

Security Assessment:
TG.Bet StakingManager

January 11, 2024

TG.Bet

- Audit Status: **Fail**
- Audit Edition: **Advance**



Project Overview

Token Summary

Parameter	Result
Address	
Name	TG.Bet
Token Tracker	TG.Bet (TGB)
Decimals	18
Supply	100,000,000
Platform	Ethereum
compiler	v0.8.19+commit.7dd6d404
Contract Name	StakingManagerV1
Optimization	Yes with 200 runs
LicenseType	MIT
Language	Solidity
Codebase	https://etherscan.io/ token/0xA908E871B3a70ed2331FcE180AAa871e9536133f#code
Payment Tx	Corporate

Main Contract Assessed Contract Name

Name	Contract	Live
TG.Bet		No

TestNet Contract was Not Assessed

Solidity Code Provided

SolidID	File Sha-1	FileName
TGB	670a6b5e48065ab3e5bda16a870f4f3806a7124f	StakingManagerV1.sol
TGB		
TGB		
TGB		
TGB		
TGB		

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	StakingManagerV1.so	L: 0 C: 0
SWC-101	Pass	Integer Overflow and Underflow.	StakingManagerV1.so	L: 0 C: 0
SWC-102	Pass	Outdated Compiler Version file.	StakingManagerV1.so	L: 0 C: 0
SWC-103	Pass	A floating pragma is set.	StakingManagerV1.so	L: 0 C: 0
SWC-104	Pass	Unchecked Call Return Value.	StakingManagerV1.so	L: 0 C: 0
SWC-105	Pass	Unprotected Ether Withdrawal.	StakingManagerV1.so	L: 0 C: 0
SWC-106	Pass	Unprotected SELFDESTRUCT Instruction	StakingManagerV1.so	L: 0 C: 0
SWC-107	Pass	Read of persistent state following external call.	StakingManagerV1.so	L: 0 C: 0
SWC-108	Pass	State variable visibility is not set..	StakingManagerV1.so	L: 0 C: 0
SWC-109	Pass	Uninitialized Storage Pointer.	StakingManagerV1.so	L: 0 C: 0
SWC-110	Pass	Assert Violation.	StakingManagerV1.so	L: 0 C: 0
SWC-111	Pass	Use of Deprecated Solidity Functions.	StakingManagerV1.so	L: 0 C: 0
SWC-112	Pass	Delegate Call to Untrusted Callee.	StakingManagerV1.so	L: 0 C: 0
SWC-113	Pass	Multiple calls are executed in the same transaction.	StakingManagerV1.so	L: 0 C: 0

ID	Severity	Name	File	location
SWC-114	Pass	Transaction Order Dependence.	StakingManagerV1.so 	L: 0 C: 0
SWC-115	Pass	Authorization through tx.origin.	StakingManagerV1.so 	L: 0 C: 0
SWC-116	Pass	A control flow decision is made based on The block.timestamp environment variable.	StakingManagerV1.so 	L: 0 C: 0
SWC-117	Pass	Signature Malleability.	StakingManagerV1.so 	L: 0 C: 0
SWC-118	Pass	Incorrect Constructor Name.	StakingManagerV1.so 	L: 0 C: 0
SWC-119	Pass	Shadowing State Variables.	StakingManagerV1.so 	L: 0 C: 0
SWC-120	Fail	Potential use of block.number as source of randomness.	StakingManagerV1.so 	L: 35 C: 978,L: 35 C: 1472,L: 35 C: 1888,L: 35 C: 4847,L: 35 C: 4931
SWC-121	Pass	Missing Protection against Signature Replay Attacks.	StakingManagerV1.so 	L: 0 C: 0
SWC-122	Pass	Lack of Proper Signature Verification.	StakingManagerV1.so 	L: 0 C: 0
SWC-123	Pass	Requirement Violation.	StakingManagerV1.so 	L: 0 C: 0
SWC-124	Pass	Write to Arbitrary Storage Location.	StakingManagerV1.so 	L: 0 C: 0
SWC-125	Pass	Incorrect Inheritance Order.	StakingManagerV1.so 	L: 0 C: 0
SWC-126	Pass	Insufficient Gas Griefing.	StakingManagerV1.so 	L: 0 C: 0
SWC-127	Pass	Arbitrary Jump with Function Type Variable.	StakingManagerV1.so 	L: 0 C: 0
SWC-128	Pass	DoS With Block Gas Limit.	StakingManagerV1.so 	L: 0 C: 0

ID	Severity	Name	File	location
SWC-129	Pass	Typographical Error.	StakingManagerV1.so 	L: 0 C: 0
SWC-130	Pass	Right-To-Left-Override control character (U+202E).	StakingManagerV1.so 	L: 0 C: 0
SWC-131	Pass	Presence of unused variables.	StakingManagerV1.so 	L: 0 C: 0
SWC-132	Pass	Unexpected Ether balance.	StakingManagerV1.so 	L: 0 C: 0
SWC-133	Pass	Hash Collisions with Multiple Variable Length Arguments.	StakingManagerV1.so 	L: 0 C: 0
SWC-134	Pass	Message call with hardcoded gas amount.	StakingManagerV1.so 	L: 0 C: 0
SWC-135	Pass	Code With No Effects (Irrelevant/Dead Code).	StakingManagerV1.so 	L: 0 C: 0
SWC-136	Pass	Unencrypted Private Data On-Chain.	StakingManagerV1.so 	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-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?)

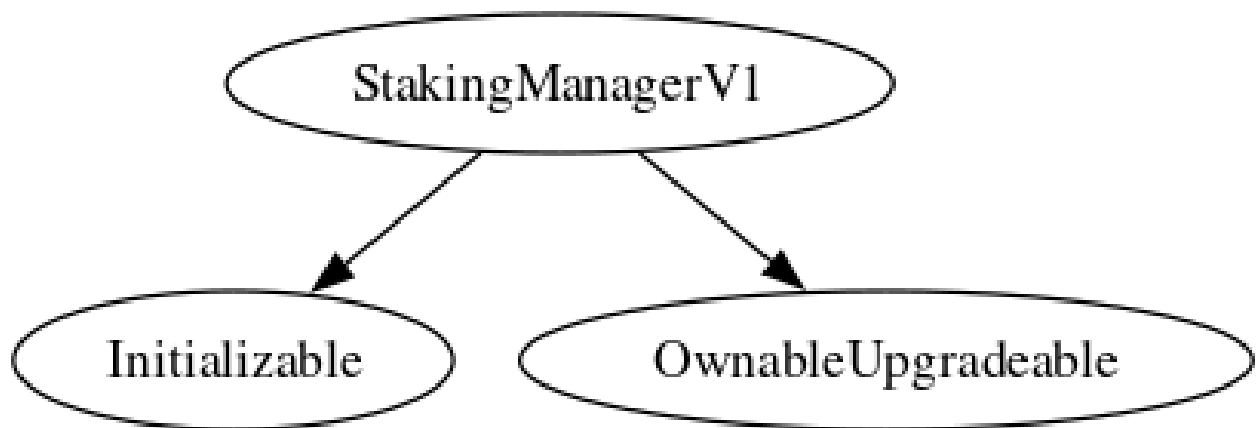
When can BLOCKHASH be safely used for a random number? When would it be unsafe?

The Run smart contract.

Inheritance

The contract for TG.Bet has the following inheritance structure.

The Project has a Total Supply of 100,000,000





Privileged Functions (onlyOwner)

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

Function Name	Parameters	Visibility
setStakeToken		public
setLaunchTime		public
setRewardsPerBlocks		public
setEndBlock		public
setLockTime		public
setPresale		public

TGB-02 | Function Visibility Optimization.

Category	Severity	Location	Status
Gas Optimization	 Informational	StakingManagerV1.sol: L: 230 C: 14,L: 234 C: 14,L: 238 C: 14,L: 242 C: 14,L: 246 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
setStakeToken		public
setLaunchTime		public
setRewardsPerBlocks		public
setEndBlock		public
setLockTime		public
setPresale		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.

TGB-03 | Lack of Input Validation.

Category	Severity	Location	Status
Volatile Code	 Low	StakingManagerV1.sol: L: 230 C: 14,L: 234 C: 14,L: 238 C: 14,L: 242 C: 14,L: 246 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.

TGB-05 | Missing Event Emission.

Category	Severity	Location	Status
Volatile Code	 Low	StakingManagerV1.sol: L: 230 C: 14,L: 234 C: 14,L: 238 C: 14,L: 242 C: 14,L: 246 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.

TGB-19 | `setStakeToken` function should not exist. ..

Category	Severity	Location	Status
Optimization	 Critical	StakingManagerV1.sol: L: 234 C: 47	 Detected

Description



`setStakeToken` function should not exist. This function allows for the owner to change the token that is being staked. This can lead to a potential rug pull.

Remediation

Remove this function.

Project Action

TGB-20 | `setPresale` can lead to critical issues..

Category	Severity	Location	Status
Optimization	 Critical	StakingManagerV1.sol: L: 230 C: 47	 Detected

Description

`setPresale` does not check that launch has already started and does not check for zero-address values. Potentially locking out users depositing from Presale.






Remediation

- Add a check to ensure that the `launchTime` is not already in the past.






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	2	2	0
 High	0	0	0
 Medium	0	0	0
 Low	2	2	0
 Informational	1	1	0
Total	5	5	0

Social Media Checks

Social Media	URL	Result
Twitter	https://twitter.com/Duchapprovn	Pass
Other	https://medium.com/@Duchapprovn	Pass
Website	https://tg.bet	Pass
Telegram	https://t.me/Duchapprovn	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	76/100
Auditor Score	55/100
Review by Section	Score
Manual Scan Score	25
SWC Scan Score	35
Advance Check Score	16

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 84 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 found.␣
- **Minor** - `withdraw` function can lock funds for potentially double the lock time if the deposit happens slightly before `launchTime`. Presale investors are definitely affected by this condition.␣
- **Major** - `setStakeToken` function should not exist. This function allows the owner to change the token that is being staked. This can lead to a potential rug pull.␣
- **Major** - `setPresale` does not check that launch has already started and does not check for zero-address values. Potentially locking out users depositing from Presale␣
- Please review supplemental report.

Auditor Score =55
Audit Fail



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

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 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 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.

