SolidityScan | Powered by CRED SHIELDS

Security Assessment Report

InfinityName

18 Oct 2025

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

Table of Contents

RESCUE TOKEN FUNCTION UNSAFE
EVENT BASED REENTRANCY
USE OF FLOATING PRAGMA
MISSING EVENTS
NONREENTRANT MODIFIER PLACEMENT
OUTDATED COMPILER VERSION
USE OWNABLE2STEP
HARD-CODED ADDRESS DETECTED
AVOID ARITHMETIC DIRECTLY WITHIN ARRAY INDICES
CONTRACT CALLS DISABLEINITIALIZERS IN IT'S CONSTRUCTOR BUT ALSO CONTAINS A INITIALIZATION FUNCTION WHICH UTILIZES THE INITIALIZER MODIFIER
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 INDEXED KEYWORDS IN EVENTS
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
NAME MAPPING PARAMETERS
RETURN INSIDE LOOP
REVERT STATEMENTS WITHIN EXTERNAL AND PUBLIC FUNCTIONS CAN BE USED TO PERFORM DOS ATTACKS
UNNAMED FUNCTION PARAMETERS
ABI ENCODE IS LESS EFFICIENT THAN ABI ENCODEPACKED
ARRAY LENGTH CACHING
AVOID RE-STORING VALUES
AVOID ZERO-TO-ONE STORAGE WRITES
CHEAPER CONDITIONAL OPERATORS
CHEAPER INEQUALITIES IN IF()
DEFAULT INT VALUES ARE MANUALLY RESET
DEFINE CONSTRUCTOR AS PAYABLE
FUNCTIONS CAN BE IN-LINED
REVERTING FUNCTIONS CAN BE PAYABLE

GAS INEFFICIENCY DUE TO MULTIPLE OPERANDS IN SINGLE IF/ELSEIF CONDITION
GAS OPTIMIZATION IN INCREMENTS
HARDCODED GAS IN FUNCTION CALLS
INTERNAL FUNCTIONS NEVER USED
LONG NUMBER LITERALS
NAMED RETURN OF LOCAL VARIABLE SAVES GAS AS COMPARED TO RETURN STATEMENT
OPTIMIZING ADDRESS ID MAPPING
PUBLIC CONSTANTS CAN BE PRIVATE
STORAGE VARIABLE CACHING IN MEMORY
STORING STORAGE VARIABLES IN MEMORY
UNNECESSARY CHECKED ARITHMETIC IN LOOP
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

InfinityName

Github Project
https://github.com/infinitynamecom/InfinityName

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 InfinityName 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:

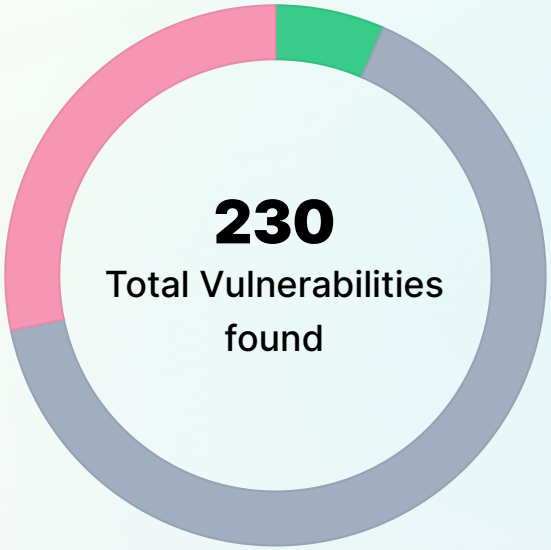
Various common and uncommon attack vectors will be investigated to ensure that the smart contracts are secure from malicious actors. The scanner modules find and flag issues related to Gas optimizations that help in reducing the overall Gas cost It scans and evaluates the codebase against industry best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

03. Findings Summary

InfinityName
View on Github

Security Score	Scan duration	Lines of code
91.63/100	**340 secs**	**592**



0	0	0	15	150	65
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	11	0	235
Fixed	*False Positive*	*Won't Fix*	*Pending Fix*

S. No.	Severity	Bug Type	Instances	Detection Method	Status
C001	Critical	CONTROLLED LOW-LEVEL CALL	4	Automated	*False Positive*
H001	High	REENTRANCY	1	Automated	*False Positive*
H002	High	WITHDRAWAL QUEUE ORDERING BUGS	1	SolidityScan AI	*False Positive*
M001	Medium	PERMIT2 MISUSE WRONG DOMAIN SEPARATOR	1	SolidityScan AI	*False Positive*
M002	Medium	ACCOUNT EXISTENCE CHECK FOR LOW LEVEL CALLS	1	Automated	*False Positive*

S. No.	Severity	Bug Type	Instances	Detection Method	Status
M003	Medium	SUPPORTSINTERFACE() CALLS MAY REVERT	1	Automated	*False Positive*
M004	Medium	NFT APPROVAL MISCONFIGURATION	1	Automated	*False Positive*
M005	Medium	UNPROTECTED ETHER WITHDRAWAL	1	SolidityScan AI	*False Positive*
M006	Medium	UNVETTED REFERRER ALLOWS UNIVERSAL DISCOUNT AND REFERRAL SIPHONING	1	SolidityScan AI	*False Positive*
L001	Low	ERC777 CALLBACK REENTRANCY	1	SolidityScan AI	*Pending Fix*
L002	Low	GETPENDINGWITHDRAWAL HIDES BALANCES FOR NON-CURRENT FEERECIPIENT	1	SolidityScan AI	*Pending Fix*
L003	Low	RESCUE TOKEN FUNCTION UNSAFE	1	SolidityScan AI	*Pending Fix*
L004	Low	EVENT BASED REENTRANCY	1	Automated	*Pending Fix*
L005	Low	USE OF FLOATING PRAGMA	2	Automated	*Pending Fix*
L006	Low	MISSING EVENTS	4	Automated	*Pending Fix*
L007	Low	NONREENTRANT MODIFIER PLACEMENT	2	Automated	*Pending Fix*
L008	Low	OUTDATED COMPILER VERSION	2	Automated	*Pending Fix*
L009	Low	USE OWNABLE2STEP	1	Automated	*Pending Fix*
I001	Informational	HARD-CODED ADDRESS DETECTED	1	Automated	*Pending Fix*
I002	Informational	AVOID ARITHMETIC DIRECTLY WITHIN ARRAY INDICES	1	Automated	*Pending Fix*
I003	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
I004	Informational	CONTRACT NAME SHOULD USE PASCALCASE	1	Automated	*Pending Fix*
I005	Informational	MISSING @AUTHOR IN NATSPEC COMMENTS FOR CONTRACT DECLARATION	2	Automated	*Pending Fix*
I006	Informational	MISSING @DEV IN NATSPEC COMMENTS FOR CONTRACT DECLARATION	2	Automated	*Pending Fix*
I007	Informational	MISSING @DEV IN NATSPEC COMMENTS FOR FUNCTIONS	3	Automated	*Pending Fix*
I008	Informational	MISSING INDEXED KEYWORDS IN EVENTS	2	Automated	*Pending Fix*
I009	Informational	MISSING @INHERITDOC ON OVERRIDE FUNCTIONS	33	Automated	*Pending Fix*
I010	Informational	MISSING NATSPEC COMMENTS IN SCOPE BLOCKS	26	Automated	*Pending Fix*
I011	Informational	MISSING NATSPEC DESCRIPTIONS FOR PUBLIC VARIABLE DECLARATIONS	12	Automated	*Pending Fix*
I012	Informational	MISSING @NOTICE IN NATSPEC COMMENTS FOR CONSTRUCTORS	1	Automated	*Pending Fix*
I013	Informational	MISSING @NOTICE IN NATSPEC COMMENTS FOR FUNCTIONS	36	Automated	*Pending Fix*
I014	Informational	MISSING PAYABLE IN CALL FUNCTION	3	Automated	*Pending Fix*
I015	Informational	MISSING UNDERSCORE IN NAMING VARIABLES	8	Automated	*Pending Fix*
I016	Informational	NAME MAPPING PARAMETERS	7	Automated	*Pending Fix*

S. No.	Severity	Bug Type	Instances	Detection Method	Status
I017	Informational	RETURN INSIDE LOOP	1	Automated	*Pending Fix*
I018	Informational	REVERT STATEMENTS WITHIN EXTERNAL AND PUBLIC FUNCTIONS CAN BE USED TO PERFORM DOS ATTACKS	11	Automated	*Pending Fix*
I019	Informational	UNNAMED FUNCTION PARAMETERS	1	Automated	*Pending Fix*
G001	Gas	ABI ENCODE IS LESS EFFICIENT THAN ABI ENCODEPACKED	1	Automated	*Pending Fix*
G002	Gas	ARRAY LENGTH CACHING	1	Automated	*Pending Fix*
G003	Gas	AVOID RE-STORING VALUES	3	Automated	*Pending Fix*
G004	Gas	AVOID ZERO-TO-ONE STORAGE WRITES	4	Automated	*Pending Fix*
G005	Gas	CHEAPER CONDITIONAL OPERATORS	5	Automated	*Pending Fix*
G006	Gas	CHEAPER INEQUALITIES IN IF()	6	Automated	*Pending Fix*
G007	Gas	DEFAULT INT VALUES ARE MANUALLY RESET	4	Automated	*Pending Fix*
G008	Gas	DEFINE CONSTRUCTOR AS PAYABLE	1	Automated	*Pending Fix*
G009	Gas	FUNCTIONS CAN BE IN-LINED	5	Automated	*Pending Fix*
G010	Gas	REVERTING FUNCTIONS CAN BE PAYABLE	6	Automated	*Pending Fix*
G011	Gas	GAS INEFFICIENCY DUE TO MULTIPLE OPERANDS IN SINGLE IF/ELSEIF CONDITION	6	Automated	*Pending Fix*
G012	Gas	GAS OPTIMIZATION IN INCREMENTS	2	Automated	*Pending Fix*
G013	Gas	HARDCODED GAS IN FUNCTION CALLS	1	Automated	*Pending Fix*
G014	Gas	INTERNAL FUNCTIONS NEVER USED	1	Automated	*Pending Fix*

S. No.	Severity	Bug Type	Instances	Detection Method	Status
G015	Gas	LONG NUMBER LITERALS	1	Automated	*Pending Fix*
G016	Gas	NAMED RETURN OF LOCAL VARIABLE SAVES GAS AS COMPARED TO RETURN STATEMENT	1	Automated	*Pending Fix*
G017	Gas	OPTIMIZING ADDRESS ID MAPPING	3	Automated	*Pending Fix*
G018	Gas	PUBLIC CONSTANTS CAN BE PRIVATE	4	Automated	*Pending Fix*
G019	Gas	STORAGE VARIABLE CACHING IN MEMORY	9	Automated	*Pending Fix*
G020	Gas	STORING STORAGE VARIABLES IN MEMORY	2	Automated	*Pending Fix*
G021	Gas	UNNECESSARY CHECKED ARITHMETIC IN LOOP	2	Automated	*Pending Fix*
G022	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	4

Bug ID	File Location	Line No.	Action Taken
SSP_109224_14	--	--	*False Positive*
SSP_109224_15	--	--	*False Positive*
SSP_109224_16	--	--	*False Positive*
SSP_109224_16	--	--	*False Positive*

Upgrade your Plan to view the full report

4 Critical Issues Found

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

Upgrade

Issue Type

REENTRANCY

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

Bug ID	File Location	Line No.	Action Taken
SSP_109224_194	--	--	*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

PERMIT2 MISUSE WRONG DOMAIN SEPARATOR

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

Bug ID	File Location	Line No.	Action Taken
SSP_109224_241	--	--	*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

ERC777 CALLBACK REENTRANCY

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

Bug ID	File Location	Line No.	Action Taken
SSP_109224_245	--	--	*Pending Fix*

Issue Type

HARD-CODED ADDRESS DETECTED

S. No.	Severity	Detection Method	Instances
I001	Informational	Automated	1

Bug ID	File Location	Line No.	Action Taken
SSP_109224_199	--	--	*Pending Fix*

Upgrade your Plan to view the full report

1 Informational Issues Found

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

Upgrade

Issue Type

ABI ENCODE IS LESS EFFICIENT THAN ABI ENCODEPACKED

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

Description

The contract is using `abi.encode()` in the function. In `abi.encode()`, all elementary types are padded to 32 bytes and dynamic arrays include their length, whereas `abi.encodePacked()` will only use the minimal required memory to enco de the data.

Bug ID	File Location	Line No.	Action Taken
SSP_109224_213	InfinityNameUpgr...deable.sol	L560 - L560	*Pending Fix*

Issue Type

ARRAY LENGTH CACHING

S. No.	Severity	Detection Method	Instances
G002	Gas	Automated	1

Description

During each iteration of the loop, reading the length of the array uses more gas than is necessary. In the most favora ble scenario, in which the length is read from a memory variable, storing the array length in the stack can save about 3 gas per iteration. In the least favorable scenario, in which external calls are made during each iteration, the amount of gas wasted can be significant.

Bug ID	File Location	Line No.	Action Taken
SSP_109224_178	InfinityNameUpgr...deable.sol	L282 - L284	*Pending Fix*

Issue Type

AVOID RE-STORING VALUES

S. No.	Severity	Detection Method	Instances
G003	Gas	Automated	3

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_109224_11	InfinityNameUpgr...deable.sol	L206 - L212	*Pending Fix*
SSP_109224_12	InfinityNameUpgr...deable.sol	L471 - L475	*Pending Fix*
SSP_109224_13	InfinityNameUpgr...deable.sol	L566 - L571	*Pending Fix*

Issue Type

AVOID ZERO-TO-ONE STORAGE WRITES

S. No.	Severity	Detection Method	Instances
G004	Gas	Automated	4

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_109224_225	InfinityNameUpgr...deable.sol	L111 - L111	*Pending Fix*
SSP_109224_226	InfinityNameUpgr...deable.sol	L112 - L112	*Pending Fix*
SSP_109224_227	InfinityNameUpgr...deable.sol	L371 - L371	*Pending Fix*
SSP_109224_228	InfinityNameUpgr...deable.sol	L473 - L473	*Pending Fix*

Issue Type

CHEAPER CONDITIONAL OPERATORS

S. No.	Severity	Detection Method	Instances
G005	Gas	Automated	5

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_109224_18	InfinityNameUpgr...deable.sol	L388 - L388	*Pending Fix*
SSP_109224_19	InfinityNameUpgr...deable.sol	L422 - L422	*Pending Fix*
SSP_109224_20	InfinityNameUpgr...deable.sol	L497 - L497	*Pending Fix*
SSP_109224_21	InfinityNameUpgr...deable.sol	L175 - L175	*Pending Fix*
SSP_109224_22	InfinityNameUpgr...deable.sol	L546 - L546	*Pending Fix*

Issue Type

CHEAPER INEQUALITIES IN IF()

S. No.	Severity	Detection Method	Instances
G006	Gas	Automated	6

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_109224_206	InfinityNameUpgr...deable.sol	L142 - L142	*Pending Fix*
SSP_109224_207	InfinityNameUpgr...deable.sol	L175 - L175	*Pending Fix*
SSP_109224_208	InfinityNameUpgr...deable.sol	L194 - L194	*Pending Fix*
SSP_109224_209	InfinityNameUpgr...deable.sol	L248 - L248	*Pending Fix*
SSP_109224_210	InfinityNameUpgr...deable.sol	L355 - L355	*Pending Fix*
SSP_109224_211	InfinityNameUpgr...deable.sol	L546 - L546	*Pending Fix*

Issue Type

DEFAULT INT VALUES ARE MANUALLY RESET

S. No.	Severity	Detection Method	Instances
G007	Gas	Automated	4

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_109224_63	InfinityNameUpgr...deable.sol	L112 - L112	*Pending Fix*
SSP_109224_64	InfinityNameUpgr...deable.sol	L330 - L330	*Pending Fix*
SSP_109224_65	InfinityNameUpgr...deable.sol	L390 - L390	*Pending Fix*
SSP_109224_66	InfinityNameUpgr...deable.sol	L424 - L424	*Pending Fix*

Issue Type

DEFINE CONSTRUCTOR AS PAYABLE

S. No.	Severity	Detection Method	Instances
G008	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 deployment.

Bug ID	File Location	Line No.	Action Taken
SSP_109224_205	InfinityNameUpgr...deable.sol	L96 - L98	*Pending Fix*

Issue Type

FUNCTIONS CAN BE IN-LINED

S. No.	Severity	Detection Method	Instances
G009	Gas	Automated	5

Description

The internal function was called only once throughout the contract. Internal functions cost more gas due to additiona l JUMP instructions and stack operations.

Bug ID	File Location	Line No.	Action Taken
SSP_109224_231	InfinityNameSVG.sol	L40 - L58	*Pending Fix*
SSP_109224_232	InfinityNameSVG.sol	L63 - L79	*Pending Fix*
SSP_109224_233	InfinityNameSVG.sol	L84 - L94	*Pending Fix*
SSP_109224_234	InfinityNameSVG.sol	L99 - L109	*Pending Fix*
SSP_109224_235	InfinityNameSVG.sol	L114 - L130	*Pending Fix*

Issue Type

REVERTING FUNCTIONS CAN BE PAYABLE

S. No.	Severity	Detection Method	Instances
G010	Gas	Automated	6

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_109224_214	InfinityNameUpgr...deable.sol	L471 - L475	*Pending Fix*
SSP_109224_215	InfinityNameUpgr...deable.sol	L481 - L483	*Pending Fix*
SSP_109224_216	InfinityNameUpgr...deable.sol	L488 - L490	*Pending Fix*
SSP_109224_217	InfinityNameUpgr...deable.sol	L495 - L503	*Pending Fix*
SSP_109224_218	InfinityNameUpgr...deable.sol	L508 - L513	*Pending Fix*
SSP_109224_219	InfinityNameUpgr...deable.sol	L566 - L571	*Pending Fix*

Issue Type

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

S. No.	Severity	Detection Method	Instances
G011	Gas	Automated	6

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_109224_41	InfinityNameUpgr...deable.sol	L175 - L192	*Pending Fix*
SSP_109224_42	InfinityNameUpgr...deable.sol	L248 - L250	*Pending Fix*
SSP_109224_43	InfinityNameUpgr...deable.sol	L264 - L266	*Pending Fix*
SSP_109224_44	InfinityNameUpgr...deable.sol	L329 - L332	*Pending Fix*
SSP_109224_45	InfinityNameUpgr...deable.sol	L355 - L365	*Pending Fix*
SSP_109224_46	InfinityNameUpgr...deable.sol	L445 - L448	*Pending Fix*

Issue Type

GAS OPTIMIZATION IN INCREMENTS

S. No.	Severity	Detection Method	Instances
G012	Gas	Automated	2

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_109224_23	InfinityNameUpgr...deable.sol	L252 - L252	*Pending Fix*
SSP_109224_24	InfinityNameUpgr...deable.sol	L282 - L282	*Pending Fix*

Issue Type

HARDCODED GAS IN FUNCTION CALLS

S. No.	Severity	Detection Method	Instances
G013	Gas	Automated	1

Description

The function makes a call with a fixed amount of gas. If the receiver is a contract this may be insufficient to process t he receive() function. As a result the user would be unable to receive funds from this function or the call might fail.

Bug ID	File Location	Line No.	Action Taken
SSP_109224_212	InfinityNameUpgr...deable.sol	L177 - L177	*Pending Fix*

Issue Type

INTERNAL FUNCTIONS NEVER USED

S. No.	Severity	Detection Method	Instances
G014	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_109224_204	InfinityNameUpgr...deable.sol	L122 - L124	*Pending Fix*

Issue Type

LONG NUMBER LITERALS

S. No.	Severity	Detection Method	Instances
G015	Gas	Automated	1

Description

Solidity supports multiple rational and integer literals, including decimal fractions and scientific notations. The use of very large numbers with too many digits was detected in the code that could have been optimized using a different n otation also supported by Solidity.

Bug ID	File Location	Line No.	Action Taken
SSP_109224_174	InfinityNameUpgr...deable.sol	L111 - L111	*Pending Fix*

Issue Type

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

S. No.	Severity	Detection Method	Instances
G016	Gas	Automated	1

Description

The function having a return type is found to be declaring a local variable for returning, which causes extra gas consumption. 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_109224_53	InfinityNameUpgr...deable.sol	L276 - L287	*Pending Fix*

Issue Type

OPTIMIZING ADDRESS ID MAPPING

S. No.	Severity	Detection Method	Instances
G017	Gas	Automated	3

Description

Combining multiple address/ID mappings into a single mapping using a struct enhances storage efficiency, simplifies code, and reduces gas costs, resulting in a more streamlined and cost-effective smart contract design.
It saves storage slot for the mapping and depending on the circumstances and sizes of types, it can avoid a Gsset (2 0000 gas) per mapping combined. Reads and subsequent writes can also be cheaper when a function requires both values and they fit in the same storage slot.

Bug ID	File Location	Line No.	Action Taken
SSP_109224_236	InfinityNameUpgr...deable.sol	L38 - L38	*Pending Fix*
SSP_109224_237	InfinityNameUpgr...deable.sol	L41 - L41	*Pending Fix*
SSP_109224_238	InfinityNameUpgr...deable.sol	L51 - L51	*Pending Fix*

Issue Type

PUBLIC CONSTANTS CAN BE PRIVATE

S. No.	Severity	Detection Method	Instances
G018	Gas	Automated	4

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_109224_47	InfinityNameUpgr...deable.sol	L26 - L26	*Pending Fix*
SSP_109224_48	InfinityNameUpgr...deable.sol	L27 - L27	*Pending Fix*
SSP_109224_49	InfinityNameUpgr...deable.sol	L28 - L28	*Pending Fix*
SSP_109224_50	InfinityNameUpgr...deable.sol	L29 - L29	*Pending Fix*

Issue Type

STORAGE VARIABLE CACHING IN MEMORY

S. No.	Severity	Detection Method	Instances
G019	Gas	Automated	9

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_109224_58	InfinityNameUpgr...deable.sol	L128 - L198	*Pending Fix*
SSP_109224_58	InfinityNameUpgr...deable.sol	L128 - L198	*Pending Fix*
SSP_109224_58	InfinityNameUpgr...deable.sol	L128 - L198	*Pending Fix*
SSP_109224_58	InfinityNameUpgr...deable.sol	L128 - L198	*Pending Fix*
SSP_109224_59	InfinityNameUpgr...deable.sol	L321 - L345	*Pending Fix*
SSP_109224_59	InfinityNameUpgr...deable.sol	L321 - L345	*Pending Fix*
SSP_109224_60	InfinityNameUpgr...deable.sol	L350 - L377	*Pending Fix*
SSP_109224_61	InfinityNameUpgr...deable.sol	L444 - L450	*Pending Fix*
SSP_109224_62	InfinityNameUpgr...deable.sol	L545 - L555	*Pending Fix*

Issue Type

STORING STORAGE VARIABLES IN MEMORY

S. No.	Severity	Detection Method	Instances
G020	Gas	Automated	2

Description

Whenever a struct, array, or a mapping is stored and copied to a memory variable, each member is read from the sto rage and then copied. This becomes expensive. This could easily be optimized by using the storage keyword which j ust stores a pointer to the storage instead, making the whole process a lot cheaper.

Bug ID	File Location	Line No.	Action Taken
SSP_109224_200	InfinityNameUpgr...deable.sol	L279 - L279	*Pending Fix*
SSP_109224_201	InfinityNameUpgr...deable.sol	L306 - L306	*Pending Fix*

Issue Type

UNNECESSARY CHECKED ARITHMETIC IN LOOP

S. No.	Severity	Detection Method	Instances
G021	Gas	Automated	2

Description

Increments inside a loop could never overflow due to the fact that the transaction will run out of gas before the variable 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_109224_51	InfinityNameUpgr...deable.sol	L252 - L252	*Pending Fix*
SSP_109224_52	InfinityNameUpgr...deable.sol	L282 - L282	*Pending Fix*

Issue Type

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

S. No.	Severity	Detection Method	Instances
G022	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_109224_202	InfinityNameUpgr...deable.sol	L496 - L496	*Pending Fix*

05. Scan History

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

No	Date	Security Score	Scan Overview
1.	2025-10-18	**91.63**	● 0 ● 0 ● 0 ● 15 ● 150 ● 65

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.