_AmetFinance.sol

Issue Description

The compiler version detected in the code is too recent. Therefore, it is not time-tested and may be susceptible to multiple bugs and vulnerabilities, both from the usage and security perspectives. The following compiler versions were detected which were too recent - ['/contracts/zcb-v2/ZeroCouponBondsIssuerV1_AmetFinance.sol'] - 0.8.22

Issue Remediation

It is suggested to use a compiler version that is neither too recent nor too old i.e., Solidity `0.8.18`. A stable compiler version should be used that is time-tested by the community, which fixed vulnerabilities introduced in older compiler versions.

The code should be kept updated according to the compiler release cycle. It should be tested before going on the Mainnet to reduce the chances of new vulnerabilities being introduced.

Bug ID

SSP_3497_68

Severity

Low

Confidence

Certain

Line nos

2-2

Action Taken

False Positive

Bug Type

COMPILER VERSION TOO RECENT

File Location

/contracts/zcb-v2/ZeroCouponBondsV1_AmetFinance.sol

Issue Description

The compiler version detected in the code is too recent. Therefore, it is not time-tested and may be susceptible to multiple bugs and vulnerabilities, both from the usage and security perspectives. The following compiler versions were detected which were too recent - ['/contracts/zcb-v2/ZeroCouponBondsV1_AmetFinance.sol'] - 0.8.22

Issue Remediation

It is suggested to use a compiler version that is neither too recent nor too old i.e., Solidity `0.8.18`. A stable compiler version should be used that is time-tested by the community, which fixed vulnerabilities introduced in older compiler versions.

The code should be kept updated according to the compiler release cycle. It should be tested before going on the Mainnet to reduce the chances of new vulnerabilities being introduced.

Bug ID

SSP_3497_6

Severity

Gas

Confidence

Tentative

Line nos

25-28

Action Taken

False Positive

Bug Type

DEFINE CONSTRUCTOR AS PAYABLE

File Location

/contracts/zcb-v2/ZeroCouponBondsIssuerV1_AmetFinance.sol

Issue Description

Developers can save around 10 opcodes and some gas if the constructors are defined as payable.
However, it should be noted that it comes with risks because payable constructors can accept ETH during deployment.

Issue Remediation

It is suggested to mark the constructors as payable to save some gas. Make sure it does not lead to any adverse effects in case an upgrade pattern is involved.

Bug ID

SSP_3497_76

Severity

Gas

Confidence

Tentative

Line nos

62-87

Action Taken

False Positive

Bug Type

DEFINE CONSTRUCTOR AS PAYABLE

File Location

/contracts/zcb-v2/ZeroCouponBondsV1_AmetFinance.sol

Issue Description

Developers can save around 10 opcodes and some gas if the constructors are defined as payable.
However, it should be noted that it comes with risks because payable constructors can accept ETH during deployment.

Issue Remediation

It is suggested to mark the constructors as payable to save some gas. Make sure it does not lead to any adverse effects in case an upgrade pattern is involved.

Bug ID

SSP_3497_7

Severity

Gas

Confidence

Tentative

Line nos

60-84

Action Taken

Fixed

Bug Type

DEFINE CONSTRUCTOR AS PAYABLE

File Location

/contracts/zcb-v2/ZeroCouponBondsV1_AmetFinance.sol

Issue Description

Developers can save around 10 opcodes and some gas if the constructors are defined as payable.
However, it should be noted that it comes with risks because payable constructors can accept ETH during deployment.

Issue Remediation

It is suggested to mark the constructors as payable to save some gas. Make sure it does not lead to any adverse effects in case an upgrade pattern is involved.

Bug ID

SSP_3497_102

Severity

Low

Confidence

Firm

Line nos

134-143

Action Taken

False Positive

Bug Type

EVENT BASED REENTRANCY

File Location

/contracts/zcb-v2/ZeroCouponBondsV1_AmetFinance.sol

Issue Description

In a Re-entrancy attack, a malicious contract calls back into the calling contract before the first invocation of the function is finished. This may cause the different invocations of the function to interact in undesirable ways, especially in cases where the function is updating state variables after the external calls.
In the case of event-based Re-entrancy attacks, events are emitted after an external call leading to missing event calls.

Issue Remediation

It is recommended to add a [Re-entrancy Guard] to the functions making external calls. The functions should use a Checks-Effects-Interactions pattern. The external calls should be executed at the end of the function and all the state-changing and event emits must happen before the call.

Bug ID

SSP_3497_97

Severity

Low

Confidence

Firm

Line nos

134-142

Action Taken

Fixed

Bug Type

EVENT BASED REENTRANCY

File Location

/contracts/zcb-v2/ZeroCouponBondsV1_AmetFinance.sol

Issue Description

In a Re-entrancy attack, a malicious contract calls back into the calling contract before the first invocation of the function is finished. This may cause the different invocations of the function to interact in undesirable ways, especially in cases where the function is updating state variables after the external calls.
In the case of event-based Re-entrancy attacks, events are emitted after an external call leading to missing event calls.

Issue Remediation

It is recommended to add a [Re-entrancy Guard] to the functions making external calls. The functions should use a Checks-Effects-Interactions pattern. The external calls should be executed at the end of the function and all the state-changing and event emits must happen before the call.

Bug ID

SSP_3497_32

Severity

Low

Confidence

Certain

Line nos

2-2

Action Taken

Fixed

Bug Type

USE OF FLOATING PRAGMA

File Location

/contracts/zcb-v2/ZeroCouponBondsIssuerV1_AmetFinance.sol

Issue Description

Solidity source files indicate the versions of the compiler they can be compiled with using a pragma directive at the top of the solidity file. This can either be a floating pragma or a specific compiler version.
The contract was found to be using a floating pragma which is not considered safe as it can be compiled with all the versions described.
The following affected files were found to be using floating pragma:
`['/contracts/zcb-v2/ZeroCouponBondsIssuerV1_AmetFinance.sol'] - ^0.8.18`

Issue Remediation

It is recommended to use a fixed pragma version, as future compiler versions may handle certain language constructions in a way the developer did not foresee.
Using a floating pragma may introduce several vulnerabilities if compiled with an older version.
The developers should always use the exact Solidity compiler version when designing their contracts as it may break the changes in the future.
Instead of `^0.8.18` use `pragma solidity 0.8.18`, which is a stable and recommended version right now.

Bug ID

SSP_3497_33

Severity

Low

Confidence

Certain

Line nos

2-2

Action Taken

Fixed

Bug Type

USE OF FLOATING PRAGMA

File Location

/contracts/zcb-v2/ZeroCouponBondsV1_AmetFinance.sol

Issue Description

Solidity source files indicate the versions of the compiler they can be compiled with using a pragma directive at the top of the solidity file. This can either be a floating pragma or a specific compiler version.
The contract was found to be using a floating pragma which is not considered safe as it can be compiled with all the versions described.
The following affected files were found to be using floating pragma:
`['/contracts/zcb-v2/ZeroCouponBondsV1_AmetFinance.sol'] - ^0.8.18`

Issue Remediation

It is recommended to use a fixed pragma version, as future compiler versions may handle certain language constructions in a way the developer did not foresee.
Using a floating pragma may introduce several vulnerabilities if compiled with an older version.
The developers should always use the exact Solidity compiler version when designing their contracts as it may break the changes in the future.
Instead of `^0.8.18` use `pragma solidity 0.8.18`, which is a stable and recommended version right now.

Bug ID

SSP_3497_88

Severity

Gas

Confidence

Tentative

Line nos

199-229

Action Taken

False Positive

Bug Type

FUNCTION SHOULD RETURN STRUCT

File Location

/contracts/zcb-v2/ZeroCouponBondsV1_AmetFinance.sol

Issue Description

The function getInfo was detected to be returning multiple values.
Consider using a `struct` instead of multiple return values for the function. It can improve code readability.

Issue Remediation

Use `struct` for returning multiple values inside a function, which returns several parameters and improves code readability.

Bug ID

SSP_3497_31

Severity

Gas

Confidence

Tentative

Line nos

190-220

Action Taken

Fixed

Bug Type

FUNCTION SHOULD RETURN STRUCT

File Location

/contracts/zcb-v2/ZeroCouponBondsV1_AmetFinance.sol

Issue Description

The function getInfo was detected to be returning multiple values.
Consider using a `struct` instead of multiple return values for the function. It can improve code readability.

Issue Remediation

Use `struct` for returning multiple values inside a function, which returns several parameters and improves code readability.

Bug ID

SSP_3497_8

Severity

High

Confidence

Tentative

Line nos

179-179

Action Taken

False Positive

Bug Type

UNCHECKED ARRAY LENGTH

File Location

/contracts/zcb-v2/ZeroCouponBondsV1_AmetFinance.sol

Issue Description

Ethereum is a very resource-constrained environment. Prices per computational step are orders of magnitude higher than with centralized providers. Moreover, Ethereum miners impose a limit on the total number of Gas consumed in a block. If `array.length` is large enough, the function exceeds the block gas limit, and transactions calling it will never be confirmed.

```
for (uint256 i = 0; i < array.length ; i++) { cosltyFunc(); }
```

This becomes a security issue if an external actor influences `array.length`. E.g., if an array enumerates all registered addresses, an adversary can register many addresses, causing the problem described above.

Issue Remediation

Either explicitly or just due to normal operation, the number of iterations in a loop can grow beyond the block gas limit, which can cause the complete contract to be stalled at a certain point. Therefore, loops with a bigger or unknown number of steps should always be avoided.

Bug ID

SSP_3497_67

Severity

Critical

Confidence

Firm

Line nos

30-55

Action Taken

False Positive

Bug Type

INCORRECT ACCESS CONTROL

File Location

/contracts/zcb-v2/ZeroCouponBondsIssuerV1_AmetFinance.sol

Issue Description

Access control plays an important role in segregation of privileges in smart contracts and other applications. If this is misconfigured or not properly validated on sensitive functions, it may lead to loss of funds, tokens and in some cases compromise of the smart contract.

The contract ZeroCouponBondsIssuerV1_AmetFinance is importing an access control library @openzeppelin/contracts/access/Ownable.sol but the function create is missing the modifier onlyOwner.

Issue Remediation

It is recommended to go through the contract and observe the functions that are lacking an access control modifier. If they contain sensitive administrative actions, it is advised to add a suitable modifier to the same

Bug ID

SSP_3497_13

Severity

Critical

Confidence

Firm

Line nos

29-54

Action Taken

Fixed

Bug Type

INCORRECT ACCESS CONTROL

File Location

/contracts/zcb-v2/ZeroCouponBondsIssuerV1_AmetFinance.sol

Issue Description

Access control plays an important role in segregation of privileges in smart contracts and other applications. If this is misconfigured or not properly validated on sensitive functions, it may lead to loss of funds, tokens and in some cases compromise of the smart contract.

The contract ZeroCouponBondsIssuerV1_AmetFinance is importing an access control library @openzeppelin/contracts/access/Ownable.sol but the function create is missing the modifier onlyOwner.

Issue Remediation

It is recommended to go through the contract and observe the functions that are lacking an access control modifier. If they contain sensitive administrative actions, it is advised to add a suitable modifier to the same

Bug ID

SSP_3497_70

Severity

Low

Confidence

Firm

Line nos

59-62

Action Taken

False Positive

Bug Type

MISSING EVENTS

File Location

/contracts/zcb-v2/ZeroCouponBondsIssuerV1_AmetFinance.sol

Issue Description

Events are inheritable members of contracts. When you call them, they cause the arguments to be stored in the transaction's log—a special data structure in the blockchain.
These logs are associated with the address of the contract which can then be used by developers and auditors to keep track of the transactions.
The contract ZeroCouponBondsIssuerV1_AmetFinance was found to be missing these events on the function withdraw which would make it difficult or impossible to track these transactions off-chain.

Issue Remediation

Consider emitting events for the functions mentioned above. It is also recommended to have the addresses indexed.

Bug ID

SSP_3497_99

Severity

Low

Confidence

Firm

Line nos

148-167

Action Taken

False Positive

Bug Type

MISSING EVENTS

File Location

/contracts/zcb-v2/ZeroCouponBondsV1_AmetFinance.sol

Issue Description

Events are inheritable members of contracts. When you call them, they cause the arguments to be stored in the transaction's log—a special data structure in the blockchain.
These logs are associated with the address of the contract which can then be used by developers and auditors to keep track of the transactions.
The contract ZeroCouponBondsV1_AmetFinance was found to be missing these events on the function purchase which would make it difficult or impossible to track these transactions off-chain.

Issue Remediation

Consider emitting events for the functions mentioned above. It is also recommended to have the addresses indexed.

Bug ID

SSP_3497_81

Severity

Low

Confidence

Firm

Line nos

169-193

Action Taken

False Positive

Bug Type

MISSING EVENTS

File Location

/contracts/zcb-v2/ZeroCouponBondsV1_AmetFinance.sol

Issue Description

Events are inheritable members of contracts. When you call them, they cause the arguments to be stored in the transaction's log—a special data structure in the blockchain.
These logs are associated with the address of the contract which can then be used by developers and auditors to keep track of the transactions.
The contract ZeroCouponBondsV1_AmetFinance was found to be missing these events on the function redeem which would make it difficult or impossible to track these transactions off-chain.

Issue Remediation

Consider emitting events for the functions mentioned above. It is also recommended to have the addresses indexed.

Bug ID

SSP_3497_80

Severity

Low

Confidence

Firm

Line nos

147-164

Action Taken

Fixed

Bug Type

MISSING EVENTS

File Location

/contracts/zcb-v2/ZeroCouponBondsV1_AmetFinance.sol

Issue Description

Events are inheritable members of contracts. When you call them, they cause the arguments to be stored in the transaction's log—a special data structure in the blockchain.
These logs are associated with the address of the contract which can then be used by developers and auditors to keep track of the transactions.
The contract ZeroCouponBondsV1_AmetFinance was found to be missing these events on the function purchase which would make it difficult or impossible to track these transactions off-chain.

Issue Remediation

Consider emitting events for the functions mentioned above. It is also recommended to have the addresses indexed.

Bug ID

SSP_3497_26

Severity

Low

Confidence

Firm

Line nos

75-77

Action Taken

Fixed

Bug Type

MISSING EVENTS

File Location

/contracts/zcb-v2/ZeroCouponBondsIssuerV1_AmetFinance.sol

Issue Description

Events are inheritable members of contracts. When you call them, they cause the arguments to be stored in the transaction's log—a special data structure in the blockchain.
These logs are associated with the address of the contract which can then be used by developers and auditors to keep track of the transactions.
The contract ZeroCouponBondsIssuerV1_AmetFinance was found to be missing these events on the function changePauseState which would make it difficult or impossible to track these transactions off-chain.

Issue Remediation

Consider emitting events for the functions mentioned above. It is also recommended to have the addresses indexed.

Bug ID

SSP_3497_27

Severity

Low

Confidence

Firm

Line nos

100-102

Action Taken

Fixed

Bug Type

MISSING EVENTS

File Location

/contracts/zcb-v2/ZeroCouponBondsV1_AmetFinance.sol

Issue Description

Events are inheritable members of contracts. When you call them, they cause the arguments to be stored in the transaction's log—a special data structure in the blockchain.
These logs are associated with the address of the contract which can then be used by developers and auditors to keep track of the transactions.
The contract ZeroCouponBondsV1_AmetFinance was found to be missing these events on the function changeBaseURI which would make it difficult or impossible to track these transactions off-chain.

Issue Remediation

Consider emitting events for the functions mentioned above. It is also recommended to have the addresses indexed.

Bug ID

SSP_3497_79

Severity

Low

Confidence

Firm

Line nos

131-137

Action Taken

Fixed

Bug Type

MISSING EVENTS

File Location

/contracts/zcb-v2/ZeroCouponBondsV1_AmetFinance.sol

Issue Description

Events are inheritable members of contracts. When you call them, they cause the arguments to be stored in the transaction's log—a special data structure in the blockchain.
These logs are associated with the address of the contract which can then be used by developers and auditors to keep track of the transactions.
The contract ZeroCouponBondsV1_AmetFinance was found to be missing these events on the function withdrawRemaining which would make it difficult or impossible to track these transactions off-chain.

Issue Remediation

Consider emitting events for the functions mentioned above. It is also recommended to have the addresses indexed.

Bug ID

SSP_3497_28

Severity

Low

Confidence

Firm

Line nos

130-136

Action Taken

Fixed

Bug Type

MISSING EVENTS

File Location

/contracts/zcb-v2/ZeroCouponBondsV1_AmetFinance.sol

Issue Description

Events are inheritable members of contracts. When you call them, they cause the arguments to be stored in the transaction's log—a special data structure in the blockchain.
These logs are associated with the address of the contract which can then be used by developers and auditors to keep track of the transactions.
The contract ZeroCouponBondsV1_AmetFinance was found to be missing these events on the function withdrawRemaining which would make it difficult or impossible to track these transactions off-chain.

Issue Remediation

Consider emitting events for the functions mentioned above. It is also recommended to have the addresses indexed.

Bug ID

SSP_3497_29

Severity

Low

Confidence

Firm

Line nos

141-158

Action Taken

Fixed

Bug Type

MISSING EVENTS

File Location

/contracts/zcb-v2/ZeroCouponBondsV1_AmetFinance.sol

Issue Description

Events are inheritable members of contracts. When you call them, they cause the arguments to be stored in the transaction's log—a special data structure in the blockchain.
These logs are associated with the address of the contract which can then be used by developers and auditors to keep track of the transactions.
The contract ZeroCouponBondsV1_AmetFinance was found to be missing these events on the function purchase which would make it difficult or impossible to track these transactions off-chain.

Issue Remediation

Consider emitting events for the functions mentioned above. It is also recommended to have the addresses indexed.

Bug ID

SSP_3497_30

Severity

Low

Confidence

Firm

Line nos

160-184

Action Taken

Fixed

Bug Type

MISSING EVENTS

File Location

/contracts/zcb-v2/ZeroCouponBondsV1_AmetFinance.sol

Issue Description

Events are inheritable members of contracts. When you call them, they cause the arguments to be stored in the transaction's log—a special data structure in the blockchain.
These logs are associated with the address of the contract which can then be used by developers and auditors to keep track of the transactions.
The contract ZeroCouponBondsV1_AmetFinance was found to be missing these events on the function redeem which would make it difficult or impossible to track these transactions off-chain.

Issue Remediation

Consider emitting events for the functions mentioned above. It is also recommended to have the addresses indexed.

Bug ID

SSP_3497_25

Severity

Low

Confidence

Firm

Line nos

58-60

Action Taken

Fixed

Bug Type

MISSING EVENTS

File Location

/contracts/zcb-v2/ZeroCouponBondsIssuerV1_AmetFinance.sol

Issue Description

Events are inheritable members of contracts. When you call them, they cause the arguments to be stored in the transaction's log—a special data structure in the blockchain.
These logs are associated with the address of the contract which can then be used by developers and auditors to keep track of the transactions.
The contract ZeroCouponBondsIssuerV1_AmetFinance was found to be missing these events on the function withdraw which would make it difficult or impossible to track these transactions off-chain.

Issue Remediation

Consider emitting events for the functions mentioned above. It is also recommended to have the addresses indexed.

Bug ID

SSP_3497_104

Severity

Informational

Confidence

Certain

Line nos

23-23

Action Taken

False Positive

Bug Type

MISSING INDEXED KEYWORDS IN EVENTS

File Location

/contracts/zcb-v2/ZeroCouponBondsIssuerV1_AmetFinance.sol

Issue Description

Events are essential for tracking off-chain data and when the event paraemters are `indexed` they can be used as filter options which will help getting only the specific data instead of all the logs.

Issue Remediation

Consider adding `indexed` keyword to crucial event parameters that could be used in off-chain tracking. Do remember that the `indexed` keyword costs more gas.

Bug ID

SSP_3497_34

Severity

Informational

Confidence

Certain

Line nos

23-23

Action Taken

Fixed

Bug Type

MISSING INDEXED KEYWORDS IN EVENTS

File Location

/contracts/zcb-v2/ZeroCouponBondsIssuerV1_AmetFinance.sol

Issue Description

Events are essential for tracking off-chain data and when the event paraemters are `indexed` they can be used as filter options which will help getting only the specific data instead of all the logs.

Issue Remediation

Consider adding `indexed` keyword to crucial event parameters that could be used in off-chain tracking. Do remember that the `indexed` keyword costs more gas.

Bug ID

SSP_3497_35

Severity

Informational

Confidence

Certain

Line nos

51-51

Action Taken

Fixed

Bug Type

MISSING INDEXED KEYWORDS IN EVENTS

File Location

/contracts/zcb-v2/ZeroCouponBondsV1_AmetFinance.sol

Issue Description

Events are essential for tracking off-chain data and when the event paraemters are `indexed` they can be used as filter options which will help getting only the specific data instead of all the logs.

Issue Remediation

Consider adding `indexed` keyword to crucial event parameters that could be used in off-chain tracking. Do remember that the `indexed` keyword costs more gas.

Bug ID

SSP_3497_36

Severity

Informational

Confidence

Certain

Line nos

52-52

Action Taken

Fixed

Bug Type

MISSING INDEXED KEYWORDS IN EVENTS

File Location

/contracts/zcb-v2/ZeroCouponBondsV1_AmetFinance.sol

Issue Description

Events are essential for tracking off-chain data and when the event paraemters are `indexed` they can be used as filter options which will help getting only the specific data instead of all the logs.

Issue Remediation

Consider adding `indexed` keyword to crucial event parameters that could be used in off-chain tracking. Do remember that the `indexed` keyword costs more gas.

Bug ID

SSP_3497_37

Severity

Informational

Confidence

Certain

Line nos

55-55

Action Taken

Fixed

Bug Type

MISSING INDEXED KEYWORDS IN EVENTS

File Location

/contracts/zcb-v2/ZeroCouponBondsV1_AmetFinance.sol

Issue Description

Events are essential for tracking off-chain data and when the event paraemters are `indexed` they can be used as filter options which will help getting only the specific data instead of all the logs.

Issue Remediation

Consider adding `indexed` keyword to crucial event parameters that could be used in off-chain tracking. Do remember that the `indexed` keyword costs more gas.

Bug ID

SSP_3497_59

Severity

Informational

Confidence

Tentative

Line nos

60-60

Action Taken

False Positive

Bug Type

MISSING PAYABLE IN CALL FUNCTION

File Location

/contracts/zcb-v2/ZeroCouponBondsIssuerV1_AmetFinance.sol

Issue Description

The contract is using a `.call()` method to make external calls along with passing some Ether as `msg.value`. Since the function withdraw is not marked as `payable`, the transaction might fail if the contract does not have ETH.

Issue Remediation

If the function needs to pass some Ether as `msg.value` inside a function, make sure to set that function as `payable`. No changes are required if the use case is to send Ether from the contract's balance.

Bug ID

SSP_3497_14

Severity

Informational

Confidence

Tentative

Line nos

17-17

Action Taken

Fixed

Bug Type

MISSING UNDERSCORE IN NAMING VARIABLES

File Location

/contracts/zcb-v2/ZeroCouponBondsV1_AmetFinance.sol

Issue Description

Solidity style guide suggests using underscores as the prefix for non-external functions and state variables (private or internal) but the contract was not found to be following the same.

Issue Remediation

It is recommended to use an underscore for internal and private variables and functions to be in accordance with the Solidity style guide which will also make the code much easier to read.

Bug ID

SSP_3497_15

Severity

Informational

Confidence

Tentative

Line nos

19-19

Action Taken

Fixed

Bug Type

MISSING UNDERSCORE IN NAMING VARIABLES

File Location

/contracts/zcb-v2/ZeroCouponBondsV1_AmetFinance.sol

Issue Description

Solidity style guide suggests using underscores as the prefix for non-external functions and state variables (private or internal) but the contract was not found to be following the same.

Issue Remediation

It is recommended to use an underscore for internal and private variables and functions to be in accordance with the Solidity style guide which will also make the code much easier to read.

Bug ID

SSP_3497_16

Severity

Informational

Confidence

Tentative

Line nos

20-20

Action Taken

Fixed

Bug Type

MISSING UNDERSCORE IN NAMING VARIABLES

File Location

/contracts/zcb-v2/ZeroCouponBondsV1_AmetFinance.sol

Issue Description

Solidity style guide suggests using underscores as the prefix for non-external functions and state variables (private or internal) but the contract was not found to be following the same.

Issue Remediation

It is recommended to use an underscore for internal and private variables and functions to be in accordance with the Solidity style guide which will also make the code much easier to read.

Bug ID

SSP_3497_17

Severity

Informational

Confidence

Tentative

Line nos

21-21

Action Taken

Fixed

Bug Type

MISSING UNDERSCORE IN NAMING VARIABLES

File Location

/contracts/zcb-v2/ZeroCouponBondsV1_AmetFinance.sol

Issue Description

Solidity style guide suggests using underscores as the prefix for non-external functions and state variables (private or internal) but the contract was not found to be following the same.

Issue Remediation

It is recommended to use an underscore for internal and private variables and functions to be in accordance with the Solidity style guide which will also make the code much easier to read.

Bug ID

SSP_3497_18

Severity

Informational

Confidence

Tentative

Line nos

23-23

Action Taken

Fixed

Bug Type

MISSING UNDERSCORE IN NAMING VARIABLES

File Location

/contracts/zcb-v2/ZeroCouponBondsV1_AmetFinance.sol

Issue Description

Solidity style guide suggests using underscores as the prefix for non-external functions and state variables (private or internal) but the contract was not found to be following the same.

Issue Remediation

It is recommended to use an underscore for internal and private variables and functions to be in accordance with the Solidity style guide which will also make the code much easier to read.

Bug ID

SSP_3497_61

Severity

Informational

Confidence

Tentative

Line nos

25-25

Action Taken

Fixed

Bug Type

MISSING UNDERSCORE IN NAMING VARIABLES

File Location

/contracts/zcb-v2/ZeroCouponBondsV1_AmetFinance.sol

Issue Description

Solidity style guide suggests using underscores as the prefix for non-external functions and state variables (private or internal) but the contract was not found to be following the same.

Issue Remediation

It is recommended to use an underscore for internal and private variables and functions to be in accordance with the Solidity style guide which will also make the code much easier to read.

Bug ID

SSP_3497_62

Severity

Informational

Confidence

Tentative

Line nos

26-26

Action Taken

Fixed

Bug Type

MISSING UNDERSCORE IN NAMING VARIABLES

File Location

/contracts/zcb-v2/ZeroCouponBondsV1_AmetFinance.sol

Issue Description

Solidity style guide suggests using underscores as the prefix for non-external functions and state variables (private or internal) but the contract was not found to be following the same.

Issue Remediation

It is recommended to use an underscore for internal and private variables and functions to be in accordance with the Solidity style guide which will also make the code much easier to read.

Bug ID

SSP_3497_63

Severity

Informational

Confidence

Tentative

Line nos

28-28

Action Taken

Fixed

Bug Type

MISSING UNDERSCORE IN NAMING VARIABLES

File Location

/contracts/zcb-v2/ZeroCouponBondsV1_AmetFinance.sol

Issue Description

Solidity style guide suggests using underscores as the prefix for non-external functions and state variables (private or internal) but the contract was not found to be following the same.

Issue Remediation

It is recommended to use an underscore for internal and private variables and functions to be in accordance with the Solidity style guide which will also make the code much easier to read.

Bug ID

SSP_3497_64

Severity

Informational

Confidence

Tentative

Line nos

29-29

Action Taken

Fixed

Bug Type

MISSING UNDERSCORE IN NAMING VARIABLES

File Location

/contracts/zcb-v2/ZeroCouponBondsV1_AmetFinance.sol

Issue Description

Solidity style guide suggests using underscores as the prefix for non-external functions and state variables (private or internal) but the contract was not found to be following the same.

Issue Remediation

It is recommended to use an underscore for internal and private variables and functions to be in accordance with the Solidity style guide which will also make the code much easier to read.

Bug ID

SSP_3497_65

Severity

- **Informational**

Confidence

Tentative

Line nos

31-31

Action Taken

Fixed

Bug Type

MISSING UNDERSCORE IN NAMING VARIABLES

File Location

/contracts/zcb-v2/ZeroCouponBondsV1_AmetFinance.sol

Issue Description

Solidity style guide suggests using underscores as the prefix for non-external functions and state variables (private or internal) but the contract was not found to be following the same.

Issue Remediation

It is recommended to use an underscore for internal and private variables and functions to be in accordance with the Solidity style guide which will also make the code much easier to read.

Bug ID

SSP_3497_66

Severity

Informational

Confidence

Tentative

Line nos

32-32

Action Taken

Fixed

Bug Type

MISSING UNDERSCORE IN NAMING VARIABLES

File Location

/contracts/zcb-v2/ZeroCouponBondsV1_AmetFinance.sol

Issue Description

Solidity style guide suggests using underscores as the prefix for non-external functions and state variables (private or internal) but the contract was not found to be following the same.

Issue Remediation

It is recommended to use an underscore for internal and private variables and functions to be in accordance with the Solidity style guide which will also make the code much easier to read.

Bug ID

SSP_3497_19

Severity

Informational

Confidence

Tentative

Line nos

25-25

Action Taken

Fixed

Bug Type

MISSING UNDERSCORE IN NAMING VARIABLES

File Location

/contracts/zcb-v2/ZeroCouponBondsV1_AmetFinance.sol

Issue Description

Solidity style guide suggests using underscores as the prefix for non-external functions and state variables (private or internal) but the contract was not found to be following the same.

Issue Remediation

It is recommended to use an underscore for internal and private variables and functions to be in accordance with the Solidity style guide which will also make the code much easier to read.

Bug ID

SSP_3497_20

Severity

- **Informational**

Confidence

Tentative

Line nos

26-26

Action Taken

Fixed

Bug Type

MISSING UNDERSCORE IN NAMING VARIABLES

File Location

/contracts/zcb-v2/ZeroCouponBondsV1_AmetFinance.sol

Issue Description

Solidity style guide suggests using underscores as the prefix for non-external functions and state variables (private or internal) but the contract was not found to be following the same.

Issue Remediation

It is recommended to use an underscore for internal and private variables and functions to be in accordance with the Solidity style guide which will also make the code much easier to read.

Bug ID

SSP_3497_21

Severity

Informational

Confidence

Tentative

Line nos

28-28

Action Taken

Fixed

Bug Type

MISSING UNDERSCORE IN NAMING VARIABLES

File Location

/contracts/zcb-v2/ZeroCouponBondsV1_AmetFinance.sol

Issue Description

Solidity style guide suggests using underscores as the prefix for non-external functions and state variables (private or internal) but the contract was not found to be following the same.

Issue Remediation

It is recommended to use an underscore for internal and private variables and functions to be in accordance with the Solidity style guide which will also make the code much easier to read.

Bug ID

SSP_3497_22

Severity

Informational

Confidence

Tentative

Line nos

29-29

Action Taken

Fixed

Bug Type

MISSING UNDERSCORE IN NAMING VARIABLES

File Location

/contracts/zcb-v2/ZeroCouponBondsV1_AmetFinance.sol

Issue Description

Solidity style guide suggests using underscores as the prefix for non-external functions and state variables (private or internal) but the contract was not found to be following the same.

Issue Remediation

It is recommended to use an underscore for internal and private variables and functions to be in accordance with the Solidity style guide which will also make the code much easier to read.

Bug ID

SSP_3497_23

Severity

Informational

Confidence

Tentative

Line nos

31-31

Action Taken

Fixed

Bug Type

MISSING UNDERSCORE IN NAMING VARIABLES

File Location

/contracts/zcb-v2/ZeroCouponBondsV1_AmetFinance.sol

Issue Description

Solidity style guide suggests using underscores as the prefix for non-external functions and state variables (private or internal) but the contract was not found to be following the same.

Issue Remediation

It is recommended to use an underscore for internal and private variables and functions to be in accordance with the Solidity style guide which will also make the code much easier to read.

Bug ID

SSP_3497_24

Severity

Informational

Confidence

Tentative

Line nos

32-32

Action Taken

Fixed

Bug Type

MISSING UNDERSCORE IN NAMING VARIABLES

File Location

/contracts/zcb-v2/ZeroCouponBondsV1_AmetFinance.sol

Issue Description

Solidity style guide suggests using underscores as the prefix for non-external functions and state variables (private or internal) but the contract was not found to be following the same.

Issue Remediation

It is recommended to use an underscore for internal and private variables and functions to be in accordance with the Solidity style guide which will also make the code much easier to read.

Bug ID

SSP_3497_100

Severity

High

Confidence

Certain

Line nos

134-143

Action Taken

False Positive

Bug Type

REENTRANCY

File Location

/contracts/zcb-v2/ZeroCouponBondsV1_AmetFinance.sol

Issue Description

In a Re-entrancy attack, a malicious contract calls back into the calling contract before the first invocation of the function is finished. This may cause the different invocations of the function to interact in undesirable ways, especially in cases where the function is updating state variables after the external calls.
This may lead to loss of funds, improper value updates, token loss, etc.

Issue Remediation

It is recommended to add a [Re-entrancy Guard] to the functions making external calls. The functions should use a Checks-Effects-Interactions pattern. The external calls should be executed at the end of the function and all the state-changing must happen before the call.

Bug ID

SSP_3497_78

Severity

High

Confidence

Certain

Line nos

169-193

Action Taken

False Positive

Bug Type

REENTRANCY

File Location

/contracts/zcb-v2/ZeroCouponBondsV1_AmetFinance.sol

Issue Description

In a Re-entrancy attack, a malicious contract calls back into the calling contract before the first invocation of the function is finished. This may cause the different invocations of the function to interact in undesirable ways, especially in cases where the function is updating state variables after the external calls.
This may lead to loss of funds, improper value updates, token loss, etc.

Issue Remediation

It is recommended to add a [Re-entrancy Guard] to the functions making external calls. The functions should use a Checks-Effects-Interactions pattern. The external calls should be executed at the end of the function and all the state-changing must happen before the call.

Bug ID

SSP_3497_96

Severity

High

Confidence

Certain

Line nos

134-142

Action Taken

Fixed

Bug Type

REENTRANCY

File Location

/contracts/zcb-v2/ZeroCouponBondsV1_AmetFinance.sol

Issue Description

In a Re-entrancy attack, a malicious contract calls back into the calling contract before the first invocation of the function is finished. This may cause the different invocations of the function to interact in undesirable ways, especially in cases where the function is updating state variables after the external calls.
This may lead to loss of funds, improper value updates, token loss, etc.

Issue Remediation

It is recommended to add a [Re-entrancy Guard] to the functions making external calls. The functions should use a Checks-Effects-Interactions pattern. The external calls should be executed at the end of the function and all the state-changing must happen before the call.

Bug ID

SSP_3497_77

Severity

High

Confidence

Certain

Line nos

131-137

Action Taken

Fixed

Bug Type

REENTRANCY

File Location

/contracts/zcb-v2/ZeroCouponBondsV1_AmetFinance.sol

Issue Description

In a Re-entrancy attack, a malicious contract calls back into the calling contract before the first invocation of the function is finished. This may cause the different invocations of the function to interact in undesirable ways, especially in cases where the function is updating state variables after the external calls.
This may lead to loss of funds, improper value updates, token loss, etc.

Issue Remediation

It is recommended to add a [Re-entrancy Guard] to the functions making external calls. The functions should use a Checks-Effects-Interactions pattern. The external calls should be executed at the end of the function and all the state-changing must happen before the call.

Bug ID

SSP_3497_11

Severity

High

Confidence

Certain

Line nos

130-136

Action Taken

Fixed

Bug Type

REENTRANCY

File Location

/contracts/zcb-v2/ZeroCouponBondsV1_AmetFinance.sol

Issue Description

In a Re-entrancy attack, a malicious contract calls back into the calling contract before the first invocation of the function is finished. This may cause the different invocations of the function to interact in undesirable ways, especially in cases where the function is updating state variables after the external calls.
This may lead to loss of funds, improper value updates, token loss, etc.

Issue Remediation

It is recommended to add a [Re-entrancy Guard] to the functions making external calls. The functions should use a Checks-Effects-Interactions pattern. The external calls should be executed at the end of the function and all the state-changing must happen before the call.

Bug ID

SSP_3497_12

Severity

High

Confidence

Certain

Line nos

160-184

Action Taken

Fixed

Bug Type

REENTRANCY

File Location

/contracts/zcb-v2/ZeroCouponBondsV1_AmetFinance.sol

Issue Description

In a Re-entrancy attack, a malicious contract calls back into the calling contract before the first invocation of the function is finished. This may cause the different invocations of the function to interact in undesirable ways, especially in cases where the function is updating state variables after the external calls.
This may lead to loss of funds, improper value updates, token loss, etc.

Issue Remediation

It is recommended to add a [Re-entrancy Guard] to the functions making external calls. The functions should use a Checks-Effects-Interactions pattern. The external calls should be executed at the end of the function and all the state-changing must happen before the call.

Bug ID

SSP_3497_89

Severity

Gas

Confidence

Tentative

Line nos

23-23

Action Taken

False Positive

Bug Type

STORAGE VARIABLE CACHING IN MEMORY

File Location

/contracts/zcb-v2/ZeroCouponBondsV1_AmetFinance.sol

Issue Description

The contract `ZeroCouponBondsV1_AmetFinance` is using the state variable `_feePercentage` multiple times in the function `decreaseFeePercentage`. `SLOADs` are expensive (100 gas after the 1st one) compared to `MLOAD`/`MSTORE` (3 gas each).

Issue Remediation

Storage variables read multiple times inside a function should instead be cached in the memory the first time (costing 1 `SLOAD`) and then read from this cache to avoid multiple `SLOADs`.

Bug ID

SSP_3497_90

Severity

Gas

Confidence

Tentative

Line nos

25-25

Action Taken

False Positive

Bug Type

STORAGE VARIABLE CACHING IN MEMORY

File Location

/contracts/zcb-v2/ZeroCouponBondsV1_AmetFinance.sol

Issue Description

The contract `ZeroCouponBondsV1_AmetFinance` is using the state variable `_redeemLockPeriod` multiple times in the function `decreaseRedeemLockPeriod`. `SLOADs` are expensive (100 gas after the 1st one) compared to `MLOAD`/`MSTORE` (3 gas each).

Issue Remediation

Storage variables read multiple times inside a function should instead be cached in the memory the first time (costing 1 `SLOAD`) and then read from this cache to avoid multiple `SLOADs`.

Bug ID

SSP_3497_91

Severity

Gas

Confidence

Tentative

Line nos

19-19

Action Taken

False Positive

Bug Type

STORAGE VARIABLE CACHING IN MEMORY

File Location

/contracts/zcb-v2/ZeroCouponBondsV1_AmetFinance.sol

Issue Description

The contract `ZeroCouponBondsV1_AmetFinance` is using the state variable `_total` multiple times in the function `issueBonds`.
`SLOADs` are expensive (100 gas after the 1st one) compared to `MLOAD`/`MSTORE` (3 gas each).

Issue Remediation

Storage variables read multiple times inside a function should instead be cached in the memory the first time (costing 1 `SLOAD`) and then read from this cache to avoid multiple `SLOADs`.

Bug ID

SSP_3497_91

Severity

Gas

Confidence

Tentative

Line nos

19-19

Action Taken

False Positive

Bug Type

STORAGE VARIABLE CACHING IN MEMORY

File Location

/contracts/zcb-v2/ZeroCouponBondsV1_AmetFinance.sol

Issue Description

The contract `ZeroCouponBondsV1_AmetFinance` is using the state variable `_total` multiple times in the function `burnUnsoldBonds`.
`SLOADs` are expensive (100 gas after the 1st one) compared to `MLOAD`/`MSTORE` (3 gas each).

Issue Remediation

Storage variables read multiple times inside a function should instead be cached in the memory the first time (costing 1 `SLOAD`) and then read from this cache to avoid multiple `SLOADs`.

Bug ID

SSP_3497_93

Severity

Gas

Confidence

Tentative

Line nos

20-20

Action Taken

False Positive

Bug Type

STORAGE VARIABLE CACHING IN MEMORY

File Location

/contracts/zcb-v2/ZeroCouponBondsV1_AmetFinance.sol

Issue Description

The contract `ZeroCouponBondsV1_AmetFinance` is using the state variable `_purchased` multiple times in the function `purchase`.
`SLOADs` are expensive (100 gas after the 1st one) compared to `MLOAD`/`MSTORE` (3 gas each).

Issue Remediation

Storage variables read multiple times inside a function should instead be cached in the memory the first time (costing 1 `SLOAD`) and then read from this cache to avoid multiple `SLOADs`.

Bug ID

SSP_3497_94

Severity

Gas

Confidence

Tentative

Line nos

21-21

Action Taken

False Positive

Bug Type

STORAGE VARIABLE CACHING IN MEMORY

File Location

/contracts/zcb-v2/ZeroCouponBondsV1_AmetFinance.sol

Issue Description

The contract `ZeroCouponBondsV1_AmetFinance` is using the state variable `_redeemed` multiple times in the function `redeem`.
`SLOADs` are expensive (100 gas after the 1st one) compared to `MLOAD`/`MSTORE` (3 gas each).

Issue Remediation

Storage variables read multiple times inside a function should instead be cached in the memory the first time (costing 1 `SLOAD`) and then read from this cache to avoid multiple `SLOADs`.

Bug ID

SSP_3497_58

Severity

Gas

Confidence

Tentative

Line nos

34-34

Action Taken

False Positive

Bug Type

STORAGE VARIABLE CACHING IN MEMORY

File Location

/contracts/zcb-v2/ZeroCouponBondsV1_AmetFinance.sol

Issue Description

The contract `ZeroCouponBondsV1_AmetFinance` is using the state variable `_purchaseDates` multiple times in the function `redeem`.
`SLOADs` are expensive (100 gas after the 1st one) compared to `MLOAD`/`MSTORE` (3 gas each).

Issue Remediation

Storage variables read multiple times inside a function should instead be cached in the memory the first time (costing 1 `SLOAD`) and then read from this cache to avoid multiple `SLOADs`.

Bug ID

SSP_3497_49

Severity

Gas

Confidence

Tentative

Line nos

16-16

Action Taken

Fixed

Bug Type

STORAGE VARIABLE CACHING IN MEMORY

File Location

/contracts/zcb-v2/ZeroCouponBondsIssuerV1_AmetFinance.sol

Issue Description

The contract `ZeroCouponBondsIssuerV1_AmetFinance` is using the state variable `creationFee` multiple times in the function `changeCreationFee`.
`SLOADs` are expensive (100 gas after the 1st one) compared to `MLOAD`/`MSTORE` (3 gas each).

Issue Remediation

Storage variables read multiple times inside a function should instead be cached in the memory the first time (costing 1 `SLOAD`) and then read from this cache to avoid multiple `SLOADs`.

Bug ID

SSP_3497_50

Severity

Gas

Confidence

Tentative

Line nos

19-19

Action Taken

Fixed

Bug Type

STORAGE VARIABLE CACHING IN MEMORY

File Location

/contracts/zcb-v2/ZeroCouponBondsIssuerV1_AmetFinance.sol

Issue Description

The contract `ZeroCouponBondsIssuerV1_AmetFinance` is using the state variable `creationFeePercentage` multiple times in the function `changeCreationFeePercentage`.
`SLOADs` are expensive (100 gas after the 1st one) compared to `MLOAD` / `MSTORE` (3 gas each).

Issue Remediation

Storage variables read multiple times inside a function should instead be cached in the memory the first time (costing 1 `SLOAD`) and then read from this cache to avoid multiple `SLOADs`.

Bug ID

SSP_3497_51

Severity

Gas

Confidence

Tentative

Line nos

14-14

Action Taken

Fixed

Bug Type

STORAGE VARIABLE CACHING IN MEMORY

File Location

/contracts/zcb-v2/ZeroCouponBondsV1_AmetFinance.sol

Issue Description

The contract `ZeroCouponBondsV1_AmetFinance` is using the state variable `AMET_VAULT` multiple times in the function `changeVaultAddress`.
`SLOADs` are expensive (100 gas after the 1st one) compared to `MLOAD`/`MSTORE` (3 gas each).

Issue Remediation

Storage variables read multiple times inside a function should instead be cached in the memory the first time (costing 1 `SLOAD`) and then read from this cache to avoid multiple `SLOADs`.

Bug ID

SSP_3497_92

Severity

Gas

Confidence

Tentative

Line nos

31-31

Action Taken

Fixed

Bug Type

STORAGE VARIABLE CACHING IN MEMORY

File Location

/contracts/zcb-v2/ZeroCouponBondsV1_AmetFinance.sol

Issue Description

The contract `ZeroCouponBondsV1_AmetFinance` is using the state variable `_interestToken` multiple times in the function `withdrawRemaining`. `SLOADs` are expensive (100 gas after the 1st one) compared to `MLOAD`/`MSTORE` (3 gas each).

Issue Remediation

Storage variables read multiple times inside a function should instead be cached in the memory the first time (costing 1 `SLOAD`) and then read from this cache to avoid multiple `SLOADs`.

Bug ID

SSP_3497_52

Severity

Gas

Confidence

Tentative

Line nos

23-23

Action Taken

Fixed

Bug Type

STORAGE VARIABLE CACHING IN MEMORY

File Location

/contracts/zcb-v2/ZeroCouponBondsV1_AmetFinance.sol

Issue Description

The contract `ZeroCouponBondsV1_AmetFinance` is using the state variable `feePercentage` multiple times in the function `decreaseFeePercentage`. `SLOADs` are expensive (100 gas after the 1st one) compared to `MLOAD`/`MSTORE` (3 gas each).

Issue Remediation

Storage variables read multiple times inside a function should instead be cached in the memory the first time (costing 1 `SLOAD`) and then read from this cache to avoid multiple `SLOADs`.

Bug ID

SSP_3497_73

Severity

Gas

Confidence

Tentative

Line nos

25-25

Action Taken

Fixed

Bug Type

STORAGE VARIABLE CACHING IN MEMORY

File Location

/contracts/zcb-v2/ZeroCouponBondsV1_AmetFinance.sol

Issue Description

The contract `ZeroCouponBondsV1_AmetFinance` is using the state variable `redeemLockPeriod` multiple times in the function `decreaseRedeemLockPeriod`. `SLOADs` are expensive (100 gas after the 1st one) compared to `MLOAD`/`MSTORE` (3 gas each).

Issue Remediation

Storage variables read multiple times inside a function should instead be cached in the memory the first time (costing 1 `SLOAD`) and then read from this cache to avoid multiple `SLOADs`.

Bug ID

SSP_3497_54

Severity

Gas

Confidence

Tentative

Line nos

19-19

Action Taken

Fixed

Bug Type

STORAGE VARIABLE CACHING IN MEMORY

File Location

/contracts/zcb-v2/ZeroCouponBondsV1_AmetFinance.sol

Issue Description

The contract `ZeroCouponBondsV1_AmetFinance` is using the state variable `total` multiple times in the function `issueBonds`.
`SLOADs` are expensive (100 gas after the 1st one) compared to `MLOAD`/`MSTORE` (3 gas each).

Issue Remediation

Storage variables read multiple times inside a function should instead be cached in the memory the first time (costing 1 `SLOAD`) and then read from this cache to avoid multiple `SLOADs`.

Bug ID

SSP_3497_54

Severity

Gas

Confidence

Tentative

Line nos

19-19

Action Taken

Fixed

Bug Type

STORAGE VARIABLE CACHING IN MEMORY

File Location

/contracts/zcb-v2/ZeroCouponBondsV1_AmetFinance.sol

Issue Description

The contract `ZeroCouponBondsV1_AmetFinance` is using the state variable `total` multiple times in the function `burnUnsoldBonds`.
`SLOADs` are expensive (100 gas after the 1st one) compared to `MLOAD`/`MSTORE` (3 gas each).

Issue Remediation

Storage variables read multiple times inside a function should instead be cached in the memory the first time (costing 1 `SLOAD`) and then read from this cache to avoid multiple `SLOADs`.

Bug ID

SSP_3497_74

Severity

Gas

Confidence

Tentative

Line nos

31-31

Action Taken

Fixed

Bug Type

STORAGE VARIABLE CACHING IN MEMORY

File Location

/contracts/zcb-v2/ZeroCouponBondsV1_AmetFinance.sol

Issue Description

The contract `ZeroCouponBondsV1_AmetFinance` is using the state variable `interestToken` multiple times in the function `withdrawRemaining`. `SLOADs` are expensive (100 gas after the 1st one) compared to `MLOAD`/`MSTORE` (3 gas each).

Issue Remediation

Storage variables read multiple times inside a function should instead be cached in the memory the first time (costing 1 `SLOAD`) and then read from this cache to avoid multiple `SLOADs`.

Bug ID

SSP_3497_56

Severity

Gas

Confidence

Tentative

Line nos

20-20

Action Taken

Fixed

Bug Type

STORAGE VARIABLE CACHING IN MEMORY

File Location

/contracts/zcb-v2/ZeroCouponBondsV1_AmetFinance.sol

Issue Description

The contract `ZeroCouponBondsV1_AmetFinance` is using the state variable `purchased` multiple times in the function `purchase`.
`SLOADs` are expensive (100 gas after the 1st one) compared to `MLOAD`/`MSTORE` (3 gas each).

Issue Remediation

Storage variables read multiple times inside a function should instead be cached in the memory the first time (costing 1 `SLOAD`) and then read from this cache to avoid multiple `SLOADs`.

Bug ID

SSP_3497_57

Severity

Gas

Confidence

Tentative

Line nos

21-21

Action Taken

Fixed

Bug Type

STORAGE VARIABLE CACHING IN MEMORY

File Location

/contracts/zcb-v2/ZeroCouponBondsV1_AmetFinance.sol

Issue Description

The contract `ZeroCouponBondsV1_AmetFinance` is using the state variable `redeemed` multiple times in the function `redeem`.
`SLOADs` are expensive (100 gas after the 1st one) compared to `MLOAD`/`MSTORE` (3 gas each).

Issue Remediation

Storage variables read multiple times inside a function should instead be cached in the memory the first time (costing 1 `SLOAD`) and then read from this cache to avoid multiple `SLOADs`.

Bug ID

SSP_3497_53

Severity

Gas

Confidence

Tentative

Line nos

25-25

Action Taken

Fixed

Bug Type

STORAGE VARIABLE CACHING IN MEMORY

File Location

/contracts/zcb-v2/ZeroCouponBondsV1_AmetFinance.sol

Issue Description

The contract `ZeroCouponBondsV1_AmetFinance` is using the state variable `redeemLockPeriod` multiple times in the function `decreaseRedeemLockPeriod`. `SLOADs` are expensive (100 gas after the 1st one) compared to `MLOAD`/`MSTORE` (3 gas each).

Issue Remediation

Storage variables read multiple times inside a function should instead be cached in the memory the first time (costing 1 `SLOAD`) and then read from this cache to avoid multiple `SLOADs`.

Bug ID

SSP_3497_55

Severity

Gas

Confidence

Tentative

Line nos

31-31

Action Taken

Fixed

Bug Type

STORAGE VARIABLE CACHING IN MEMORY

File Location

/contracts/zcb-v2/ZeroCouponBondsV1_AmetFinance.sol

Issue Description

The contract `ZeroCouponBondsV1_AmetFinance` is using the state variable `interestToken` multiple times in the function `withdrawRemaining`. `SLOADs` are expensive (100 gas after the 1st one) compared to `MLOAD`/`MSTORE` (3 gas each).

Issue Remediation

Storage variables read multiple times inside a function should instead be cached in the memory the first time (costing 1 `SLOAD`) and then read from this cache to avoid multiple `SLOADs`.

Bug ID

SSP_3497_84

Severity

Informational

Confidence

Certain

Line nos

30-30

Action Taken

Fixed

Bug Type

UNUSED RECEIVE FALLBACK

File Location

/contracts/zcb-v2/ZeroCouponBondsIssuerV1_AmetFinance.sol

Issue Description

The contract was found to be defining an empty receive function.
It is not recommended to leave them empty unless there's a specific use case such as to receive Ether via an empty `receive()` function.

Issue Remediation

It is recommended to go through the code to make sure these functions are properly implemented and are not missing any validations in the definition.

Bug ID

SSP_3497_48

Severity

Informational

Confidence

Certain

Line nos

59-59

Action Taken

Fixed

Bug Type

USE CALL INSTEAD OF TRANSFER OR SEND

File Location

/contracts/zcb-v2/ZeroCouponBondsIssuerV1_AmetFinance.sol

Issue Description

The contract was found to be using `transfer` or `send` function call. This is unsafe as `transfer` has hard coded gas budget and can fail if the user is a smart contract.

Issue Remediation

It is recommended to use `call` which does not have any hardcoded gas.

Bug ID

SSP_3497_103

Severity

Low

Confidence

Tentative

Line nos

10-78

Action Taken

False Positive

Bug Type

USE OWNABLE2STEP

File Location

/contracts/zcb-v2/ZeroCouponBondsIssuerV1_AmetFinance.sol

Issue Description

`Ownable2Step` is safer than `Ownable` for smart contracts because the owner cannot accidentally transfer the ownership to a mistyped address. Rather than directly transferring to the new owner, the transfer only completes when the new owner accepts ownership.

Issue Remediation

It is recommended to use either `Ownable2Step` or `Ownable2StepUpgradeable` depending on the smart contract.

Bug ID

SSP_3497_98

Severity

Low

Confidence

Tentative

Line nos

10-78

Action Taken

Fixed

Bug Type

USE OWNABLE2STEP

File Location

/contracts/zcb-v2/ZeroCouponBondsIssuerV1_AmetFinance.sol

Issue Description

`Ownable2Step` is safer than `Ownable` for smart contracts because the owner cannot accidentally transfer the ownership to a mistyped address. Rather than directly transferring to the new owner, the transfer only completes when the new owner accepts ownership.

Issue Remediation

It is recommended to use either `Ownable2Step` or `Ownable2StepUpgradeable` depending on the smart contract.

Bug ID

SSP_3497_95

Severity

Low

Confidence

Tentative

Line nos

10-80

Action Taken

Fixed

Bug Type

USE OWNABLE2STEP

File Location

/contracts/zcb-v2/ZeroCouponBondsIssuerV1_AmetFinance.sol

Issue Description

`Ownable2Step` is safer than `Ownable` for smart contracts because the owner cannot accidentally transfer the ownership to a mistyped address. Rather than directly transferring to the new owner, the transfer only completes when the new owner accepts ownership.

Issue Remediation

It is recommended to use either `Ownable2Step` or `Ownable2StepUpgradeable` depending on the smart contract.

Bug ID

SSP_3497_75

Severity

Low

Confidence

Tentative

Line nos

10-78

Action Taken

Fixed

Bug Type

USE OWNABLE2STEP

File Location

/contracts/zcb-v2/ZeroCouponBondsIssuerV1_AmetFinance.sol

Issue Description

`Ownable2Step` is safer than `Ownable` for smart contracts because the owner cannot accidentally transfer the ownership to a mistyped address. Rather than directly transferring to the new owner, the transfer only completes when the new owner accepts ownership.

Issue Remediation

It is recommended to use either `Ownable2Step` or `Ownable2StepUpgradeable` depending on the smart contract.

Bug ID

SSP_3497_60

Severity

Low

Confidence

Tentative

Line nos

10-76

Action Taken

Fixed

Bug Type

USE OWNABLE2STEP

File Location

/contracts/zcb-v2/ZeroCouponBondsIssuerV1_AmetFinance.sol

Issue Description

`Ownable2Step` is safer than `Ownable` for smart contracts because the owner cannot accidentally transfer the ownership to a mistyped address. Rather than directly transferring to the new owner, the transfer only completes when the new owner accepts ownership.

Issue Remediation

It is recommended to use either `Ownable2Step` or `Ownable2StepUpgradeable` depending on the smart contract.

Bug ID

SSP_3497_9

Severity

Low

Confidence

Tentative

Line nos

10-75

Action Taken

Fixed

Bug Type

USE OWNABLE2STEP

File Location

/contracts/zcb-v2/ZeroCouponBondsIssuerV1_AmetFinance.sol

Issue Description

`Ownable2Step` is safer than `Ownable` for smart contracts because the owner cannot accidentally transfer the ownership to a mistyped address. Rather than directly transferring to the new owner, the transfer only completes when the new owner accepts ownership.

Issue Remediation

It is recommended to use either `Ownable2Step` or `Ownable2StepUpgradeable` depending on the smart contract.

Bug ID

SSP_3497_1

Severity

Informational

Confidence

Tentative

Line nos

28-28

Action Taken

Fixed

Bug Type

VARIABLES SHOULD BE IMMUTABLE

File Location

/contracts/zcb-v2/ZeroCouponBondsV1_AmetFinance.sol

Issue Description

Constants and Immutables should be used in their appropriate contexts. `constant` should only be used for literal values written into the code. `immutable` variables should be used for expressions, or values calculated in, or passed into the constructor.

Issue Remediation

It is recommended to use `immutable` instead of `constant`.

Bug ID

SSP_3497_2

Severity

Informational

Confidence

Tentative

Line nos

29-29

Action Taken

Fixed

Bug Type

VARIABLES SHOULD BE IMMUTABLE

File Location

/contracts/zcb-v2/ZeroCouponBondsV1_AmetFinance.sol

Issue Description

Constants and Immutables should be used in their appropriate contexts. `constant` should only be used for literal values written into the code. `immutable` variables should be used for expressions, or values calculated in, or passed into the constructor.

Issue Remediation

It is recommended to use `immutable` instead of `constant`.

Bug ID

SSP_3497_3

Severity

Informational

Confidence

Tentative

Line nos

31-31

Action Taken

Fixed

Bug Type

VARIABLES SHOULD BE IMMUTABLE

File Location

/contracts/zcb-v2/ZeroCouponBondsV1_AmetFinance.sol

Issue Description

Constants and Immutables should be used in their appropriate contexts. `constant` should only be used for literal values written into the code. `immutable` variables should be used for expressions, or values calculated in, or passed into the constructor.

Issue Remediation

It is recommended to use `immutable` instead of `constant`.

Bug ID

SSP_3497_4

Severity

Informational

Confidence

Tentative

Line nos

32-32

Action Taken

Fixed

Bug Type

VARIABLES SHOULD BE IMMUTABLE

File Location

/contracts/zcb-v2/ZeroCouponBondsV1_AmetFinance.sol

Issue Description

Constants and Immutables should be used in their appropriate contexts. `constant` should only be used for literal values written into the code. `immutable` variables should be used for expressions, or values calculated in, or passed into the constructor.

Issue Remediation

It is recommended to use `immutable` instead of `constant`.

Bug ID

SSP_3497_5

Severity

Informational

Confidence

Tentative

Line nos

26-26

Action Taken

Fixed

Bug Type

VARIABLES SHOULD BE IMMUTABLE

File Location

/contracts/zcb-v2/ZeroCouponBondsV1_AmetFinance.sol

Issue Description

Constants and Immutables should be used in their appropriate contexts. `constant` should only be used for literal values written into the code. `immutable` variables should be used for expressions, or values calculated in, or passed into the constructor.

Issue Remediation

It is recommended to use `immutable` instead of `constant`.

Scan History

Critical High Medium Low Informational Gas

No	Date	Security Score	Scan Overview
1.	2023-11-14	100.00	0 0 0 0 0 0
2.	2023-11-14	97.40	0 0 0 0 0 9
3.	2023-11-14	95.80	0 0 0 0 0 14
4.	2023-11-14	95.40	0 0 0 0 1 14
5.	2023-11-14	89.80	0 0 0 5 4 19
6.	2023-11-14	86.00	0 0 0 4 18 19
7.	2023-11-14	76.40	0 3 0 9 25 21

Disclaimer

The Reports neither endorse nor condemn any specific project or team, nor do they guarantee the security of any specific project. The contents of this report do not, and should not be interpreted as having any bearing on, the economics of tokens, token sales, or any other goods, services, or assets.

The security audit is not meant to replace functional testing done before a software release.

There is no warranty that all possible security issues of a particular smart contract(s) will be found by the tool, i.e., It is not guaranteed that there will not be any further findings based solely on the results of this evaluation.

Emerging technologies such as Smart Contracts and Solidity carry a high level of technical risk and uncertainty. There is no warranty or representation made by this report to any Third Party in regards to the quality of code, the business model or the proprietors of any such business model, or the legal compliance of any business.

In no way should a third party use these reports to make any decisions about buying or selling a token, product, service, or any other asset. It should be noted that this report is not investment advice, is not intended to be relied on as investment advice, and has no endorsement of this project or team. It does not serve as a guarantee as to the project's absolute security.

The assessment provided by SolidityScan 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. SolidityScan owes no duty to any third party by virtue of publishing these Reports.

As one audit-based assessment cannot be considered comprehensive, we always recommend proceeding with several independent manual audits including manual audit and a public bug bounty program to ensure the security of the smart contracts.