SolidityScan | Powered by CRED SHIELDS

Security Assessment Report

gmBoost

21 Nov 2025

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

Self-published

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

Table of Contents

ERC1155 CALLBACK REENTRANCY
ERC777 CALLBACK REENTRANCY
INSECURE OWNERSHIP TRANSFER
MISSING ONLYOWNER MODIFIER
BALANCE EQUALITY
EVENT BASED REENTRANCY
MISSING ZERO ADDRESS VALIDATION
MISSING @AUTHOR IN NATSPEC COMMENTS FOR CONTRACT DECLARATION
MISSING @DEV IN NATSPEC COMMENTS FOR CONTRACT DECLARATION
MISSING @DEV IN NATSPEC COMMENTS FOR FUNCTIONS
MISSING INHERITANCE
MISSING @INHERITDOC ON OVERRIDE FUNCTIONS
MISSING NATSPEC COMMENTS IN SCOPE BLOCKS
MISSING NATSPEC DESCRIPTIONS FOR PUBLIC VARIABLE DECLARATIONS
MISSING @NOTICE IN NATSPEC COMMENTS FOR FUNCTIONS
MISSING @NOTICE IN NATSPEC COMMENTS FOR MODIFIERS
MISSING @PARAM IN NATSPEC COMMENTS FOR MODIFIERS
MISSING PAYABLE IN CALL FUNCTION
REVERT STATEMENTS WITHIN EXTERNAL AND PUBLIC FUNCTIONS CAN BE USED TO PERFORM DOS ATTACKS

UNUSED RECEIVE FALLBACK
AVOID RE-STORING VALUES
AVOID ZERO-TO-ONE STORAGE WRITES
CHEAPER CONDITIONAL OPERATORS
CHEAPER INEQUALITIES IN IF()
DEFINE CONSTRUCTOR AS PAYABLE
REVERTING FUNCTIONS CAN BE PAYABLE
GAS INEFFICIENCY DUE TO MULTIPLE OPERANDS IN SINGLE IF/ELSEIF CONDITION
SMALLER DATA TYPES COST MORE
SPLITTING REVERT STATEMENTS
STORAGE VARIABLE CACHING IN MEMORY
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

gmBoost

Uploaded Solidity File(s)

Published on 21 Nov 2025

Language	Audit Methodology	Website
Solidity	**Static Scanning**	**https://gmboost.xyz/**
Publishers/Owner Name	**Organization**	**Contact Email**
gmBoost	**gmBoost**	**github@gmboost.xyz**

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 gmBoost 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:

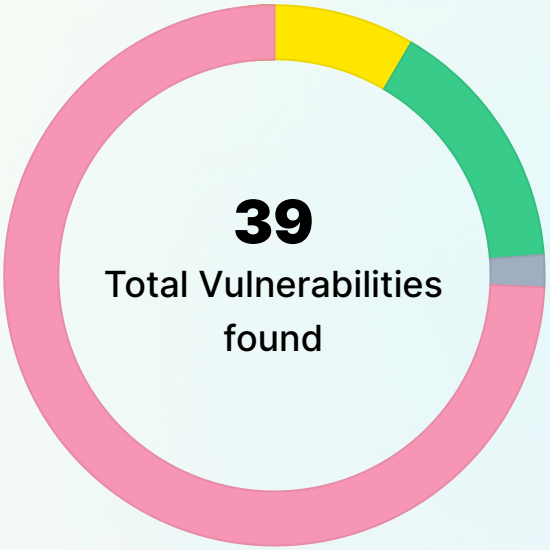
Various common and uncommon attack vectors will be investigated to ensure that the smart contracts are secure from malicious actors. The scanner modules find and flag issues related to Gas optimizations that help in reducing the overall Gas cost It scans and evaluates the codebase against industry best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

03. Findings Summary

gmBoost
File Scan

Security Score	Scan duration	Lines of code
93.69/100	273 secs	362



0	0	4	6	1	28
Crit	High	Med	Low	Info	Gas

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

ACTION TAKEN

0	59	38	3
Fixed	*False Positive*	*Won't Fix*	*Pending Fix*

S. No.	Severity	Bug Type	Instances	Detection Method	Status
C001	Critical	CONTROLLED LOW-LEVEL CALL	2	Automated	*False Positive*
H001	High	MISSING MODIFIER IN INITIALIZE	1	Automated	*False Positive*
M001	Medium	FEE SWITCH OWNER CHECKS MISCONFIGURED	1	SolidityScan AI	*Pending Fix*
M002	Medium	IMMEDIATE DEPLOYFEEWEI UPDATES ENABLE FRONT-RUN/DOS OF DEPLOYMENTS (NO TIMELOCK/GUARDRAILS)	1	SolidityScan AI	*Pending Fix*
M003	Medium	MISSING CONTRACT-CODE VALIDATION FOR GMBOOST_IMPLEMENTATION AND FEE_MANAGER ADDRESSES	1	SolidityScan AI	*Pending Fix*
M004	Medium	RESERVE FACTOR GT 100 PERCENT NOT BLOCKED	1	SolidityScan AI	*False Positive*

S. No.	Severity	Bug Type	Instances	Detection Method	Status
M005	Medium	SWAP FEE UPPER BOUND NOT ENFORCED	1	SolidityScan AI	*Pending Fix*
L001	Low	CHAIN-COMPATIBILITY RISK: CREATE-BASED MINIMAL PROXIES MAY BE UNUSABLE ON SOME L2S (E.G., ZKSYNC)	1	SolidityScan AI	*Pending Fix*
L002	Low	DAO TREASURY SWEEP WITHOUT MULTISIG	1	SolidityScan AI	*Pending Fix*
L003	Low	ERC1155 CALLBACK REENTRANCY	1	SolidityScan AI	*Pending Fix*
L004	Low	ERC777 CALLBACK REENTRANCY	1	SolidityScan AI	*Pending Fix*
L005	Low	INSECURE OWNERSHIP TRANSFER	1	SolidityScan AI	*Pending Fix*
L006	Low	MISSING ONLYOWNER MODIFIER	1	SolidityScan AI	*False Positive*
L007	Low	BALANCE EQUALITY	1	Automated	*False Positive*
L008	Low	EVENT BASED REENTRANCY	2	Automated	*False Positive*
L009	Low	MISSING ZERO ADDRESS VALIDATION	3	Automated	*False Positive*
I001	Informational	MISSING @AUTHOR IN NATSPEC COMMENTS FOR CONTRACT DECLARATION	5	Automated	*False Positive*
I002	Informational	MISSING @DEV IN NATSPEC COMMENTS FOR CONTRACT DECLARATION	3	Automated	*False Positive*
I003	Informational	MISSING @DEV IN NATSPEC COMMENTS FOR FUNCTIONS	7	Automated	*False Positive*
I004	Informational	MISSING INHERITANCE	1	Automated	*False Positive*
I005	Informational	MISSING @INHERITDOC ON OVERRIDE FUNCTIONS	4	Automated	*False Positive*
I006	Informational	MISSING NATSPEC COMMENTS IN SCOPE BLOCKS	3	Automated	*False Positive*

S. No.	Severity	Bug Type	Instances	Detection Method	Status
I007	Informational	MISSING NATSPEC DESCRIPTIONS FOR PUBLIC VARIABLE DECLARATIONS	2	Automated	*False Positive*
I008	Informational	MISSING @NOTICE IN NATSPEC COMMENTS FOR FUNCTIONS	4	Automated	*False Positive*
I009	Informational	MISSING @NOTICE IN NATSPEC COMMENTS FOR MODIFIERS	1	Automated	*False Positive*
I010	Informational	MISSING @PARAM IN NATSPEC COMMENTS FOR MODIFIERS	1	Automated	*False Positive*
I011	Informational	MISSING PAYABLE IN CALL FUNCTION	1	Automated	*False Positive*
I012	Informational	REVERT STATEMENTS WITHIN EXTERNAL AND PUBLIC FUNCTIONS CAN BE USED TO PERFORM DOS ATTACKS	15	Automated	*False Positive*
I013	Informational	UNUSED RECEIVE FALLBACK	1	Automated	*Pending Fix*
G001	Gas	AVOID RE-STORING VALUES	1	Automated	*False Positive*
G002	Gas	AVOID ZERO-TO-ONE STORAGE WRITES	6	Automated	*Pending Fix*
G003	Gas	CHEAPER CONDITIONAL OPERATORS	1	Automated	*Pending Fix*
G004	Gas	CHEAPER INEQUALITIES IN IF()	5	Automated	*Pending Fix*
G005	Gas	DEFINE CONSTRUCTOR AS PAYABLE	3	Automated	*Pending Fix*
G006	Gas	REVERTING FUNCTIONS CAN BE PAYABLE	5	Automated	*Pending Fix*
G007	Gas	GAS INEFFICIENCY DUE TO MULTIPLE OPERANDS IN SINGLE IF/ELSEIF CONDITION	3	Automated	*Pending Fix*

S. No.	Severity	Bug Type	Instances	Detection Method	Status
G008	Gas	SMALLER DATA TYPES COST MORE	1	Automated	*Pending Fix*
G009	Gas	SPLITTING REVERT STATEMENTS	1	Automated	*Pending Fix*
G010	Gas	STORAGE VARIABLE CACHING IN MEMORY	2	Automated	*Pending Fix*
G011	Gas	USE SELFBALANCE() INSTEAD OF ADDRESS(THIS).BALANCE	1	Automated	*Pending Fix*

04. Vulnerability Details

Issue Type

CONTROLLED LOW-LEVEL CALL

S. No.	Severity	Detection Method	Instances
C001	Critical	Automated	2

Description

The contract was using `delegatecall()` or `call()` which was accepting address controlled by a user. This can have de vastating effects on the contract as a delegate call allows the contract to execute code belonging to other contracts but using it's own storage. This can very easily lead to a loss of funds and compromise of the contract.

Bug ID	File Location	Line No.	Action Taken
SSP_114556_55	gmBoost/gmBoost/...mBoost.sol	L110 - L110	*False Positive*
SSP_114556_56	gmBoost/gmBoost/...actory.sol	L89 - L89	*False Positive*

Issue Type

MISSING MODIFIER IN INITIALIZE

S. No.	Severity	Detection Method	Instances
H001	High	Automated	1

Description

The `initialize` function in the contract was found to be missing critical modifiers. This allows a malicious user to call t he `initialize` function multiple times. There are no checks to see if the function has already been initialized. A malicio us owner can abuse that by retroactively changing certain configurations.

Bug ID	File Location	Line No.	Action Taken
SSP_114556_21	gmBoost/gmBoost/...mBoost.sol	L73 - L80	*False Positive*

Issue Type

FEE SWITCH OWNER CHECKS MISCONFIGURED

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

Description

The contract controls a protocol fee switch with an owner-only guard that is implemented incorrectly (e.g., missing o r referencing the wrong storage slot/variable, using tx.origin, or otherwise bypassable). As a result an attacker or any unauthorized caller may toggle fee collection, change fee recipient addresses, or alter fee parameters. This can redir ect or disable protocol revenue and undermine expected economic flows. The bug frequently appears in upgradeabl e/proxy patterns where owner storage layout is inconsistent or when custom access checks are incomplete.

Bug ID	File Location	Line No.	Action Taken
SSP_114556_92	gmBoost/gmBoost/...anager.sol	L105 - L110	*Won't Fix*

Issue Type

IMMEDIATE DEPLOYFEEWEI UPDATES ENABLE FRONT-RUN/DOS OF DEPLOYMENTS (NO TIMELOCK/GUARDRAILS)

S. No.	Severity	Detection Method	Instances
M002	Medium	SolidityScan AI	1

Description

setDeployFeeWei lets the owner change the required deployment fee instantly and without bounds. Per comments, external factory logic will enforce this value. Without a delay, the owner (or compromised key) can front-run pending deployments or toggle fees to disrupt creation flows.

Bug ID	File Location	Line No.	Action Taken
SSP_114556_90	gmBoost/gmBoost/...anager.sol	L134 - L138	*Won't Fix*

Issue Type

MISSING CONTRACT-CODE VALIDATION FOR GMBOOST_IMPLEMENTATION AND FEE_MANAGER ADDRESSES

S. No.	Severity	Detection Method	Instances
M003	Medium	SolidityScan AI	1

Description

The constructor only checks for non-zero addresses but does not verify that gmBoostImpl_ and feeManager_ actually contain contract code. If either address is an EOA or an address without code (or later becomes so via an upgrade/selfdestruct on some chains), createGmBoost will revert when attempting to clone (implementation) or read the deploy config (fee manager). This misconfiguration bricks the factory at deployment time and is not detectable on-chain by the contract beyond the revert during use.

Bug ID	File Location	Line No.	Action Taken
SSP_114556_94	gmBoost/gmBoost/...actory.sol	L59 - L65	*Won't Fix*

Issue Type

RESERVE FACTOR GT 100 PERCENT NOT BLOCKED

S. No.	Severity	Detection Method	Instances
M004	Medium	SolidityScan AI	1

Description

The protocol does not prevent setting the reserve factor higher than 100%, allowing the contract to allocate more tha n 100% of accrued interest to protocol reserves. This breaks accounting invariants: suppliers may receive zero or neg ative net interest and reserve minting logic can attempt to take more tokens than available, leading to reverts or per manent loss/locking of funds. If setter functions lack proper bounds checks on the percentage/mantissa, an attacker or compromised admin can redirect all interest to reserves or trigger arithmetic errors during accrual and distribution.

Bug ID	File Location	Line No.	Action Taken
SSP_114556_98	gmBoost/gmBoost/...mBoost.sol	L90 - L118	*False Positive*

Issue Type

SWAP FEE UPPER BOUND NOT ENFORCED

S. No.	Severity	Detection Method	Instances
M005	Medium	SolidityScan AI	1

Description

The contract allows the swap fee parameter to be changed without validating an explicit maximum value. A malicious or compromised privileged actor can set an arbitrarily large fee (including 100% or larger), which can steal user fund s, break expected pricing, or render swaps unusable. Lack of an upper bound also enables governance or controller misconfiguration to cause permanent economic loss or denial of service for traders. This is an economic-critical confi guration validation bug affecting user funds and protocol integrity.

Bug ID	File Location	Line No.	Action Taken
SSP_114556_89	gmBoost/gmBoost/...anager.sol	L115 - L119	*Won't Fix*

Issue Type

CHAIN-COMPATIBILITY RISK: CREATE-BASED MINIMAL PROXIES MAY BE UNUSABLE ON SOME L2S (E.G., ZKSYNC)

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

Description

The factory uses OpenZeppelin Clones.clone() which relies on the CREATE opcode. Certain L2s (e.g., zkSync Era) have differences in CREATE semantics for arbitrary bytecode that can cause deployment failures for clone patterns depending on the library and chain. If clone deployment fails on such a chain, createGmBoost will revert and the protocol becomes unusable there. While funds are not lost (fee forwarding happens after clone), the function's availability is impacted on affected chains.

Bug ID	File Location	Line No.	Action Taken
SSP_114556_96	gmBoost/gmBoost/...actory.sol	L71 - L96	*Won't Fix*

Issue Type

DAO TREASURY SWEEP WITHOUT MULTISIG

S. No.	Severity	Detection Method	Instances
L002	Low	SolidityScan AI	1

Description

A treasury sweep or recovery function that can be executed by a single privileged account or owner without multisig nature or governance checks allows immediate unilateral withdrawal of protocol assets. If the single key is compromi sed or malicious, attacker can drain ETH/tokens from the DAO treasury. The contract typically lacks timelock, propos al verification or signature thresholds that would provide notice or require consent from multiple trustees. This weakn ess results in high-confidence, irreversible loss of user funds and undermines decentralization guarantees.

Bug ID	File Location	Line No.	Action Taken
SSP_114556_95	gmBoost/gmBoost/...actory.sol	L71 - L96	*Won't Fix*

Issue Type

ERC1155 CALLBACK REENTRANCY

S. No.	Severity	Detection Method	Instances
L003	Low	SolidityScan AI	1

Description

A reentrancy flaw that arises when ERC\u20111155 safeTransferFrom/safeBatchTransferFrom invoke the recipient hook (onERC1155Received/onERC1155BatchReceived). If the token contract performs external calls to the recipient before fully updating critical state (balances, approvals, totals) or lacks reentrancy protection, a malicious receiver can call back into the token contract and manipulate state, perform additional transfers, or bypass checks. This may lead to unauthorized token theft, double-spends, or state corruption across token accounting and business logic.

Bug ID	File Location	Line No.	Action Taken
SSP_114556_97	gmBoost/gmBoost/...actory.sol	L71 - L96	*Won't Fix*

Issue Type

ERC777 CALLBACK REENTRANCY

S. No.	Severity	Detection Method	Instances
L004	Low	SolidityScan AI	1

Description

ERC777 tokens invoke tokensToSend/tokensReceived callbacks on sender and recipient during transfers, allowing th e callee to execute arbitrary code. If a contract performs ERC777 transfers or calls external addresses before finishin g critical state updates, the recipient can reenter the contract and manipulate state or drain assets. This reentrancy v ector breaks assumptions made for ERC20-style transfers and can lead to loss of funds or corrupted invariants.

Bug ID	File Location	Line No.	Action Taken
SSP_114556_100	gmBoost/gmBoost/...mBoost.sol	L90 - L118	*Won't Fix*

Issue Type

INSECURE OWNERSHIP TRANSFER

S. No.	Severity	Detection Method	Instances
L005	Low	SolidityScan AI	1

Description

The contract transfers ownership via direct assignment or single-call transfer without an explicit acceptance step or sufficient validation. Missing checks (e.g., non-zero address) and lack of a two-step claim pattern can lead to accide ntal renouncement, assignment to an inaccessible address, or unauthorized takeover if the initiator is tricked or a bu g is exploited. Absence of transfer events and insufficient access restrictions increase auditability and recovery diffic ulty. This weakness endangers control over privileged functions and funds managed by the contract.

Bug ID	File Location	Line No.	Action Taken
SSP_114556_93	gmBoost/gmBoost/...anager.sol	L95 - L100	*Won't Fix*

Issue Type

MISSING ONLYOWNER MODIFIER

S. No.	Severity	Detection Method	Instances
L006	Low	SolidityScan AI	1

Description

Functions that are intended to be restricted to the contract owner lack an onlyOwner-style access modifier, so any e xternal account can invoke them. This enables attackers to perform unauthorized administrative actions such as cha nging parameters, minting tokens, withdrawing funds, upgrading logic, or disabling protections. The problem typicall y arises when code reads an owner variable (or assumes an owner) but never enforces msg.sender == owner on sen sitive functions. Without enforced access control, critical invariants and funds can be compromised.

Bug ID	File Location	Line No.	Action Taken
SSP_114556_91	gmBoost/gmBoost/...anager.sol	L124 - L129	*False Positive*

Issue Type

BALANCE EQUALITY

S. No.	Severity	Detection Method	Instances
L007	Low	Automated	1

Description

A common defensive programming technique that is useful in enforcing correct state transitions or validating operati ons is invariant-checking.
In particular, there is one apparent invariant that may be tempting to use but can in fact be manipulated by external u sers (regardless of the rules put in place in the smart contract). This is the current ether stored in the contract. (`this. balance`)
There are two ways in which malicious actors can (forcibly) send Ether to a contract without using a payable function or executing any code on the contract therefore compromising the contract logic depending on `this.balance` .
These are `selfdestruct/suicide` and pre-loading the contract with Ether.

Bug ID	File Location	Line No.	Action Taken
SSP_114556_38	gmBoost/gmBoost/...mBoost.sol	L124 - L124	*False Positive*

Issue Type

EVENT BASED REENTRANCY

S. No.	Severity	Detection Method	Instances
L008	Low	Automated	2

Description

In a Re-entrancy attack, a malicious contract calls back into the calling contract before the first invocation of the fun ction is finished. This may cause the different invocations of the function to interact in undesirable ways, especially i n 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 cal ls.

Bug ID	File Location	Line No.	Action Taken
SSP_114556_16	gmBoost/gmBoost/...mBoost.sol	L90 - L118	*False Positive*
SSP_114556_17	gmBoost/gmBoost/...actory.sol	L71 - L96	*False Positive*

Issue Type

MISSING ZERO ADDRESS VALIDATION

S. No.	Severity	Detection Method	Instances
L009	Low	Automated	3

Description

The contract is found to lack proper validation for zero address inputs in critical functions, particularly where an 'addr ess' input is expected, like during initialization or setting ownership. Failure to validate zero addresses can result in u nwanted contract states, such as having critical roles or operations reference the zero address (0×0), which is univer sally considered an invalid and non-operational address within Ethereum, potentially leading to the loss of assets or control over the contract's functionality.

Bug ID	File Location	Line No.	Action Taken
SSP_114556_27	gmBoost/gmBoost/...anager.sol	L62 - L83	*False Positive*
SSP_114556_28	gmBoost/gmBoost/...mBoost.sol	L73 - L80	*False Positive*
SSP_114556_29	gmBoost/gmBoost/...actory.sol	L59 - L65	*False Positive*

Issue Type

MISSING @AUTHOR IN NATSPEC COMMENTS FOR CONTRACT DECLARATION

S. No.	Severity	Detection Method	Instances
I001	Informational	Automated	5

Description

Natspec `@author` tags are missing from contract declarations in the code. This reduces code clarity and makes it difficult to determine the original author or organization responsible for the contract. The absence of an `@author` tag can lead to challenges in tracking ownership, verifying authenticity, and maintaining proper documentation.

Bug ID	File Location	Line No.	Action Taken
SSP_114556_64	gmBoost/gmBoost/...anager.sol	L7 - L24	*False Positive*
SSP_114556_65	gmBoost/gmBoost/...anager.sol	L10 - L158	*False Positive*
SSP_114556_66	gmBoost/gmBoost/...mBoost.sol	L11 - L135	*False Positive*
SSP_114556_67	gmBoost/gmBoost/...actory.sol	L10 - L15	*False Positive*
SSP_114556_68	gmBoost/gmBoost/...actory.sol	L21 - L97	*False Positive*

Issue Type

MISSING @DEV IN NATSPEC COMMENTS FOR CONTRACT DECLARATION

S. No.	Severity	Detection Method	Instances
I002	Informational	Automated	3

Description

Natspec `@dev` tags are missing from contract declarations in the code. This reduces code clarity and makes it difficult for developers to understand the contract's internal logic, implementation details, and potential caveats.

Bug ID	File Location	Line No.	Action Taken
SSP_114556_86	gmBoost/gmBoost/...anager.sol	L7 - L24	*False Positive*
SSP_114556_87	gmBoost/gmBoost/...mBoost.sol	L11 - L135	*False Positive*
SSP_114556_88	gmBoost/gmBoost/...actory.sol	L10 - L15	*False Positive*

Issue Type

MISSING @DEV IN NATSPEC COMMENTS FOR FUNCTIONS

S. No.	Severity	Detection Method	Instances
I003	Informational	Automated	7

Description

Natspec `@dev` comments are missing from a function declaration in the code. The `@dev` tag provides a detailed ex planation of a function's purpose, assumptions, and behavior. Lack of `@dev` documentation can: reduce code maint ainability, making it harder for developers to understand the function's implementation details.

Bug ID	File Location	Line No.	Action Taken
SSP_114556_75	gmBoost/gmBoost/...anager.sol	L12 - L15	*False Positive*
SSP_114556_76	gmBoost/gmBoost/...anager.sol	L20 - L23	*False Positive*
SSP_114556_77	gmBoost/gmBoost/...anager.sol	L142 - L148	*False Positive*
SSP_114556_78	gmBoost/gmBoost/...anager.sol	L151 - L157	*False Positive*
SSP_114556_79	gmBoost/gmBoost/...mBoost.sol	L121 - L130	*False Positive*
SSP_114556_80	gmBoost/gmBoost/...mBoost.sol	L134 - L134	*False Positive*
SSP_114556_81	gmBoost/gmBoost/...actory.sol	L14 - L14	*False Positive*

Issue Type

MISSING INHERITANCE

S. No.	Severity	Detection Method	Instances
I004	Informational	Automated	1

Description

The contract probably intends to inherit the interface but does not do so by deriving from it. This suggests missing lo gic or dead code.

Bug ID	File Location	Line No.	Action Taken
SSP_114556_19	gmBoost/gmBoost/...mBoost.sol	L11 - L135	*False Positive*

Issue Type

MISSING @INHERITDOC ON OVERRIDE FUNCTIONS

S. No.	Severity	Detection Method	Instances
I005	Informational	Automated	4

Description

Overridden functions in Solidity should include the `@inheritdoc` tag in their NatSpec comments to ensure documentation consistency and clarity. Failing to include `@inheritdoc` can lead to incomplete or missing documentation, making it difficult for developers and auditors to understand the function's intended behavior. This lack of clarity can increase maintenance overhead and potentially lead to misinterpretations of contract logic.

Bug ID	File Location	Line No.	Action Taken
SSP_114556_71	gmBoost/gmBoost/...anager.sol	L12 - L15	*False Positive*
SSP_114556_72	gmBoost/gmBoost/...anager.sol	L142 - L148	*False Positive*
SSP_114556_73	gmBoost/gmBoost/...anager.sol	L151 - L157	*False Positive*
SSP_114556_74	gmBoost/gmBoost/...actory.sol	L14 - L14	*False Positive*

Issue Type

MISSING NATSPEC COMMENTS IN SCOPE BLOCKS

S. No.	Severity	Detection Method	Instances
I006	Informational	Automated	3

Description

NatSpec comments are missing from an unnamed scope block. Scope blocks can group computations or temporary l ogic and should be documented to clarify purpose and expected behavior. Lack of NatSpec documentation can redu ce readability, making it hard to understand why the scope block exists.

Bug ID	File Location	Line No.	Action Taken
SSP_114556_33	gmBoost/gmBoost/...mBoost.sol	L107 - L112	*False Positive*
SSP_114556_34	gmBoost/gmBoost/...actory.sol	L60 - L62	*False Positive*
SSP_114556_35	gmBoost/gmBoost/...actory.sol	L86 - L91	*False Positive*

Issue Type

MISSING NATSPEC DESCRIPTIONS FOR PUBLIC VARIABLE DECLARATIONS

S. No.	Severity	Detection Method	Instances
I007	Informational	Automated	2

Description

Public variable declarations without NatSpec descriptions are found in the code. NatSpec comments improve readabi lity and provide clear documentation for externally accessible variables. Without them, developers and users may str uggle to understand the role and impact of these variables when interacting with the contract.

Bug ID	File Location	Line No.	Action Taken
SSP_114556_84	gmBoost/gmBoost/...anager.sol	L16 - L16	*False Positive*
SSP_114556_85	gmBoost/gmBoost/...mBoost.sol	L28 - L28	*False Positive*

Issue Type

MISSING @NOTICE IN NATSPEC COMMENTS FOR FUNCTIONS

S. No.	Severity	Detection Method	Instances
I008	Informational	Automated	4

Description

Natspec `@notice` comments are missing from a function declaration in the code. The `@notice` tag provides an over view of what the function does, helping users and developers quickly understand its purpose.

Bug ID	File Location	Line No.	Action Taken
SSP_114556_49	gmBoost/gmBoost/...anager.sol	L12 - L15	*False Positive*
SSP_114556_50	gmBoost/gmBoost/...anager.sol	L142 - L148	*False Positive*
SSP_114556_51	gmBoost/gmBoost/...anager.sol	L151 - L157	*False Positive*
SSP_114556_52	gmBoost/gmBoost/...actory.sol	L14 - L14	*False Positive*

Issue Type

MISSING @NOTICE IN NATSPEC COMMENTS FOR MODIFIERS

S. No.	Severity	Detection Method	Instances
I009	Informational	Automated	1

Description

Natspec `@notice` comments are missing from a modifier declaration in the code. The `@notice` tag provides a high-level description of a modifier's purpose, making it easier to understand its function. Lack of `@notice` documentation can reduce code readability, making it harder to understand restrictions applied by modifiers.

Bug ID	File Location	Line No.	Action Taken
SSP_114556_45	gmBoost/gmBoost/...anager.sol	L86 - L89	*False Positive*

Issue Type

MISSING @PARAM IN NATSPEC COMMENTS FOR MODIFIERS

S. No.	Severity	Detection Method	Instances
I010	Informational	Automated	1

Description

Natspec `@param` comments are missing from a modifier declaration in the code. This reduces clarity, making it harder to understand the purpose of modifier parameters, their expected values, and their role in contract execution.

Bug ID	File Location	Line No.	Action Taken
SSP_114556_18	gmBoost/gmBoost/...anager.sol	L86 - L89	*False Positive*

Issue Type

MISSING PAYABLE IN CALL FUNCTION

S. No.	Severity	Detection Method	Instances
I011	Informational	Automated	1

Description

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

Bug ID	File Location	Line No.	Action Taken
SSP_114556_54	gmBoost/gmBoost/...mBoost.sol	L127 - L127	*False Positive*

Issue Type

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

S. No.	Severity	Detection Method	Instances
I012	Informational	Automated	15

Description

In public and external functions, improper use of `revert` can be exploited for Denial of Service (DoS) attacks. An attacker can intentionally trigger these 'revert' conditions, causing legitimate transactions to consistently fail. For example, if a function relies on specific conditions from user input or contract state, an attacker could manipulate these to continually force reverts, blocking the function's execution.

Bug ID	File Location	Line No.	Action Taken
SSP_114556_1	gmBoost/gmBoost/...anager.sol	L96 - L96	*False Positive*
SSP_114556_2	gmBoost/gmBoost/...anager.sol	L106 - L106	*False Positive*
SSP_114556_3	gmBoost/gmBoost/...anager.sol	L125 - L125	*False Positive*
SSP_114556_4	gmBoost/gmBoost/...mBoost.sol	L74 - L74	*False Positive*
SSP_114556_5	gmBoost/gmBoost/...mBoost.sol	L75 - L75	*False Positive*
SSP_114556_6	gmBoost/gmBoost/...mBoost.sol	L76 - L76	*False Positive*
SSP_114556_7	gmBoost/gmBoost/...mBoost.sol	L96 - L96	*False Positive*
SSP_114556_8	gmBoost/gmBoost/...mBoost.sol	L99 - L99	*False Positive*

Bug ID	File Location	Line No.	Action Taken
SSP_114556_9	gmBoost/gmBoost/...mBoost.sol	L111 - L111	*False Positive*
SSP_114556_10	gmBoost/gmBoost/...mBoost.sol	L122 - L122	*False Positive*
SSP_114556_11	gmBoost/gmBoost/...mBoost.sol	L124 - L124	*False Positive*
SSP_114556_12	gmBoost/gmBoost/...mBoost.sol	L128 - L128	*False Positive*
SSP_114556_13	gmBoost/gmBoost/...actory.sol	L76 - L76	*False Positive*
SSP_114556_14	gmBoost/gmBoost/...actory.sol	L79 - L79	*False Positive*
SSP_114556_15	gmBoost/gmBoost/...actory.sol	L90 - L90	*False Positive*

Issue Type

UNUSED RECEIVE FALLBACK

S. No.	Severity	Detection Method	Instances
I013	Informational	Automated	1

Description

The contract was found to be defining an empty 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.

Bug ID	File Location	Line No.	Action Taken
SSP_114556_20	gmBoost/gmBoost/...mBoost.sol	L134 - L134	*Won't Fix*

Issue Type

AVOID RE-STORING VALUES

S. No.	Severity	Detection Method	Instances
G001	Gas	Automated	1

Description

The function is found to be allowing re-storing the value in the contract's state variable even when the old value is equal to the new value. This practice results in unnecessary gas consumption due to the `Gsreset` operation (2900 gas), which could be avoided. If the old value and the new value are the same, not updating the storage would avoid this 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_114556_39	gmBoost/gmBoost/...mBoost.sol	L73 - L80	*False Positive*

Issue Type

AVOID ZERO-TO-ONE STORAGE WRITES

S. No.	Severity	Detection Method	Instances
G002	Gas	Automated	6

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 access), making it one of the most expensive operations. This is why patterns like OpenZeppelin's `ReentrancyGuard` use `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_114556_57	gmBoost/gmBoost/...anager.sol	L74 - L74	*Won't Fix*
SSP_114556_58	gmBoost/gmBoost/...anager.sol	L75 - L75	*Won't Fix*
SSP_114556_59	gmBoost/gmBoost/...anager.sol	L76 - L76	*Won't Fix*
SSP_114556_60	gmBoost/gmBoost/...anager.sol	L117 - L117	*Won't Fix*
SSP_114556_61	gmBoost/gmBoost/...anager.sol	L127 - L127	*Won't Fix*
SSP_114556_62	gmBoost/gmBoost/...anager.sol	L136 - L136	*Won't Fix*

Issue Type

CHEAPER CONDITIONAL OPERATORS

S. No.	Severity	Detection Method	Instances
G003	Gas	Automated	1

Description

During compilation, `x != 0` is cheaper than `x > 0` for unsigned integers in solidity inside conditional statements.

Bug ID	File Location	Line No.	Action Taken
SSP_114556_37	gmBoost/gmBoost/...actory.sol	L86 - L86	*Won't Fix*

Issue Type

CHEAPER INEQUALITIES IN IF()

S. No.	Severity	Detection Method	Instances
G004	Gas	Automated	5

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_114556_40	gmBoost/gmBoost/...anager.sol	L70 - L70	*Won't Fix*
SSP_114556_41	gmBoost/gmBoost/...anager.sol	L125 - L125	*Won't Fix*
SSP_114556_42	gmBoost/gmBoost/...mBoost.sol	L99 - L99	*Won't Fix*
SSP_114556_43	gmBoost/gmBoost/...actory.sol	L79 - L79	*Won't Fix*
SSP_114556_44	gmBoost/gmBoost/...actory.sol	L86 - L86	*Won't Fix*

Issue Type

DEFINE CONSTRUCTOR AS PAYABLE

S. No.	Severity	Detection Method	Instances
G005	Gas	Automated	3

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.

Bug ID	File Location	Line No.	Action Taken
SSP_114556_30	gmBoost/gmBoost/...anager.sol	L62 - L83	*Won't Fix*
SSP_114556_31	gmBoost/gmBoost/...mBoost.sol	L56 - L60	*Won't Fix*
SSP_114556_32	gmBoost/gmBoost/...actory.sol	L59 - L65	*Won't Fix*

Issue Type

REVERTING FUNCTIONS CAN BE PAYABLE

S. No.	Severity	Detection Method	Instances
G006	Gas	Automated	5

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_114556_22	gmBoost/gmBoost/...anager.sol	L95 - L100	*Won't Fix*
SSP_114556_23	gmBoost/gmBoost/...anager.sol	L105 - L110	*Won't Fix*
SSP_114556_24	gmBoost/gmBoost/...anager.sol	L115 - L119	*Won't Fix*
SSP_114556_25	gmBoost/gmBoost/...anager.sol	L124 - L129	*Won't Fix*
SSP_114556_26	gmBoost/gmBoost/...anager.sol	L134 - L138	*Won't Fix*

Issue Type

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

S. No.	Severity	Detection Method	Instances
G007	Gas	Automated	3

Description

The contract is found to use multiple operands within a single `if` or `else if` statement, which can lead to unnecessary 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 result in more complex bytecode and higher gas usage compared to using nested `if` statements. This inefficiency is particularly relevant in functions that are called frequently or within loops.

Bug ID	File Location	Line No.	Action Taken
SSP_114556_46	gmBoost/gmBoost/...anager.sol	L69 - L69	*Won't Fix*
SSP_114556_47	gmBoost/gmBoost/...mBoost.sol	L76 - L76	*Won't Fix*
SSP_114556_48	gmBoost/gmBoost/...actory.sol	L60 - L62	*Won't Fix*

Issue Type

SMALLER DATA TYPES COST MORE

S. No.	Severity	Detection Method	Instances
G008	Gas	Automated	1

Description

Usage of smaller integer types such as `uint8`, `uint16`, `int8`, or `int16` in arithmetic operations incur additional gas costs compared to the default `uint` and `int` types, which are typically `uint256` and `int256` respectively.

Bug ID	File Location	Line No.	Action Taken
SSP_114556_53	gmBoost/gmBoost/...mBoost.sol	L103 - L103	*Won't Fix*

Issue Type

SPLITTING REVERT STATEMENTS

S. No.	Severity	Detection Method	Instances
G009	Gas	Automated	1

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_114556_36	gmBoost/gmBoost/...actory.sol	L60 - L62	*Won't Fix*

Issue Type

STORAGE VARIABLE CACHING IN MEMORY

S. No.	Severity	Detection Method	Instances
G010	Gas	Automated	2

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_114556_69	gmBoost/gmBoost/...mBoost.sol	L121 - L130	*Won't Fix*
SSP_114556_70	gmBoost/gmBoost/...actory.sol	L71 - L96	*Won't Fix*

Issue Type

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

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

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_114556_63	gmBoost/gmBoost/...mBoost.sol	L123 - L123	*Won't Fix*

05. Scan History

Critical High Medium Low Informational Gas

No	Date	Security Score	Scan Overview
1.	2025-11-21	93.69	0 0 4 6 1 28

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.