Powered by

Security Assessment Report

OnChainGMV2

8 Nov 2025

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

Table of Contents

01 Vulnerability Classification and Severity

02 Executive Summary

03 Findings Summary

04 Vulnerability Details

- MISSING SAFE ERC20 USAGE
- REDEMPTION FEE MISAPPLIED
- STRICT EQUALITY CHECK IN BLOCK.TIMESTAMP
- HASH COLLISIONS WITH STRING ARGUMENT ON ABI.ENCODEPACKED
- INCORRECT ERC20 INTERFACE
- PRECISION LOSS DURING DIVISION BY LARGE NUMBERS
- FEE SWITCH OWNER CHECKS MISCONFIGURED
- MINTING DOS IF FEERECIPIENT CANNOT RECEIVE ETH
- REINITIALIZATION VULNERABILITY
- BALANCE EQUALITY
- ERROR-PRONE TYPECASTING
- USE OF FLOATING PRAGMA
- LACK OF ZERO VALUE CHECK IN TOKEN TRANSFERS
- MISSING EVENTS
- MISSING ZERO ADDRESS VALIDATION

NONREENTRANT MODIFIER PLACEMENT
OUTDATED COMPILER VERSION
USE OF _MINT()
USE OWNABLE2STEP
ABI.ENCODEPACKED MAY CAUSE COLLISION
HARD-CODED ADDRESS DETECTED
BLOCK VALUES AS A PROXY FOR TIME
CONTRACT CALLS DISABLEINITIALIZERS IN IT'S CONSTRUCTOR BUT ALSO CONTAINS A INITIALIZATION FUNCTION WHICH UTILIZES THE INITIALIZER MODIFIER
CONSIDER USING UINT48 FOR TIME-RELATED VARIABLES
CONTRACT NAME SHOULD USE PASCALCASE
MISSING @AUTHOR IN NATSPEC COMMENTS FOR CONTRACT DECLARATION
MISSING @DEV IN NATSPEC COMMENTS FOR CONTRACT DECLARATION
MISSING @DEV IN NATSPEC COMMENTS FOR FUNCTIONS
MISSING @INHERITDOC ON OVERRIDE FUNCTIONS
MISSING NATSPEC COMMENTS IN SCOPE BLOCKS
MISSING NATSPEC DESCRIPTIONS FOR PUBLIC VARIABLE DECLARATIONS
MISSING @NOTICE IN NATSPEC COMMENTS FOR CONSTRUCTORS
MISSING @NOTICE IN NATSPEC COMMENTS FOR FUNCTIONS
MISSING PAYABLE IN CALL FUNCTION
MISSING UNDERSCORE IN NAMING VARIABLES
REQUIRE INSTEAD OF REVERT

REVERT STATEMENTS WITHIN EXTERNAL AND PUBLIC FUNCTIONS CAN BE USED TO PERFORM DOS ATTACKS

UNNAMED FUNCTION PARAMETERS

UNUSED RECEIVE FALLBACK

AVOID RE-STORING VALUES

AVOID ZERO-TO-ONE STORAGE WRITES

CACHE ADDRESS(THIS) WHEN USED MORE THAN ONCE

CHEAPER INEQUALITIES IN IF()

CUSTOM ERRORS TO SAVE GAS

DEFAULT INT VALUES ARE MANUALLY RESET

DEFINE CONSTRUCTOR AS PAYABLE

REVERTING FUNCTIONS CAN BE PAYABLE

GAS INEFFICIENCY DUE TO MULTIPLE OPERANDS IN SINGLE IF/ELSEIF CONDITION

GAS OPTIMIZATION IN INCREMENTS

INTERNAL FUNCTIONS NEVER USED

LONG REQUIRE/REVERT STRINGS

NAMED RETURN OF LOCAL VARIABLE SAVES GAS AS COMPARED TO RETURN STATEMENT

PUBLIC CONSTANTS CAN BE PRIVATE

SPLITTING REVERT STATEMENTS

STORAGE VARIABLE CACHING IN MEMORY

UNNECESSARY CHECKED ARITHMETIC IN LOOP

USE BYTES.CONCAT() INSTEAD OF ABI.ENCODEPACKED

USE SELFBALANCE() INSTEAD OF ADDRESS(THIS).BALANCE

05 Scan History

06 Disclaimer

01. Vulnerability Classification and Severity

Description

To enhance navigability, the document is organized in descending order of severity for easy reference. Issues are categorized as ✓ *Fixed*, ⚠ *Pending Fix*, or *Won't Fix*, indicating their current status. *Won't Fix* denotes that the team is aware of the issue but has chosen not to resolve it. Issues labeled as ⚠ *Pending Fix* state that the bug is yet to be resolved. Additionally, each issue's severity is assessed based on the risk of exploitation or the potential for other unexpected or unsafe behavior.

● Critical

The issue affects the contract in such a way that funds may be lost, allocated incorrectly, or otherwise result in a significant loss.

● High

High-severity vulnerabilities pose a significant risk to both the Smart Contract and the organization. They can lead to user fund losses, may have conditional requirements, and are challenging to exploit.

● Medium

The issue affects the ability of the contract to operate in a way that doesn't significantly hinder its behavior.

● Low

The issue has minimal impact on the contract's ability to operate.

● Informational

The issue does not affect the contract's operational capability but is considered good practice to address.

● Gas

This category deals with optimizing code and refactoring to conserve gas.

02. Executive Summary

OnChainGMV2

Github Project
https://github.com/OnChainGM/OnChainGMV2

Language	Audit Methodology	Commit Hash
Solidity	**Static Scanning**	-
Website	Publishers/Owner Name	Organization
-	-	-
Contact Email		
-		

Security Score is GREAT

The SolidityScan score is calculated based on lines of code and weights assigned to each issue depending on the severity and confidence. To improve your score, view the detailed result and leverage the remediation solutions provided.

This report has been prepared for OnChainGMV2 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 700+ modules. The scanning and auditing process covers the following areas:

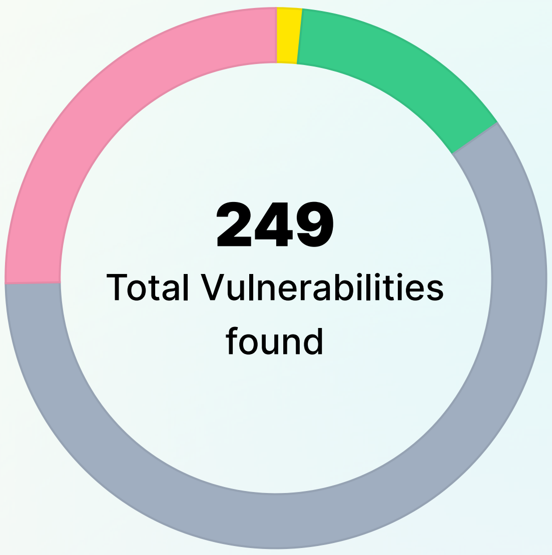
Various common and uncommon attack vectors will be investigated to ensure that the smart contracts are secure from malicious actors. The scanner modules find and flag issues related to Gas optimizations that help in reducing the overall Gas cost It scans and evaluates the codebase against industry best 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 OnChainGMV2 introduces new features or refactors the code.

03. Findings Summary

OnChainGMV2
View on Github

Security Score	Scan duration	Lines of code
80.41/100	247 secs	367



Crit	High	Med	Low	Info	Gas
0	0	4	34	148	63

This audit report has not been verified by the SolidityScan team. To learn more about our published reports. click here

ACTION TAKEN

0	8	0	250
Fixed	*False Positive*	*Won't Fix*	*Pending Fix*

S. No.	Severity	Bug Type	Instances	Detection Method	Status
H001	● High	MISSING SAFE ERC20 USAGE	1	SolidityScan AI	*False Positive*
M001	● Medium	REDEMPTION FEE MISAPPLIED	1	SolidityScan AI	*False Positive*
M002	● Medium	STRICT EQUALITY CHECK IN BLOCK.TIMESTAMP	1	Automated	*False Positive*
M003	● Medium	HASH COLLISIONS WITH STRING ARGUMENT ON ABI.ENCODEPACKED	3	Automated	*False Positive*
M004	● Medium	INCORRECT ERC20 INTERFACE	2	Automated	*False Positive*
M005	● Medium	PRECISION LOSS DURING DIVISION BY LARGE NUMBERS	4	Automated	*Pending Fix*

S. No.	Severity	Bug Type	Instances	Detection Method	Status
L001	● Low	FEE SWITCH OWNER CHECKS MISCONFIGURED	1	SolidityScan AI	⚠ *Pending Fix*
L002	● Low	MINTING DOS IF FEERECIPIENT CANNOT RECEIVE ETH	1	SolidityScan AI	⚠ *Pending Fix*
L003	● Low	REINITIALIZATION VULNERABILITY	1	SolidityScan AI	⚠ *Pending Fix*
L004	● Low	BALANCE EQUALITY	2	Automated	⚠ *Pending Fix*
L005	● Low	ERROR-PRONE TYPECASTING	11	Automated	⚠ *Pending Fix*
L006	● Low	USE OF FLOATING PRAGMA	1	Automated	⚠ *Pending Fix*
L007	● Low	LACK OF ZERO VALUE CHECK IN TOKEN TRANSFERS	1	Automated	⚠ *Pending Fix*
L008	● Low	MISSING EVENTS	10	Automated	⚠ *Pending Fix*
L009	● Low	MISSING ZERO ADDRESS VALIDATION	2	Automated	⚠ *Pending Fix*
L010	● Low	NONREENTRANT MODIFIER PLACEMENT	3	Automated	⚠ *Pending Fix*
L011	● Low	OUTDATED COMPILER VERSION	1	Automated	⚠ *Pending Fix*
L012	● Low	USE OF _MINT()	1	Automated	⚠ *Pending Fix*
L013	● Low	USE OWNABLE2STEP	1	Automated	⚠ *Pending Fix*
I001	● Informational	ABI.ENCODEPACKED MAY CAUSE COLLISION	3	Automated	⚠ *Pending Fix*
I002	● Informational	HARD-CODED ADDRESS DETECTED	1	Automated	⚠ *Pending Fix*
I003	● Informational	BLOCK VALUES AS A PROXY FOR TIME	4	Automated	⚠ *Pending Fix*
I004	● Informational	CONTRACT CALLS DISABLEINITIALIZERS IN IT'S CONSTRUCTOR BUT ALSO CONTAINS A INITIALIZATION FUNCTION WHICH UTILIZES THE INITIALIZER MODIFIER	1	Automated	⚠ *Pending Fix*

S. No.	Severity	Bug Type	Instances	Detection Method	Status
I005	● Informational	CONSIDER USING UINT48 FOR TIME-RELATED VARIABLES	1	Automated	⚠ *Pending Fix*
I006	● Informational	CONTRACT NAME SHOULD USE PASCALCASE	1	Automated	⚠ *Pending Fix*
I007	● Informational	MISSING @AUTHOR IN NATSPEC COMMENTS FOR CONTRACT DECLARATION	1	Automated	⚠ *Pending Fix*
I008	● Informational	MISSING @DEV IN NATSPEC COMMENTS FOR CONTRACT DECLARATION	1	Automated	⚠ *Pending Fix*
I009	● Informational	MISSING @DEV IN NATSPEC COMMENTS FOR FUNCTIONS	17	Automated	⚠ *Pending Fix*
I010	● Informational	MISSING @INHERITDOC ON OVERRIDE FUNCTIONS	27	Automated	⚠ *Pending Fix*
I011	● Informational	MISSING NATSPEC COMMENTS IN SCOPE BLOCKS	32	Automated	⚠ *Pending Fix*
I012	● Informational	MISSING NATSPEC DESCRIPTIONS FOR PUBLIC VARIABLE DECLARATIONS	10	Automated	⚠ *Pending Fix*
I013	● Informational	MISSING @NOTICE IN NATSPEC COMMENTS FOR CONSTRUCTORS	1	Automated	⚠ *Pending Fix*
I014	● Informational	MISSING @NOTICE IN NATSPEC COMMENTS FOR FUNCTIONS	32	Automated	⚠ *Pending Fix*
I015	● Informational	MISSING PAYABLE IN CALL FUNCTION	2	Automated	⚠ *Pending Fix*
I016	● Informational	MISSING UNDERSCORE IN NAMING VARIABLES	3	Automated	⚠ *Pending Fix*
I017	● Informational	REQUIRE INSTEAD OF REVERT	5	Automated	⚠ *Pending Fix*

S. No.	Severity	Bug Type	Instances	Detection Method	Status
I018	● Informational	REVERT STATEMENTS WITHIN EXTERNAL AND PUBLIC FUNCTIONS CAN BE USED TO PERFORM DOS ATTACKS	12	Automated	⚠ *Pending Fix*
I019	● Informational	UNNAMED FUNCTION PARAMETERS	4	Automated	⚠ *Pending Fix*
I020	● Informational	UNUSED RECEIVE FALLBACK	1	Automated	⚠ *Pending Fix*
G001	● Gas	AVOID RE-STORING VALUES	4	Automated	⚠ *Pending Fix*
G002	● Gas	AVOID ZERO-TO-ONE STORAGE WRITES	7	Automated	⚠ *Pending Fix*
G003	● Gas	CACHE ADDRESS(THIS) WHEN USED MORE THAN ONCE	5	Automated	⚠ *Pending Fix*
G004	● Gas	CHEAPER INEQUALITIES IN IF()	3	Automated	⚠ *Pending Fix*
G005	● Gas	CUSTOM ERRORS TO SAVE GAS	9	Automated	⚠ *Pending Fix*
G006	● Gas	DEFAULT INT VALUES ARE MANUALLY RESET	1	Automated	⚠ *Pending Fix*
G007	● Gas	DEFINE CONSTRUCTOR AS PAYABLE	1	Automated	⚠ *Pending Fix*
G008	● Gas	REVERTING FUNCTIONS CAN BE PAYABLE	8	Automated	⚠ *Pending Fix*
G009	● Gas	GAS INEFFICIENCY DUE TO MULTIPLE OPERANDS IN SINGLE IF/ELSEIF CONDITION	8	Automated	⚠ *Pending Fix*
G010	● Gas	GAS OPTIMIZATION IN INCREMENTS	1	Automated	⚠ *Pending Fix*
G011	● Gas	INTERNAL FUNCTIONS NEVER USED	1	Automated	⚠ *Pending Fix*
G012	● Gas	LONG REQUIRE/REVERT STRINGS	4	Automated	⚠ *Pending Fix*
G013	● Gas	NAMED RETURN OF LOCAL VARIABLE SAVES GAS AS COMPARED TO RETURN STATEMENT	2	Automated	⚠ *Pending Fix*
G014	● Gas	PUBLIC CONSTANTS CAN BE PRIVATE	1	Automated	⚠ *Pending Fix*

S. No.	Severity	Bug Type	Instances	Detection Method	Status
G015	● Gas	SPLITTING REVERT STATEMENTS	5	Automated	⚠ *Pending Fix*
G016	● Gas	STORAGE VARIABLE CACHING IN MEMORY	4	Automated	⚠ *Pending Fix*
G017	● Gas	UNNECESSARY CHECKED ARITHMETIC IN LOOP	1	Automated	⚠ *Pending Fix*
G018	● Gas	USE BYTES.CONCAT() INSTEAD OF ABI.ENCODEPACKED	3	Automated	⚠ *Pending Fix*
G019	● Gas	USE SELFBALANCE() INSTEAD OF ADDRESS(THIS).BALANCE	3	Automated	⚠ *Pending Fix*

04. Vulnerability Details

Issue Type

MISSING SAFE ERC20 USAGE

S. No.	Severity	Detection Method	Instances
H001	● High	SolidityScan AI	1

Bug ID	File Location	Line No.	Action Taken
SSP_112517_254	--	--	*False Positive*

Upgrade your Plan to view the full report

1 High Issues Found

Please upgrade your plan to view all the issues in your report.

Upgrade

Issue Type

REDEMPTION FEE MISAPPLIED

S. No.	Severity	Detection Method	Instances
M001	● Medium	SolidityScan AI	1

Bug ID	File Location	Line No.	Action Taken
SSP_112517_255	--	--	*False Positive*

Upgrade your Plan to view the full report

1 Medium Issues Found

Please upgrade your plan to view all the issues in your report.

Upgrade

Issue Type

FEE SWITCH OWNER CHECKS MISCONFIGURED

S. No.	Severity	Detection Method	Instances
L001	● Low	SolidityScan AI	1

Bug ID	File Location	Line No.	Action Taken
SSP_112517_258	--	--	*Pending Fix*

Upgrade your Plan to view the full report

1 Low Issues Found

Please upgrade your plan to view all the issues in your report.

Upgrade

Issue Type

ABI.ENCODEPACKED MAY CAUSE COLLISION

S. No.	Severity	Detection Method	Instances
I001	● Informational	Automated	3

Bug ID	File Location	Line No.	Action Taken
SSP_112517_31	--	--	⚠ *Pending Fix*
SSP_112517_32	--	--	⚠ *Pending Fix*
SSP_112517_33	--	--	⚠ *Pending Fix*

Upgrade your Plan to view the full report

3 Informational Issues Found

Please upgrade your plan to view all the issues in your report.

Upgrade

Issue Type

AVOID RE-STORING VALUES

S. No.	Severity	Detection Method	Instances
G001	● Gas	Automated	4

Description

The function is found to be allowing re-storing the value in the contract's state variable even when the old value is eq ual to the new value. This practice results in unnecessary gas consumption due to the `Gsreset` operation (2900 ga s), which could be avoided. If the old value and the new value are the same, not updating the storage would avoid thi s cost and could instead incur a `Gcoldsload` (2100 gas) or a `Gwarmaccess` (100 gas), potentially saving gas.

Bug ID	File Location	Line No.	Action Taken
SSP_112517_229	OnChainGMV2.sol	L291 - L296	*Pending Fix*
SSP_112517_230	OnChainGMV2.sol	L301 - L306	*Pending Fix*
SSP_112517_231	OnChainGMV2.sol	L311 - L316	*Pending Fix*
SSP_112517_232	OnChainGMV2.sol	L321 - L326	*Pending Fix*

Issue Type

AVOID ZERO-TO-ONE STORAGE WRITES

S. No.	Severity	Detection Method	Instances
G002	● Gas	Automated	7

Description

Writing a storage variable from zero to a non-zero value costs 22,100 gas (20,000 for the write and 2,100 for cold acc ess), making it one of the most expensive operations. This is why patterns like OpenZeppelin's `ReentrancyGuard` us e `1` and `2` instead of `0` and `1`—to avoid the high cost of zero-to-non-zero writes. Non-zero to non-zero updates cost only 5,000 gas.

Bug ID	File Location	Line No.	Action Taken
SSP_112517_193	OnChainGMV2.sol	L59 - L59	⚠ *Pending Fix*
SSP_112517_194	OnChainGMV2.sol	L61 - L61	⚠ *Pending Fix*
SSP_112517_195	OnChainGMV2.sol	L62 - L62	⚠ *Pending Fix*
SSP_112517_196	OnChainGMV2.sol	L63 - L63	⚠ *Pending Fix*
SSP_112517_197	OnChainGMV2.sol	L305 - L305	⚠ *Pending Fix*
SSP_112517_198	OnChainGMV2.sol	L315 - L315	⚠ *Pending Fix*
SSP_112517_199	OnChainGMV2.sol	L325 - L325	⚠ *Pending Fix*

Issue Type

CACHE ADDRESS(THIS) WHEN USED MORE THAN ONCE

S. No.	Severity	Detection Method	Instances
G003	● Gas	Automated	5

Description

The repeated usage of `address(this)` within the contract could result in increased gas costs due to multiple executions of the same computation, potentially impacting efficiency and overall transaction expenses.

Bug ID	File Location	Line No.	Action Taken
SSP_112517_21	OnChainGMV2.sol	L336 - L336	*Pending Fix*
SSP_112517_21	OnChainGMV2.sol	L360 - L360	*Pending Fix*
SSP_112517_22	OnChainGMV2.sol	L375 - L375	*Pending Fix*
SSP_112517_23	OnChainGMV2.sol	L386 - L386	*Pending Fix*
SSP_112517_24	OnChainGMV2.sol	L406 - L406	*Pending Fix*

Issue Type

CHEAPER INEQUALITIES IN IF()

S. No.	Severity	Detection Method	Instances
G004	● Gas	Automated	3

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 `(>, <)`.

Bug ID	File Location	Line No.	Action Taken
SSP_112517_57	OnChainGMV2.sol	L159 - L159	*Pending Fix*
SSP_112517_58	OnChainGMV2.sol	L342 - L342	*Pending Fix*
SSP_112517_59	OnChainGMV2.sol	L392 - L392	*Pending Fix*

Issue Type

CUSTOM ERRORS TO SAVE GAS

S. No.	Severity	Detection Method	Instances
G005	● Gas	Automated	9

Description

The contract was found to be using `revert()` statements. Since Solidity `v0.8.4`, custom errors have been introduced which are a better alternative to the revert.
This allows the developers to pass custom errors with dynamic data while reverting the transaction and also making the whole implementation a bit cheaper than using `revert`.

Bug ID	File Location	Line No.	Action Taken
SSP_112517_234	OnChainGMV2.sol	L338 - L338	*Pending Fix*
SSP_112517_234	OnChainGMV2.sol	L362 - L362	*Pending Fix*
SSP_112517_235	OnChainGMV2.sol	L343 - L343	*Pending Fix*
SSP_112517_236	OnChainGMV2.sol	L388 - L388	*Pending Fix*
SSP_112517_237	OnChainGMV2.sol	L393 - L393	*Pending Fix*
SSP_112517_238	OnChainGMV2.sol	L411 - L411	*Pending Fix*
SSP_112517_238	OnChainGMV2.sol	L415 - L415	*Pending Fix*
SSP_112517_239	OnChainGMV2.sol	L419 - L419	*Pending Fix*

Bug ID	File Location	Line No.	Action Taken
SSP_112517_239	OnChainGMV2.sol	L423 - L423	⚠ *Pending Fix*

Issue Type

DEFAULT INT VALUES ARE MANUALLY RESET

S. No.	Severity	Detection Method	Instances
G006	● Gas	Automated	1

Description

The contract is found to inefficiently reset integer variables to their default value of zero using manual assignment. In Solidity, manually setting a variable to its default value does not free up storage space, leading to unnecessary gas c onsumption. Instead, using the `.delete` keyword can achieve the same result while also freeing up storage space on the Ethereum blockchain, resulting in gas cost savings.

Bug ID	File Location	Line No.	Action Taken
SSP_112517_16	OnChainGMV2.sol	L59 - L59	⚠ *Pending Fix*

Issue Type

DEFINE CONSTRUCTOR AS PAYABLE

S. No.	Severity	Detection Method	Instances
G007	● Gas	Automated	1

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 deployme
nt.

Bug ID	File Location	Line No.	Action Taken
SSP_112517_241	OnChainGMV2.sol	L47 - L49	⚠ *Pending Fix*

Issue Type

REVERTING FUNCTIONS CAN BE PAYABLE

S. No.	Severity	Detection Method	Instances
G008	● Gas	Automated	8

Description

If a function modifier such as `onlyOwner` is used, the function will revert if a normal user tries to pay the function. Marking the function as payable will lower the gas cost for legitimate callers because the compiler will not include checks for whether a payment was provided.

Bug ID	File Location	Line No.	Action Taken
SSP_112517_244	OnChainGMV2.sol	L70 - L71	⚠ *Pending Fix*
SSP_112517_245	OnChainGMV2.sol	L291 - L296	⚠ *Pending Fix*
SSP_112517_246	OnChainGMV2.sol	L301 - L306	⚠ *Pending Fix*
SSP_112517_247	OnChainGMV2.sol	L311 - L316	⚠ *Pending Fix*
SSP_112517_248	OnChainGMV2.sol	L321 - L326	⚠ *Pending Fix*
SSP_112517_249	OnChainGMV2.sol	L331 - L350	⚠ *Pending Fix*
SSP_112517_250	OnChainGMV2.sol	L355 - L369	⚠ *Pending Fix*
SSP_112517_251	OnChainGMV2.sol	L381 - L397	⚠ *Pending Fix*

Issue Type

GAS INEFFICIENCY DUE TO MULTIPLE OPERANDS IN SINGLE IF/ELSEIF CONDITION

S. No.	Severity	Detection Method	Instances
G009	● Gas	Automated	8

Description

The contract is found to use multiple operands within a single `if` or `else if` statement, which can lead to unnecessar y gas consumption due to the way the EVM evaluates compound boolean expressions. Each operand in a compound condition is evaluated even if the first condition fails, unless short-circuiting occurs, and the combined logic can resul t in more complex bytecode and higher gas usage compared to using nested `if` statements. This inefficiency is parti cularly relevant in functions that are called frequently or within loops.

Bug ID	File Location	Line No.	Action Taken
SSP_112517_148	OnChainGMV2.sol	L84 - L86	*Pending Fix*
SSP_112517_149	OnChainGMV2.sol	L159 - L161	*Pending Fix*
SSP_112517_150	OnChainGMV2.sol	L218 - L241	*Pending Fix*
SSP_112517_151	OnChainGMV2.sol	L244 - L246	*Pending Fix*
SSP_112517_152	OnChainGMV2.sol	L302 - L304	*Pending Fix*
SSP_112517_153	OnChainGMV2.sol	L312 - L314	*Pending Fix*
SSP_112517_154	OnChainGMV2.sol	L322 - L324	*Pending Fix*
SSP_112517_155	OnChainGMV2.sol	L382 - L384	*Pending Fix*

Issue Type

GAS OPTIMIZATION IN INCREMENTS

S. No.	Severity	Detection Method	Instances
G010	● Gas	Automated	1

Description

`++i` costs less gas compared to `i++` or `i += 1` for unsigned integers. In `i++`, the compiler has to create a temporary variable to store the initial value. This is not the case with `++i` in which the value is directly incremented and returned, thus, making it a cheaper alternative.

Bug ID	File Location	Line No.	Action Taken
SSP_112517_221	OnChainGMV2.sol	L249 - L249	⚠ *Pending Fix*

Issue Type

INTERNAL FUNCTIONS NEVER USED

S. No.	Severity	Detection Method	Instances
G011	● Gas	Automated	1

Description

The contract declared internal functions but was not using them in any of the functions or contracts.
Since internal functions can only be called from inside the contracts, it makes no sense to have them if they are not u sed. This uses up gas and causes issues for auditors when understanding the contract logic.

Bug ID	File Location	Line No.	Action Taken
SSP_112517_30	OnChainGMV2.sol	L97 - L101	⚠ *Pending Fix*

Issue Type

LONG REQUIRE/REVERT STRINGS

S. No.	Severity	Detection Method	Instances
G012	● Gas	Automated	4

Description

The require() and revert() functions take an input string to show errors if the validation fails.
This strings inside these functions that are longer than 32 bytes require at least one additional MSTORE, along with additional overhead for computing memory offset, and other parameters.

Bug ID	File Location	Line No.	Action Taken
SSP_112517_242	OnChainGMV2.sol	L411 - L411	*Pending Fix*
SSP_112517_242	OnChainGMV2.sol	L415 - L415	*Pending Fix*
SSP_112517_243	OnChainGMV2.sol	L419 - L419	*Pending Fix*
SSP_112517_243	OnChainGMV2.sol	L423 - L423	*Pending Fix*

Issue Type

NAMED RETURN OF LOCAL VARIABLE SAVES GAS AS COMPARED TO RETURN STATEMENT

S. No.	Severity	Detection Method	Instances
G013	● Gas	Automated	2

Description

The function having a return type is found to be declaring a local variable for returning, which causes extra gas cons umption. This inefficiency arises because creating and manipulating local variables requires additional computational steps and memory allocation.

Bug ID	File Location	Line No.	Action Taken
SSP_112517_53	OnChainGMV2.sol	L83 - L95	*Pending Fix*
SSP_112517_54	OnChainGMV2.sol	L138 - L146	*Pending Fix*

Issue Type

PUBLIC CONSTANTS CAN BE PRIVATE

S. No.	Severity	Detection Method	Instances
G014	● Gas	Automated	1

Description

Public constant variables cost more gas because the EVM automatically creates getter functions for them and adds entries to the method ID table. The values can be read from the source code instead.

Bug ID	File Location	Line No.	Action Taken
SSP_112517_161	OnChainGMV2.sol	L33 - L33	*Pending Fix*

Issue Type

SPLITTING REVERT STATEMENTS

S. No.	Severity	Detection Method	Instances
G015	● Gas	Automated	5

Description

The contract is using multiple conditions in a single `if` statement followed by a revert. This costs some extra gas.

Bug ID	File Location	Line No.	Action Taken
SSP_112517_102	OnChainGMV2.sol	L84 - L86	⚠ *Pending Fix*
SSP_112517_103	OnChainGMV2.sol	L302 - L304	⚠ *Pending Fix*
SSP_112517_104	OnChainGMV2.sol	L312 - L314	⚠ *Pending Fix*
SSP_112517_105	OnChainGMV2.sol	L322 - L324	⚠ *Pending Fix*
SSP_112517_106	OnChainGMV2.sol	L382 - L384	⚠ *Pending Fix*

Issue Type

STORAGE VARIABLE CACHING IN MEMORY

S. No.	Severity	Detection Method	Instances
G016	● Gas	Automated	4

Description

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

Bug ID	File Location	Line No.	Action Taken
SSP_112517_222	OnChainGMV2.sol	L155 - L164	⚠ *Pending Fix*
SSP_112517_223	OnChainGMV2.sol	L214 - L257	⚠ *Pending Fix*
SSP_112517_223	OnChainGMV2.sol	L214 - L257	⚠ *Pending Fix*
SSP_112517_224	OnChainGMV2.sol	L355 - L369	⚠ *Pending Fix*

Issue Type

UNNECESSARY CHECKED ARITHMETIC IN LOOP

S. No.	Severity	Detection Method	Instances
G017	● Gas	Automated	1

Description

Increments inside a loop could never overflow due to the fact that the transaction will run out of gas before the varia ble reaches its limits. Therefore, it makes no sense to have checked arithmetic in such a place.

Bug ID	File Location	Line No.	Action Taken
SSP_112517_92	OnChainGMV2.sol	L249 - L249	⚠ *Pending Fix*

Issue Type

USE BYTES.CONCAT() INSTEAD OF ABI.ENCODEPACKED

S. No.	Severity	Detection Method	Instances
G018	● Gas	Automated	3

Description

The contract is found to use abi.encodePacked for concatenating byte variables, which is less gas-efficient compared to using bytes.concat. When concatenation isn't used for hashing operations, preferring bytes.concat can result in more optimized and cost-effective gas consumption.

Bug ID	File Location	Line No.	Action Taken
SSP_112517_201	OnChainGMV2.sol	L179 - L179	*Pending Fix*
SSP_112517_202	OnChainGMV2.sol	L181 - L181	*Pending Fix*
SSP_112517_203	OnChainGMV2.sol	L191 - L191	*Pending Fix*

Issue Type

USE SELFBALANCE() INSTEAD OF ADDRESS(THIS).BALANCE

S. No.	Severity	Detection Method	Instances
G019	● Gas	Automated	3

Description

In Solidity, efficient use of gas is paramount to ensure cost-effective execution on the Ethereum blockchain. Gas can be optimized when obtaining contract balance by using `selfbalance()` rather than `address(this).balance` because it bypasses gas costs and refunds, which are not required for obtaining the contract's balance.

Bug ID	File Location	Line No.	Action Taken
SSP_112517_28	OnChainGMV2.sol	L336 - L336	*Pending Fix*
SSP_112517_28	OnChainGMV2.sol	L360 - L360	*Pending Fix*
SSP_112517_29	OnChainGMV2.sol	L375 - L375	*Pending Fix*

05. Scan History

● Critical ● High ● Medium ● Low ● Informational ● Gas

No	Date	Security Score	Scan Overview
1.	2025-11-08	**80.41**	● 0 ● 0 ● 4 ● 34 ● 148 ● 63

06. **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.