SolidityScan | Powered by CRED SHIELDS

Security Assessment Report

etags_3

4 Dec 2025

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

Table of Contents

BLOCK VALUES AS A PROXY FOR TIME
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 @NOTICE IN NATSPEC COMMENTS FOR MODIFIERS
MISSING @PARAM IN NATSPEC COMMENTS FOR MODIFIERS
NAME MAPPING PARAMETERS
REVERT STATEMENTS WITHIN EXTERNAL AND PUBLIC FUNCTIONS CAN BE USED TO PERFORM DOS ATTACKS
ARRAY LENGTH CACHING
ASSIGNING TO STRUCTS CAN BE MORE EFFICIENT
AVOID RE-STORING VALUES
CHEAPER INEQUALITIES IN REQUIRE()
DEFINE CONSTRUCTOR AS PAYABLE
EMIT USED IN LOOP
FUNCTION SHOULD RETURN STRUCT
GAS INEFFICIENCY DUE TO MULTIPLE OPERANDS IN SINGLE IF/ELSEIF CONDITION

GAS OPTIMIZATION FOR STATE VARIABLES

GAS OPTIMIZATION IN INCREMENTS
NAMED RETURN OF LOCAL VARIABLE SAVES GAS AS COMPARED TO RETURN STATEMENT
PUBLIC CONSTANTS CAN BE PRIVATE
SPLITTING REQUIRE STATEMENTS
STORAGE VARIABLE CACHING IN MEMORY
SUPERFLUOUS EVENT FIELDS
UNNECESSARY CHECKED ARITHMETIC IN LOOP

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

etags 3

.sol

Uploaded Solidity File(s)

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

Security Score is AVERAGE

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 etags_3 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 etags_3 introduces new features or refactors the code.

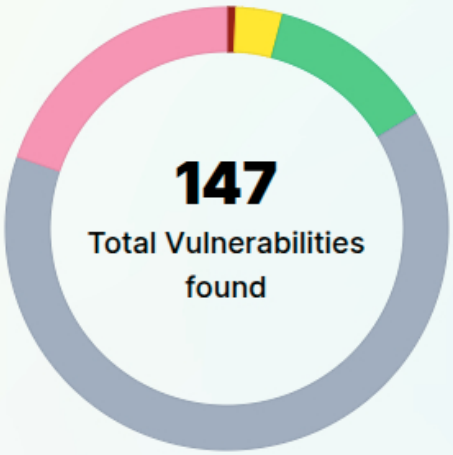
03. Findings Summary

.sol

etags_3

File Scan

Security Score	Scan duration	Lines of code
60.82/100	3 secs	336



1	0	5	18	94	29
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	0	0	147
Fixed	*False Positive*	*Won't Fix*	*Pending Fix*

S. No.	Severity	Bug Type	Instances	Detection Method	Status
C001	Critical	ERC721 SAFEMINT REENTRANCY	1	Automated	*Pending Fix*
M001	Medium	ARRAY LENGTH MANIPULATION	1	Automated	*Pending Fix*
M002	Medium	SUPPORTSINTERFACE() CALLS MAY REVERT	1	Automated	*Pending Fix*
M003	Medium	HASH COLLISIONS WITH STRING ARGUMENT ON ABI.ENCODEPACKED	3	Automated	*Pending Fix*
L001	Low	USE OF FLOATING PRAGMA	2	Automated	*Pending Fix*
L002	Low	MISSING EVENTS	9	Automated	*Pending Fix*
L003	Low	MISSING ZERO ADDRESS VALIDATION	4	Automated	*Pending Fix*

S. No.	Severity	Bug Type	Instances	Detection Method	Status
L004	Low	OUTDATED COMPILER VERSION	2	Automated	*Pending Fix*
L005	Low	UNUSUAL IMPLEMENTATION OF TOTALSUPPLY	1	Automated	*Pending Fix*
I001	Informational	ABI.ENCODEPACKED MAY CAUSE COLLISION	3	Automated	*Pending Fix*
I002	Informational	ADDING A RETURN STATEMENT WHEN THE	2	Automated	*Pending Fix*

		FUNCTION DEFINES A NAMED RETURN VARIABLE IS REDUNDANT			
I003	Informational	BLOCK VALUES AS A PROXY FOR TIME	2	Automated	Pending Fix
I004	Informational	MISSING @AUTHOR IN NATSPEC COMMENTS FOR CONTRACT DECLARATION	2	Automated	Pending Fix
I005	Informational	MISSING @DEV IN NATSPEC COMMENTS FOR CONTRACT DECLARATION	2	Automated	Pending Fix
I006	Informational	MISSING @DEV IN NATSPEC COMMENTS FOR FUNCTIONS	7	Automated	Pending Fix
I007	Informational	MISSING @INHERITDOC ON OVERRIDE FUNCTIONS	22	Automated	Pending Fix
I008	Informational	MISSING NATSPEC COMMENTS IN SCOPE BLOCKS	4	Automated	Pending Fix
I009	Informational	MISSING NATSPEC DESCRIPTIONS FOR PUBLIC VARIABLE DECLARATIONS	10	Automated	Pending Fix
I010	Informational	MISSING @NOTICE IN NATSPEC COMMENTS FOR CONSTRUCTORS	2	Automated	Pending Fix

S. No.	Severity	Bug Type	Instances	Detection Method	Status
I011	Informational	MISSING @NOTICE IN NATSPEC COMMENTS FOR FUNCTIONS	23	Automated	Pending Fix
I012	Informational	MISSING @NOTICE IN NATSPEC COMMENTS FOR MODIFIERS	1	Automated	Pending Fix
I013	Informational	MISSING @PARAM IN NATSPEC COMMENTS FOR MODIFIERS	1	Automated	Pending Fix
I014	Informational	NAME MAPPING PARAMETERS	5	Automated	Pending Fix

I015	Informational	REVERT STATEMENTS WITHIN EXTERNAL AND PUBLIC FUNCTIONS CAN BE USED TO PERFORM DOS ATTACKS	8	Automated	Pending Fix
G001	Gas	ARRAY LENGTH CACHING	1	Automated	Pending Fix
G002	Gas	ASSIGNING TO STRUCTS CAN BE MORE EFFICIENT	2	Automated	Pending Fix
G003	Gas	AVOID RE-STORING VALUES	3	Automated	Pending Fix
G004	Gas	CHEAPER INEQUALITIES IN REQUIRE()	1	Automated	Pending Fix
G005	Gas	DEFINE CONSTRUCTOR AS PAYABLE	2	Automated	Pending Fix
G006	Gas	EMIT USED IN LOOP	1	Automated	Pending Fix
G007	Gas	FUNCTION SHOULD RETURN STRUCT	2	Automated	Pending Fix
G008	Gas	GAS INEFFICIENCY DUE TO MULTIPLE OPERANDS IN SINGLE IF/ELSEIF CONDITION	1	Automated	Pending Fix
G009	Gas	GAS OPTIMIZATION FOR STATE VARIABLES	1	Automated	Pending Fix

S. No.	Severity	Bug Type	Instances	Detection Method	Status
G010	Gas	GAS OPTIMIZATION IN INCREMENTS	1	Automated	Pending Fix
G011	Gas	NAMED RETURN OF LOCAL VARIABLE SAVES GAS AS COMPARED TO RETURN STATEMENT	1	Automated	Pending Fix
G012	Gas	PUBLIC CONSTANTS CAN BE PRIVATE	4	Automated	Pending Fix
G013	Gas	SPLITTING REQUIRE STATEMENTS	1	Automated	Pending Fix
G014	Gas	STORAGE VARIABLE CACHING IN MEMORY	3	Automated	Pending Fix

G015	Gas	SUPERFLUOUS EVENT FIELDS	5	Automated	Pending Fix
G016	Gas	UNNECESSARY CHECKED ARITHMETIC IN LOOP	1	Automated	Pending Fix

04. Vulnerability Details

Issue Type

ERC721 SAFEMINT REENTRANCY

Upgrade your Plan to view the full report

1 Critical Issues Found

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

Upgrade

Issue Type

ARRAY LENGTH MANIPULATION

S. No.	Severity	Detection Method	Instances
M001	Medium	Automated	1

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

USE OF FLOATING PRAGMA

S. No.	Severity	Detection Method	Instances
L001	Low	Automated	2

Upgrade your Plan to view the full report

2 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

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

ARRAY LENGTH CACHING

S. No.	Severity	Detection Method	Instances
G001	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 favorable 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_116210_111	etags_3/smartcon...gistry.sol	L107 - L120	*Pending Fix*

Issue Type

ASSIGNING TO STRUCTS CAN BE MORE EFFICIENT

S. No.	Severity	Detection Method	Instances
G002	Gas	Automated	2

Description

The contract is found to contain a struct with multiple variables defined in it. When a struct is assigned in a sin gle operation, Solidity may perform costly storage operations, which can be inefficient. This often results in inc reased gas costs due to multiple SLOAD and SSTORE operations happening at once

Bug ID	File Location	Line No.	Action Taken
SSP_116210_57	etags_3/smartcon...gistry.sol	L81 - L88	*Pending Fix*
SSP_116210_58	etags_3/smartcon...gistry.sol	L109 - L116	*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 valu e is equal to the new value. This practice results in unnecessary gas consumption due to the `Gsreset` operati on (2900 gas), which could be avoided. If the old value and the new value are the same, not updating the stora ge would avoid this cost and could instead incur a `Gcoldsload` (2100 gas) or a `Gwarmaccess` (100 gas), pot entially saving gas.

Bug ID	File Location	Line No.	Action Taken
SSP_116210_130	etags_3/smartcon...gistry.sol	L72 - L94	*Pending Fix*
SSP_116210_131	etags_3/smartcon...gistry.sol	L99 - L122	*Pending Fix*
SSP_116210_132	etags_3/smartcon...gistry.sol	L179 - L186	*Pending Fix*

Issue Type

CHEAPER INEQUALITIES IN REQUIRE()

S. No.	Severity	Detection Method	Instances
G004	Gas	Automated	1

Description

The contract was found to be performing comparisons using inequalities inside the `require` statement. Whe n inside the `require` statements, non-strict inequalities `(>=, <=)` are usually costlier than strict equalities `(>, <)`.

Bug ID	File Location	Line No.	Action Taken
SSP_116210_110	etags_3/smartcon...gistry.sol	L105 - L105	*Pending Fix*

Issue Type

DEFINE CONSTRUCTOR AS PAYABLE

S. No.	Severity	Detection Method	Instances
G005	Gas	Automated	2

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 depl oyment.

Bug ID	File Location	Line No.	Action Taken
SSP_116210_55	etags_3/smartcon...gistry.sol	L61 - L65	*Pending Fix*
SSP_116210_56	etags_3/smartcon...ctible.sol	L48 - L55	*Pending Fix*

Issue Type

EMIT USED IN LOOP

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

Description

In Solidity, when a code emits an event inside of a loop, internally it performs a LOG operation N times, where N represents the number of iterations in the loop. This can lead to inflated gas costs and potentially impact the efficiency of the code.

Bug ID	File Location	Line No.	Action Taken
SSP_116210_129	etags_3/smartcon...gistry.sol	L108 - L119	*Pending Fix*

Issue Type

FUNCTION SHOULD RETURN STRUCT

S. No.	Severity	Detection Method	Instances
G007	Gas	Automated	2

Description

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

Bug ID	File Location	Line No.	Action Taken
SSP_116210_59	etags_3/smartcon...gistry.sol	L129 - L144	*Pending Fix*
SSP_116210_60	etags_3/smartcon...gistry.sol	L149 - L165	*Pending Fix*

Issue Type

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

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

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_116210_54	etags_3/smartcon...ctible.sol	L123 - L123	*Pending Fix*

Issue Type

GAS OPTIMIZATION FOR STATE VARIABLES

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

Description

Plus equals (`+=`) costs more gas than addition operator. The same thing happens with minus equals (`-=`).

Bug ID	File Location	Line No.	Action Taken
SSP_116210_117	etags_3/smartcon...gistry.sol	L121 - L121	⚠ *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 incre mented and returned, thus, making it a cheaper alternative.

Bug ID	File Location	Line No.	Action Taken
SSP_116210_41	etags_3/smartcon...gistry.sol	L107 - L107	*Pending Fix*

Issue Type

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

S. No.	Severity	Detection Method	Instances
G011	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 co mputational steps and memory allocation.

Bug ID	File Location	Line No.	Action Taken
SSP_116210_3	etags_3/smartcon...ctible.sol	L66 - L93	*Pending Fix*

Issue Type

PUBLIC CONSTANTS CAN BE PRIVATE

S. No.	Severity	Detection Method	Instances
G012	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_116210_42	etags_3/smartcon...gistry.sol	L15 - L15	*Pending Fix*
SSP_116210_43	etags_3/smartcon...gistry.sol	L16 - L16	*Pending Fix*
SSP_116210_44	etags_3/smartcon...ctible.sol	L16 - L16	*Pending Fix*
SSP_116210_45	etags_3/smartcon...ctible.sol	L17 - L17	*Pending Fix*

Issue Type

SPLITTING REQUIRE STATEMENTS

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

Description

Require statements when combined using operators in a single statement usually lead to a larger deployment g as cost but with each runtime calls, the whole thing ends up being cheaper by some gas units.

Bug ID	File Location	Line No.	Action Taken
SSP_116210_11	etags_3/smartcon...gistry.sol	L104 - L104	*Pending Fix*

Issue Type

STORAGE VARIABLE CACHING IN MEMORY

S. No.	Severity	Detection Method	Instances
G014	Gas	Automated	3

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_116210_135	etags_3/smartcon...gistry.sol	L99 - L122	*Pending Fix*
SSP_116210_135	etags_3/smartcon...gistry.sol	L99 - L122	*Pending Fix*
SSP_116210_136	etags_3/smartcon...gistry.sol	L129 - L144	*Pending Fix*

Issue Type

SUPERFLUOUS EVENT FIELDS

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

Description

`block.timestamp` and `block.number` are by default added to event information. Adding them manually co sts extra gas.

Bug ID	File Location	Line No.	Action Taken
SSP_116210_112	etags_3/smartcon...gistry.sol	L93 - L93	*Pending Fix*
SSP_116210_113	etags_3/smartcon...gistry.sol	L118 - L118	*Pending Fix*
SSP_116210_114	etags_3/smartcon...gistry.sol	L185 - L185	*Pending Fix*
SSP_116210_115	etags_3/smartcon...gistry.sol	L195 - L195	*Pending Fix*
SSP_116210_116	etags_3/smartcon...gistry.sol	L196 - L196	*Pending Fix*

Issue Type

UNNECESSARY CHECKED ARITHMETIC IN LOOP

S. No.	Severity	Detection Method	Instances
G016	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 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_116210_9	etags_3/smartcon...gistry.sol	L107 - L107	*Pending Fix*

05. Scan History

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

No	Date	Security Score	Scan Overview
1.	2025-12-04	60.82	● 1 ● 0 ● 5 ● 18 ● 94 ● 29

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.