Assure DeFi™

The Verification **Gold Standard ™**

Security Assessment

Website AI Staking

November 27, 2023

Audit Status: Fail

Audit Edition: Advance

Project Overview

Token Summary

Parameter	Result
Address	0xA9e408bC69727188990291fad086B3E275cb7222
Name	Website AI
Token Tracker	Website AI (WEBAI)
Decimals	9
Supply	1,000,000
Platform	Ethereum
compiler	v0.8.18+commit.87f61d96
Contract Name	StakingElixir
Optimization	Yes with 200 runs
LicenseType	MIT
Language	Solidity
Codebase	https://etherscan.io/address/0xa9e408bc69727188990291fad086b3e275cb7222#code
Payment Tx	Corporate

Main Contract Assessed
Contract Name

Name	Contract	Live
Website AI	0xA9e408bC69727188990291fad086B3E275cb7222	Yes

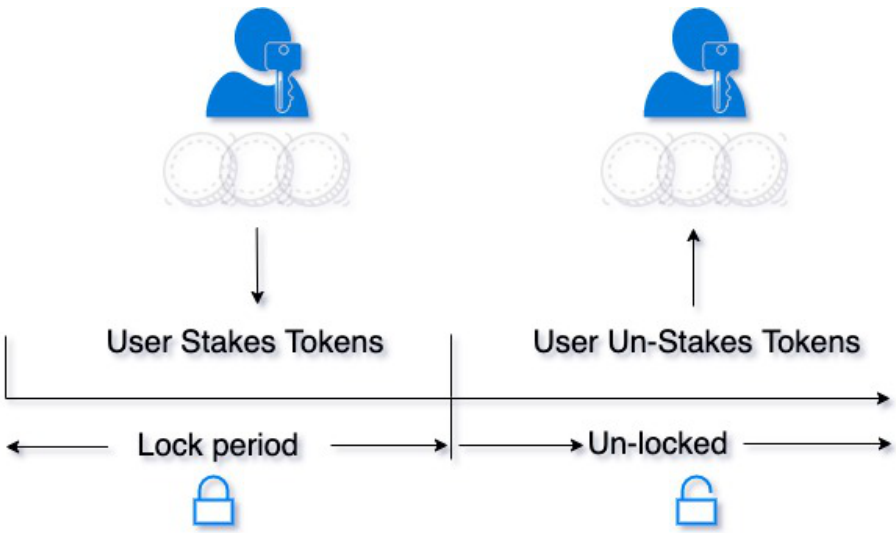
TestNet Contract was Not Assessed

Solidity Code Provided

SolID	File Sha-1	FileName
WEBAI	31f064597672c4e93bbaec1bd9da14750576776e	elixirstaking.sol
WEBAI		
WEBAI		
WEBAI		

What is a Staking Contract

A smart contract which allows users to stake and un-stake a specified ERC20 token. Staked tokens are locked for a specific length of time (set by the contrat owner at the outset). Once the time period has elapsed, the user can remove their tokens again.



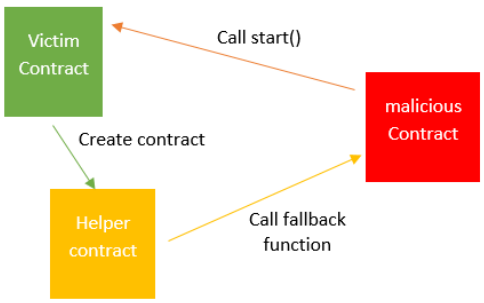
Reentrancy Check

The Project Owners of Website AI have not configure the Reentrancy Guard library.

You can read more about Reentrancy issues in the following link.
Reentrancy After Istanbul.

We recommend the team to add the library to the contract to avoid potential issues.

We recommend the team to create a new contract with Reentrancy Guard added to the same.



Smart Contract Vulnerability Checks

The Smart Contract Weakness Classification Registry (SWC Registry) is an implementation of the weakness classification scheme proposed in EIP-1470. It is loosely aligned to the terminologies and structure used in the Common Weakness Enumeration (CWE) while overlaying a wide range of weakness variants that are specific to smart contracts.

ID	Severity	Name	File	location
SWC-100	Pass	Function Default Visibility	elixirstaking.sol	L: 0 C: 0
SWC-101	Pass	Integer Overflow and Underflow.	elixirstaking.sol	L: 0 C: 0
SWC-102	Pass	Outdated Compiler Version file.	elixirstaking.sol	L: 0 C: 0
SWC-103	Low	A floating pragma is set.	elixirstaking.sol	L: 7 C: 0
SWC-104	Pass	Unchecked Call Return Value.	elixirstaking.sol	L: 0 C: 0
SWC-105	Pass	Unprotected Ether Withdrawal.	elixirstaking.sol	L: 0 C: 0
SWC-106	Pass	Unprotected SELFDESTRUCT Instruction	elixirstaking.sol	L: 0 C: 0
SWC-107	Pass	Read of persistent state following external call.	elixirstaking.sol	L: 0 C: 0
SWC-108	Pass	State variable visibility is not set..	elixirstaking.sol	L: 0 C: 0
SWC-109	Pass	Uninitialized Storage Pointer.	elixirstaking.sol	L: 0 C: 0
SWC-110	Pass	Assert Violation.	elixirstaking.sol	L: 0 C: 0
SWC-111	Pass	Use of Deprecated Solidity Functions.	elixirstaking.sol	L: 0 C: 0
SWC-112	Pass	Delegate Call to Untrusted Callee.	elixirstaking.sol	L: 0 C: 0
SWC-113	Pass	Multiple calls are executed in the same transaction.	elixirstaking.sol	L: 0 C: 0

ID	Severity	Name	File	location
SWC-114	Pass	Transaction Order Dependence.	elixirstaking.sol	L: 0 C: 0
SWC-115	Pass	Authorization through tx.origin.	elixirstaking.sol	L: 0 C: 0
SWC-116	Pass	A control flow decision is made based on The block.timestamp environment variable.	elixirstaking.sol	L: 0 C: 0
SWC-117	Pass	Signature Malleability.	elixirstaking.sol	L: 0 C: 0
SWC-118	Pass	Incorrect Constructor Name.	elixirstaking.sol	L: 0 C: 0
SWC-119	Pass	Shadowing State Variables.	elixirstaking.sol	L: 0 C: 0
SWC-120	Pass	Potential use of block.number as source of randonmness.	elixirstaking.sol	L: 0 C: 0
SWC-121	Pass	Missing Protection against Signature Replay Attacks.	elixirstaking.sol	L: 0 C: 0
SWC-122	Pass	Lack of Proper Signature Verification.	elixirstaking.sol	L: 0 C: 0
SWC-123	Pass	Requirement Violation.	elixirstaking.sol	L: 0 C: 0
SWC-124	Pass	Write to Arbitrary Storage Location.	elixirstaking.sol	L: 0 C: 0
SWC-125	Pass	Incorrect Inheritance Order.	elixirstaking.sol	L: 0 C: 0
SWC-126	Pass	Insufficient Gas Griefing.	elixirstaking.sol	L: 0 C: 0
SWC-127	Pass	Arbitrary Jump with Function Type Variable.	elixirstaking.sol	L: 0 C: 0
SWC-128	Pass	DoS With Block Gas Limit.	elixirstaking.sol	L: 0 C: 0
SWC-129	Pass	Typographical Error.	elixirstaking.sol	L: 0 C: 0
SWC-130	Pass	Right-To-Left-Override control character (U +202E).	elixirstaking.sol	L: 0 C: 0
SWC-131	Pass	Presence of unused variables.	elixirstaking.sol	L: 0 C: 0

ID	Severity	Name	File	location
SWC-132	Pass	Unexpected Ether balance.	elixirstaking.sol	L: 0 C: 0
SWC-133	Pass	Hash Collisions with Multiple Variable Length Arguments.	elixirstaking.sol	L: 0 C: 0
SWC-134	Pass	Message call with hardcoded gas amount.	elixirstaking.sol	L: 0 C: 0
SWC-135	Pass	Code With No Effects (Irrelevant/Dead Code).	elixirstaking.sol	L: 0 C: 0
SWC-136	Pass	Unencrypted Private Data On-Chain.	elixirstaking.sol	L: 0 C: 0

We scan the contract for additional security issues using MYTHX and industry-standard security scanning tools.

Smart Contract Vulnerability Details

SWC-103 - Floating Pragma.

CWE-664: Improper Control of a Resource Through its Lifetime.

References:

Description:

Contracts should be deployed with the same compiler version and flags that they have been tested with thoroughly. Locking the pragma helps to ensure that contracts do not accidentally get deployed using, for example, an outdated compiler version that might introduce bugs that affect the contract system negatively.

Remediation:

Lock the pragma version and also consider known bugs (https://github.com/ethereum/solidity/releases) for the compiler version that is chosen.

Pragma statements can be allowed to float when a contract is intended for consumption by other developers, as in the case with contracts in a library or EthPM package. Otherwise, the developer would need to manually update the pragma in order to compile locally.

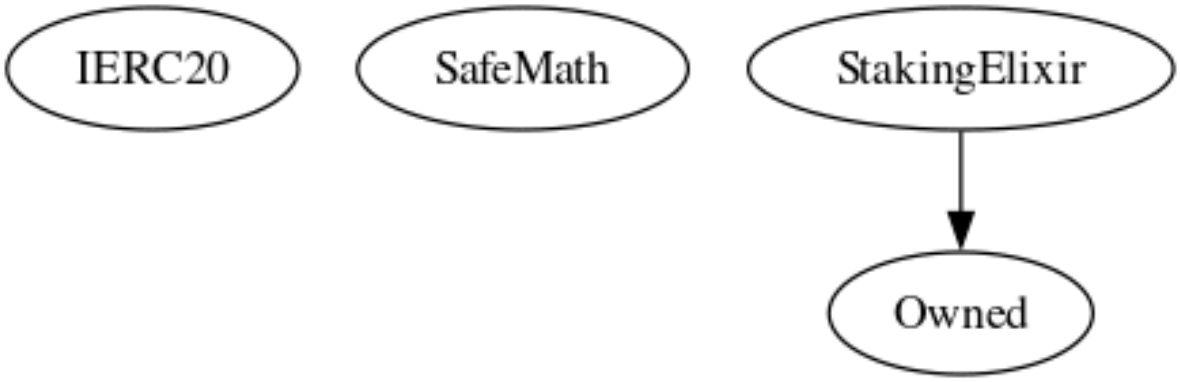
References:

Ethereum Smart Contract Best Practices - Lock pragmas to specific compiler version.

Inheritance

The contract for Website AI has the following inheritance structure.

The Project has a Total Supply of 1,000,000



Privileged Functions (onlyOwner)

Please Note if the contract is Renounced none of this functions can be executed.

Function Name	Parameters	Visibility
filter		external
setMinimumStakeValue		external
setRegistrationTax		external
setUnstakingTaxRate		external
setStakingTaxRate		external
changeActiveStatus		external
rewardPool		external
unstake		external
stake		external
transferOwnership		external

WEBAI-02 | Function Visibility Optimization.

Category	Severity	Location	Status
Gas Optimization	Low	elixirstaking.sol: L: 291 C: 14	Detected

Description

The following functions are declared as public and are not invoked in any of the contracts contained within the projects scope:

Function Name	Parameters	Visibility
transferOwnership	account (address)	public

The functions that are never called internally within the contract should have external visibility

Remediation

We advise that the function's visibility specifiers are set to external, and the array-based arguments change their data location from memory to calldata, optimizing the gas cost of the function.

References:

external vs public best practices.

WEBAI-03 | Lack of Input Validation.

Category	Severity	Location	Status
Volatile Code	Low	elixirstaking.sol: L: 319 C: 14,L: 315 C: 14,L: 311 C: 14,L: 303 C: 14,: 299 C: 14	Detected

Description

The given input is missing the check for the non-zero address.

The given input is missing the check for the missing required function.

Remediation

We advise the client to add the check for the passed-in values to prevent unexpected errors as below:

```
...
 require(receiver != address(0), "Receiver is the zero address");
...
...
require(value X limitation, "Your not able to do this function");
...
```

We also recommend customer to review the following function that is missing a required validation. missing required function.

WEBAI-05 | Missing Event Emission.

Category	Severity	Location	Status
Volatile Code	Low	elixirstaking.sol: L: 319 C: 14,L: 315 C: 14,L: 311 C: 14,L: 303 C: 14,: 299 C: 14	Detected

Description

Detected missing events for critical arithmetic parameters. There are functions that have no event emitted, so it is difficult to track off-chain changes.The linked code does not create an event for the transfer.

Remediation

Emit an event for critical parameter changes. It is recommended emitting events for the sensitive functions that are controlled by centralization roles.

WEBAI-06 | Conformance with Solidity Naming Conventions.

Category	Severity	Location	Status
Coding Style	Low	elixirstaking.sol: L: 246 C: 13	Detected

Description

Solidity defines a naming convention that should be followed. Rule exceptions: Allow constant variable name/symbol/decimals to be lowercase. Allow _ at the beginning of the mixed_case match for private variables and unused parameters.

Remediation

Follow the Solidity naming convention.

https://docs.soliditylang.org/en/v0.4.25/style-guide.html#naming-convention

WEBAI-14 | Unnecessary Use Of SafeMath

Category	Severity	Location	Status
Logical Issue	Low	elixirstaking.sol: L: 44 C: 9	Detected

Description

The SafeMath library is used unnecessarily. With Solidity compiler versions 0.8.0 or newer, arithmetic operations
will automatically revert in case of integer overflow or underflow.
library SafeMath {
An implementation of SafeMath library is found.
using SafeMath for uint256;
SafeMath library is used for uint256 type in contract.

Remediation

We advise removing the usage of SafeMath library and using the built-in arithmetic operations provided by the
Solidity programming language

Project Action

Technical Findings Summary

Classification of Risk

Severity	Description
Critical	Risks are those that impact the safe functioning of a platform and must be addressed before launch. Users should not invest in any project with outstanding critical risks.
High	Risks can include centralization issues and logical errors. Under specific circumstances, these major risks can lead to loss of funds and/or control of the project.
Medium	Risks may not pose a direct risk to users' funds, but they can affect the overall functioning of a platform
Low	Risks can be any of the above but on a smaller scale. They generally do not compromise the overall integrity of the Project, but they may be less efficient than other solutions.
Informational	Errors are often recommended to improve the code's style or certain operations to fall within industry best practices. They usually do not affect the overall functioning of the code.

Findings

Severity	Found	Pending	Resolved
Critical	0	0	0
High	0	0	0
Medium	1	1	0
Low	3	3	0
Informational	1	1	0
Total	5	5	0

Social Media Checks

Social Media	URL	Result
Twitter	https://x.com/websiteai_erc	Pass
Other		Fail
Website	https://websiteai.io	Pass
Telegram	https://t.me/Website_AI	Pass

We recommend to have 3 or more social media sources including a completed working websites.

Social Media Information Notes:

Auditor Notes: undefined

Project Owner Notes:

Audit Result

Final Audit Score

Review	Score
Security Score	79
Auditor Score	75

The Following Score System Has been Added to this page to help understand the value of the audit, the maximun score is 100, however to attain that value the project most pass and provide all the data needed for the assessment. Our Passing Score has been changed to 85 Points for a higher standard, if a project does not attain 85% is an automatic failure. Read our notes and final assessment below.

Audit Fail

Assessment Results

Important Notes:

- Several items need improvements in this code.ı
- The following is a staking contract.ı
- The contract has safemath and is no longer needed.ı
- The contract needs to implement re-entrancy guard.ı
- We consider this staking as vulnerable to exploits since is missing require and emits in many areas.

Auditor Score =75
Audit Fail

Appendix

Finding Categories

Centralization / Privilege

Centralization / Privilege findings refer to either feature logic or implementation of components that actagainst the nature of decentralization, such as explicit ownership or specialized access roles incombination with a mechanism to relocate funds.

Gas Optimization

Gas Optimization findings do not affect the functionality of the code but generate different, more optimalEVM opcodes resulting in a reduction on the total gas cost of a transaction.

Logical Issue

Logical Issue findings detail a fault in the logic of the linked code, such as an incorrect notion on howblock.timestamp works.

Control Flow

Control Flow findings concern the access control imposed on functions, such as owner-only functionsbeing invoke-able by anyone under certain circumstances.

Volatile Code

Volatile Code findings refer to segments of code that behave unexpectedly on certain edge cases that mayresult in a vulnerability.

Coding Style

Coding Style findings usually do not affect the generated byte-code but rather comment on how to makethe codebase more legible and, as a result, easily maintainable.

Inconsistency

Inconsistency findings refer to functions that should seemingly behave similarly yet contain different code,such as a constructor assignment imposing different require statements on the input variables than a setterfunction.

Coding Best Practices

ERC 20 Conding Standards are a set of rules that each developer should follow to ensure the code meet a set of creterias and is readable by all the developers.

Disclaimer

Assure Defi has conducted an independent security assessment to verify the integrity of and highlight any vulnerabilities or errors, intentional or unintentional, that may be present in the reviewed code for the scope of this assessment. This report does not constitute agreement, acceptance, or advocation for the Project, and users relying on this report should not consider this as having any merit for financial advice in any shape, form, or nature. The contracts audited do not account for any economic developments that the Project in question may pursue, and the veracity of the findings thus presented in this report relate solely to the proficiency, competence, aptitude, and discretion of our independent auditors, who make no guarantees nor assurance that the contracts are entirely free of exploits, bugs, vulnerabilities or deprecation of technologies.

All information provided in this report does not constitute financial or investment advice, nor should it be used to signal that any persons reading this report should invest their funds without sufficient individual due diligence, regardless of the findings presented. Information is provided ‘as is, and Assure Defi is under no covenant to audited completeness, accuracy, or solidity of the contracts. In no event will Assure Defi or its partners, employees, agents, or parties related to the provision of this audit report be liable to any parties for, or lack thereof, decisions or actions with regards to the information provided in this audit report.

The assessment services provided by Assure Defi are subject to dependencies and are under continuing development. You agree that your access 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. Cryptographic tokens are emergent technologies with high levels of technical risk and uncertainty. The assessment reports could include false positives, negatives, and unpredictable results. The services may access, and depend upon, multiple layers of third parties.