



CFG NINJA AUDITS

Security Assessment

AlTom Sale

July 17, 2023

Audit Status: Pass

Audit Edition: Advanced

Project Overview

Token Summary

Parameter	Result
Address	0x
Name	AITom
Token Tracker	AITom (TomSale)
Decimals	18
Supply	0
Platform	Ethereum
compiler	v0.8.19+commit.7dd6d404
Contract Name	TomSale
Optimization	Yes with 200 runs
LicenseType	MIT
Language	Solidity
Codebase	https://etherscan.io/token/
Payment Tx	Corporate



Main Contract Assessed

Contract Name

Name	Contract	Live
AITom	0x	Yes

TestNet Contract was Not Assessed

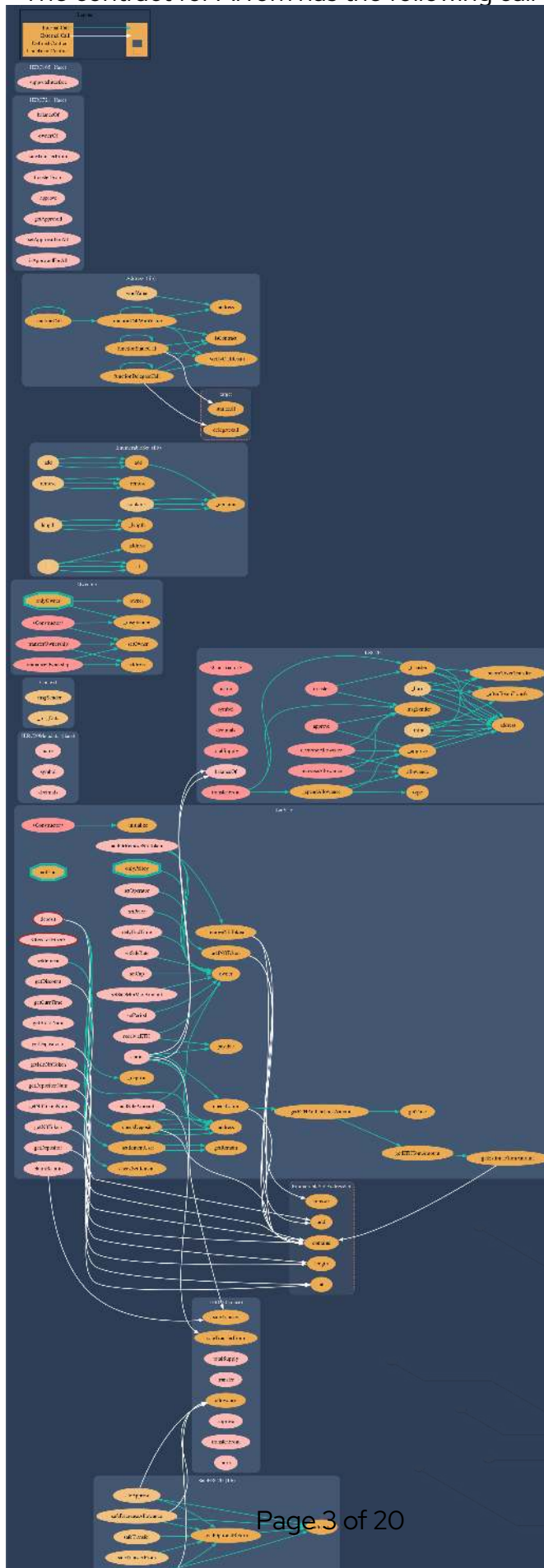
Solidity Code Provided

SolID	File Sha-1	FileName
AiTom	f7af934db07e5cfe89b59d619038345560ee7c88	TomSale.sol
AiTom	68a6306f8c5cb8b28c2c1c9c8da1219998e86d3d	Ownable.sol
AiTom	41f987558cd34ad75ab7d03b61ae89bffd243ef	Context.sol
AiTom	9614db6b9e6b88f2df65ef00236137548d7256ed	SafeERC20.sol



Call Graph

The contract for AlTom has the following call graph structure.



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	TomSale.sol	L: 0 C: 0
SWC-101	Pass	Integer Overflow and Underflow.	TomSale.sol	L: 0 C: 0
SWC-102	Pass	Outdated Compiler Version file.	TomSale.sol	L: 0 C: 0
SWC-103	Pass	A floating pragma is set.	TomSale.sol	L: 0 C: 0
SWC-104	Pass	Unchecked Call Return Value.	TomSale.sol	L: 0 C: 0
SWC-105	Pass	Unprotected Ether Withdrawal.	TomSale.sol	L: 0 C: 0
SWC-106	Pass	Unprotected SELFDESTRUCT Instruction	TomSale.sol	L: 0 C: 0
SWC-107	Pass	Read of persistent state following external call.	TomSale.sol	L: 0 C: 0
SWC-108	Low	State variable visibility is not set..	TomSale.sol	L: 79 C: 29,L: 80 C: 29
SWC-109	Pass	Uninitialized Storage Pointer.	TomSale.sol	L: 0 C: 0
SWC-110	Pass	Assert Violation.	TomSale.sol	L: 0 C: 0



ID	Severity	Name	File	location
SWC-111	Pass	Use of Deprecated Solidity Functions.	TomSale.sol	L: 0 C: 0
SWC-112	Pass	Delegate Call to Untrusted Callee.	TomSale.sol	L: 0 C: 0
SWC-113	Pass	Multiple calls are executed in the same transaction.	TomSale.sol	L: 0 C: 0
SWC-114	Pass	Transaction Order Dependence.	TomSale.sol	L: 0 C: 0
SWC-115	Pass	Authorization through tx.origin.	TomSale.sol	L: 0 C: 0
SWC-116	Pass	A control flow decision is made based on The block.timestamp environment variable.	TomSale.sol	L: 0 C: 0
SWC-117	Pass	Signature Malleability.	TomSale.sol	L: 0 C: 0
SWC-118	Pass	Incorrect Constructor Name.	TomSale.sol	L: 0 C: 0
SWC-119	Pass	Shadowing State Variables.	TomSale.sol	L: 0 C: 0
SWC-120	Low	Potential use of block.number as source of randomness.	TomSale.sol	L: 474 C: 15
SWC-121	Pass	Missing Protection against Signature Replay Attacks.	TomSale.sol	L: 0 C: 0
SWC-122	Pass	Lack of Proper Signature Verification.	TomSale.sol	L: 0 C: 0
SWC-123	Pass	Requirement Violation.	TomSale.sol	L: 0 C: 0
SWC-124	Pass	Write to Arbitrary Storage Location.	TomSale.sol	L: 0 C: 0
SWC-125	Pass	Incorrect Inheritance Order.	TomSale.sol	L: 0 C: 0



ID	Severity	Name	File	location
SWC-126	Pass	Insufficient Gas Griefing.	TomSale.sol	L: 0 C: 0
SWC-127	Pass	Arbitrary Jump with Function Type Variable.	TomSale.sol	L: 0 C: 0
SWC-128	Pass	DoS With Block Gas Limit.	TomSale.sol	L: 0 C: 0
SWC-129	Pass	Typographical Error.	TomSale.sol	L: 0 C: 0
SWC-130	Pass	Right-To-Left-Override control character (U+202E).	TomSale.sol	L: 0 C: 0
SWC-131	Pass	Presence of unused variables.	TomSale.sol	L: 0 C: 0
SWC-132	Pass	Unexpected Ether balance.	TomSale.sol	L: 0 C: 0
SWC-133	Pass	Hash Collisions with Multiple Variable Length Arguments.	TomSale.sol	L: 0 C: 0
SWC-134	Pass	Message call with hardcoded gas amount.	TomSale.sol	L: 0 C: 0
SWC-135	Low	Code With No Effects (Irrelevant/Dead Code).	TomSale.sol	L: 264 C: 12
SWC-136	Pass	Unencrypted Private Data On-Chain.	TomSale.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-108 - State Variable Default Visibility

CWE-710: Improper Adherence to Coding Standards

Description:

Labeling the visibility explicitly makes it easier to catch incorrect assumptions about who can access the variable.

Remediation:

Variables can be specified as being public, internal or private. Explicitly define visibility for all state variables.

References:

Ethereum Smart Contract Best Practices - Explicitly mark visibility in functions and state variables



Smart Contract Vulnerability Details

SWC-120 - Weak Sources of Randomness from Chain Attributes

CWE-330: Use of Insufficiently Random Values

Description:

Solidity allows for ambiguous naming of state variables when inheritance is used. Contract A with a variable x could inherit contract B that also has a state variable x defined. This would result in two separate versions of x, one of them being accessed from contract A and the other one from contract B. In more complex contract systems this condition could go unnoticed and subsequently lead to security issues.

Shadowing state variables can also occur within a single contract when there are multiple definitions on the contract and function level.

Remediation:

Using commitment scheme, e.g. RANDAO. Using external sources of randomness via oracles, e.g. Oraclize. Note that this approach requires trusting in oracle, thus it may be reasonable to use multiple oracles. Using Bitcoin block hashes, as they are more expensive to mine.

References:

How can I securely generate a random number in my smart contract?)

When can BLOCKHASH be safely used for a random number? When would it be unsafe?

The Run smart contract.



Smart Contract Vulnerability Details

SWC-135 - Code With No Effects

CWE-1164: Irrelevant Code

Description:

In Solidity, it's possible to write code that does not produce the intended effects. Currently, the solidity compiler will not return a warning for effect-free code. This can lead to the introduction of "dead" code that does not properly performing an intended action.

For example, it's easy to miss the trailing parentheses in `msg.sender.call.value(address(this).balance)("")`, which could lead to a function proceeding without transferring funds to `msg.sender`. Although, this should be avoided by checking the return value of the call.

Remediation:

It's important to carefully ensure that your contract works as intended. Write unit tests to verify correct behaviour of the code.

References:

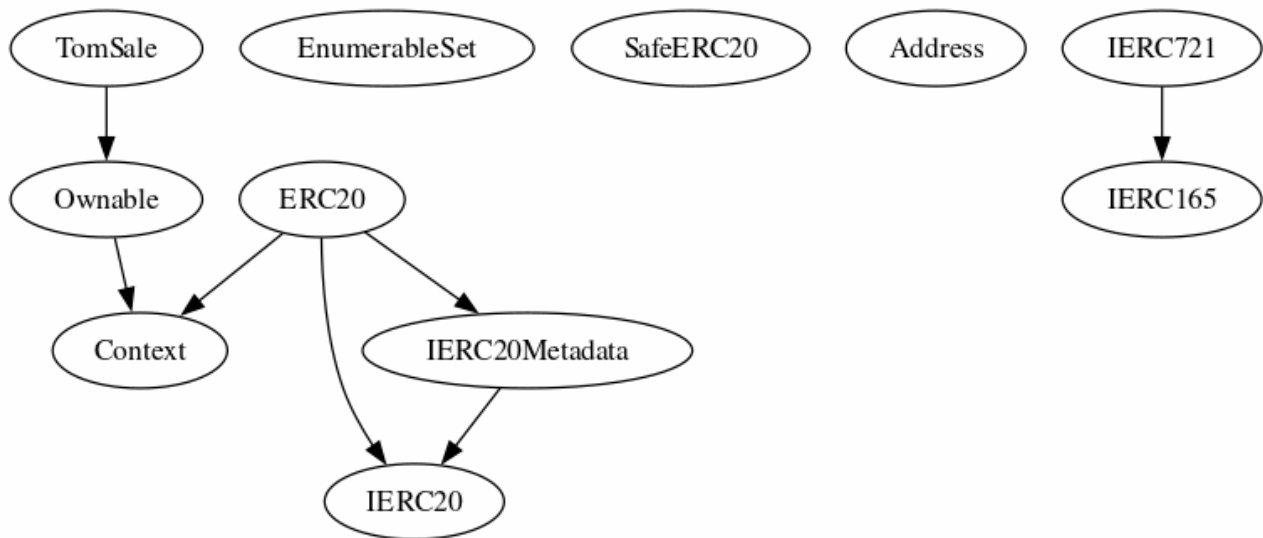
Issue on Solidity's Github - raise an error when a statement can never have side-effects

Issue on Solidity's Github - `msg.sender.call.value(address(this).balance);` should produce a warning



Inheritance

The contract for AlTom has the following inheritance structure.



Smart Contract Advance Checks



ID	Severity	Name	Result	Status
TomSale-01	Low	Potential Sandwich Attacks.	Pass	Not Detected
TomSale-02	Informational	Function Visibility Optimization	Fail	Detected
TomSale-03	Low	Lack of Input Validation.	Pass	Not Detected
TomSale-04	High	Centralized Risk In addLiquidity.	Pass	Not Detected
TomSale-05	Low	Missing Event Emission.	Pass	Not Detected
TomSale-06	Low	Conformance with Solidity Naming Conventions.	Pass	Not Detected
TomSale-07	Low	State Variables could be Declared Constant.	Pass	Not Detected
TomSale-08	Low	Dead Code Elimination.	Pass	Not Detected
TomSale-09	High	Third Party Dependencies.	Pass	Not Detected
TomSale-10	High	Initial Token Distribution.	Pass	Not Detected
TomSale-11	High	claimStuckTokens can claim own tokens.	Pass	Not Detected
TomSale-12	High	Centralization Risks In The X Role	Pass	Not Detected
TomSale-13	Informational	Extra Gas Cost For User..	Pass	Not Detected
TomSale-14	Medium	Unnecessary Use Of SafeMath	Pass	Not Detected
TomSale-15	Medium	Symbol Length Limitation due to Solidity Naming Standards.	Pass	Not Detected



ID	Severity	Name	Result	Status
TomSale-16	Medium	Taxes can be up to 100%	Pass	Not Detected
TomSale-17	Logical Issue	Highly Permissive Role Access,	Pass	Not Detected
TomSale-18	Critical	Stop Transactions by using Enable Trade.	Pass	Not Detected



TomSale-02 | Function Visibility Optimization.

Category	Severity	Location	Status
Gas Optimization	 Informational	TomSale.sol: L: 79 C: 29	 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
depositor		internal
nftToken		internal
teamlist		internal

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.








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	0	0	0
 Low	0	0	0
 Informational	1	0	0
Total	1	0	0



Social Media Checks

Social Media	URL	Result
Twitter	https://twitter.com/MemeAitom	Pass
Other		Fail
Website	https://aitom.pro/	Pass
Telegram	https://t.me/AITomPro	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:



Assessment Results

Score Results

Review	Score
Overall Score	100/100
Auditor Score	90/100
Review by Section	Score
Manual Scan Score	43/33
SWC Scan Score	31/37
Advance Check Score	34/30

The Following Score System Has been Added to this page to help understand the value of the audit, the maximum score is 100, however to attain that value the project must pass and provide all the data needed for the assessment. Our Passing Score has been changed to 80 Points, if a project does not attain 80% is an automatic failure. Read our notes and final assessment below.

Audit Passed



Assessment Results

Important Notes:

- This is a custom contract for an ido or sale.
- Please DYOR on the project.

Auditor Score =90
Audit Passed



Appendix

Finding Categories

Centralization / Privilege

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

Gas Optimization

Gas Optimization findings do not affect the functionality of the code but generate different, more optimal EVM 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 how `block.timestamp` works.

Control Flow

Control Flow findings concern the access control imposed on functions, such as owner-only functions being 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 may result in a vulnerability.

Coding Style

Coding Style findings usually do not affect the generated byte-code but rather comment on how to make the 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 requirements on the input variables than a setter function.



Coding Best Practices

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



Disclaimer

CFGNINJA 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 advocacy 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 CFGNINJA is under no covenant to audited completeness, accuracy, or solidity of the contracts. In no event will CFGNINJA 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 CFGNINJA 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.

