VIBRANIUM AUDITS

Security Assessment

ToolBlox

Verified by Vibranium Audits on 04 September 2023

Revised on 08 September 2023

Vibranium Audits Verified on September 08th, 2023

ToolBlox

The security assessment was prepared by Vibranium Audits.

Executive Summary

TYPES	ECOSYSTEM	METHODS
DEFI	Ethereum	Manual Review, penetration testing and Static Analysis
LANGUAGE	**TIMELINE**	**KEY COMPONENTS**
Solidity	Delivered on 04/09/2023	N/A

CODEBASE

https://github.com/Ideevoog/Toolblox.Token

COMMITS

1aec710c51adef84fbd3e52d279afe363c234c76
cdb28661498b7d50bff4e0b2c799ea2abbda2dec

Vulnerability Summary

11	11	0	0	11	0	0
Total Findings	Resolved	Mitigated	Partially Resolved	Acknowledged	Declined	Unresolved

0 Critical			Critical vulnerabilities are usually straightforward to exploit and can lead to the loss of user funds or contract state manipulation by external or internal actors.
3 High	0 Resolved		High vulnerabilities are usually harder to exploit, requiring specific conditions, or have a more limited scope, but can still lead to the loss of user funds or contract state manipulation by external or internal actors.
3 Medium	0 Resolved		Medium vulnerabilities are usually limited to state manipulations, but cannot lead to assets loss. Major deviations from best practices are also in this category.
3 Low	0 Resolved		Low vulnerabilities are related to outdated and unused code or minor gas optimization. These issues won't have a significant impact on code execution, but affect the code quality.
2 Informational	0 Resolved		Informational errors are often recommendations to improve the style of the code or certain operations to fall within industry best practices. They usually do not affect the overall functioning of the code.

TABLE OF CONTENTS | TOOLBLOX

CODEBASE | TOOLBLOX

Repository

https://github.com/Ideevoog/Toolblox.Token

Commits

1aec710c51adef84fbd3e52d279afe363c234c76
cdb28661498b7d50bff4e0b2c799ea2abbda2dec

AUDIT SCOPE | TOOLBLOX

2 files audited ● 2 file with Acknowledged findings ● 2 files with Resolved findings

ID	Files	Commit Hash
● RVA	RentalWorkflow.sol	1aec710c51adef84fbd3e52d279afe363c234c76
● WVA	WorkflowBase.sol	cdb28661498b7d50bff4e0b2c799ea2abbda2dec

APPROACH & METHODS | TOOLBLOX

This report has been prepared for TOOLBLOX(2023) to discover issues and vulnerabilities in the source code of the TOOLBLOX project as well as any contract dependencies that were not part of an officially recognized library. A comprehensive examination has been performed, utilizing Manual Review, rigorous Penetration Testing and Static Analysis techniques.

The auditing process pays special attention to the following considerations:

- Pen-Testing the smart contracts against both common and uncommon attack vectors.
- Assessing the code base to ensure compliance with current best practices and industry standards
- Ensuring contract logic meets the specifications and intentions of the client.
- Cross referencing contract structure and implementation against similar smart contracts produced by industry leaders.
- Thorough line-by-line manual review of the entire code base by industry experts.

The security assessment resulted in findings that ranged from critical to informational We recommend addressing these findings to ensure a high level of security standards and industry practices.
We suggest recommendations that could better serve the project from the security perspective:

- Testing the smart contracts against both common and uncommon attack vectors.
- Enhance general coding practices for better structures of source codes.
- Review unit tests to cover the possible use cases.
- Review functions for readability, especially for future development work.

FINDINGS | TOOLBLOX

11	3	3	3	2
Total Findings	High	Medium	Low	Informational

This report has been prepared to discover issues and vulnerabilities for TOOLBLOX. Through this audit, we have uncovered 11 issues ranging from different severity levels. Utilizing the techniques of Manual Review, Penetration Testing & Static Analysis to complement rigorous manual code reviews, we discovered the following findings:

ID	Title	Category	Severity	Status
RVA-01	Contract Locking Ethereum	Logical Issue	High	Resolved
RVA-02	Reentrancy-Events	Logical Issue	High	Resolved
WVA-01	Incorrect Conditional Declaration	Logical Issue	Medium	Resolved
RVA-03 WVA-02	Lack of ‘potential’ loss prevention	Logical Issue	Medium	Resolved
RVA-04	Missing Core Functionalities	Logical Issue	High	Resolved
RVA-05	Gas optimization (Modifiers)	Logical Issue	Minor	Resolved
RVA-06 WVA-03	Lack of Documentation	Coding Style	Informational	Resolved
RVA-07 WVA-04	Floating pragma	Logical Issue	Minor	Resolved

RVA-01 | Contract Locking Ethereum

Category	Severity	Location	Status
Logical Issue	High	RentalWorkflow.sol	Resolved

Description

The smart contract **'RentalWorkflow.sol'** implements both the **receive()** and **fallback()** functions, enabling it to receive Ethereum.
However, the smart contract doesn't have a function enabling the withdrawal of received ETH, making it impossible to retrieve deposited ETH in any way.

```
42      receive() external payable {}
43      fallback() external payable {}
```

Recommendation

Implementing a simple **onlyOwner** function to withdraw all of the smart contract's address balance of ETH will solve the problem.
Also, although some of **RentalWorkflow.sol's** functions have the ability to withdraw tokens, it is again recommended to implement a simple **onlyOwner** function that enables the withdrawal of all the addresse's balance of a certain token.

Example:

```
function withdrawFunds() public onlyOwner {
(bool success, ) = msg.sender.call{value: address(this).balance}("");
require(success, "Transfer failed.");
}
```

Revision

The 'ToolBlox' team decided to abandon the receive() functionality in the **RentalWorkflow.sol** contract, as it is found unnecessary and serves no purpose for the contract to receive Ether. This makes the vulnerability obsolete.

MVA-02 | Reentrancy-Events

Category	Severity	Location	Status
Logical Issue	High	RentalWorkflow.sol	Resolved

Description (Finding is present in 5 functions in RentalWorkflow.sol)

Possible manipulation of the order or value of events. This may cause issues for offchain components that rely on the values of events e.g. checking for the amount deposited to a bridge.

Functions: charge(...), endAndSettle(...), finalCharge(...), releaseCollateral(...), startRent(...)

```
function charge(uint256 id) external returns (uint256) {
    Rental memory item = getItem(id);
    _checkOwner();
    _assertStatus(item, 1);
    uint daysToCharge = ( ( block.timestamp - item.startTime ) / ( ( 24 * 60 ) * 60 ) ) - item.daysCharged;
    item.daysCharged = item.daysCharged + daysToCharge;
    item.leftoverCharge = daysToCharge * item.pricePerDay;
    item.status = 1;
    items[id] = item;
    if (owner() != address(0) && item.renter != address(0) && item.leftoverCharge > 0){
        safeTransferFromExternal(token, item.renter, owner(), item.leftoverCharge);
    }
    emit ItemUpdated(id, item.status);
    return id;
}
```

Recommendation

Use the check-effects-interactions pattern.

Recommendation

The 'ToolBlox' team implemented OpenZeppelin's ReentrancyGuard.sol smart contract with the 'nonReentrant' modifier protecting all necessary functionalities against reentrancy.

WVA-01 | Incorrect Conditional Declaration

Categor y	Severity	Location	Status
Logical Issue	Medium	Workflow.sol	Resolved

Description

Workflow.sol's getLatestIds() function includes an 'if' statement that includes a comparison of uint256 variable **toIndex** the value **0.**

Unsigned integers can't have a negative value e.g -1, thus a subtraction that would normally result in a negative value will return 0 (Protection again under/overflowing since solidity 0.8+)

This means that **toIndex < 0** will always return false. Nonetheless, **toIndex** is initialized to the **counter** variable's value, meaning it can only be **0+.**

```
18        if (fromIndex > toIndex || toIndex < 0) {
19            return new uint256[](0);
20        }
```

Recommendation

Simply change the condition to 'toIndex == 0'

```
18        if (fromIndex > toIndex || toIndex == 0) {
19            return new uint256[](0);
20        }
```

Recommendation

Fixed as recommended.

RVA-04 | Missing Core functionalities

Category	Severity	Location	Status
Logical Issue	High	RentawlWorkflow.sol	Resolved

Description

RentalWorkflow.sol's functionalities revolving around registering, starting and ending an item's rental are all under the contract owner's sole control.
There is no specified functionality allowing:

- The startRent(..) function allows an item renter to start a rent, checking that the function caller is the item's renter; how can a rent start if it already has a renter? Also there is no other specified functionality responsible for setting an address as the renter for a certain iteam.

- More importantly, a rent that has started can never be ended according to the renter address needs, this could eventually lead to internal problems where a renter wishes to end a rental but the owner isn't available to end it, leading to financial loss for the renter.

```
109     function startRent(uint256 id,uint64 numberOfDays,uint allowance) external returns (uint256) {
110         Rental memory item = getItem(id);
111         _assertOnlyRenter(item);
112         _assertStatus(item, 3);
113         require(allowance == ( numberOfDays * item.pricePerDay ), "Allowance is correct?");
114         item.numberOfDays = numberOfDays;
115         item.startTime = block.timestamp;
116         item.status = 1;
117         items[id] = item;
118         if (address(this) != address(0) && item.collateral > 0){
119             safeTransferFromExternal(token, _msgSender(), address(this), item.collateral);
120         }
121         emit ItemUpdated(id, item.status);
122         return id;
123     }
```

```
169     function endRental(uint256 id) external returns (uint256) {
170         Rental memory item = getItem(id);
171         _checkOwner();
172         _assertStatus(item, 1);
173         uint nominalFee = ( item.numberOfDays - item.daysCharged ) * item.pricePerDay;
174         uint endTime = item.startTime + ( ( ( item.numberOfDays * 24 ) * 60 ) * 60 );
175         item.leftoverCharge = nominalFee + ( ( block.timestamp > endTime ) ? ( ( endTime - block.timestamp )
176                               * ( ( ( item.pricePerDay / 24 ) / 60 ) / 60 ) ) : 0 );
177         item.status = 2;
178         items[id] = item;
179         emit ItemUpdated(id, item.status);
180         return id;
181     }
```

Recommendation

It is crucial to :

- Modify the startRent(...) function so that it accepts the order of an address to rent a certain item.
- Make the endRental(...) function accessible to both the owner and the current item renter in case they need to end a rent.

Revision

The ToolBlox team intended the modifier _assertOnlyRenter() to function as a 'renter' requirement checker and 'renter' status setter for addresses. All functionalities were untouched as they are coded the way they are intended to be.

RVA-03 WVA-02 | Lack of 'potential' loss prevention

Category	Severity	Location	Status
Logical Issue	Medium	RentalWorkflow.sol WorkflowBase.sol	Resolved

Description

This vulnerability isn't related to any present problems within the code base, but rather a standard measure of prevention against future currently unknown vulnerabilities. All smart contracts with mutative functions that handle sensitive data and/or user/admin funds should implement the ability to be temporarily paused.

This is easily achieved through implementing OpenZeppelin's Pausable Smart Contract.

Reference:
https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts/security/Pausable.sol

Revision

The ToolBlox fixed the vulnerability as recommended by implementing smart contract Pausability.

RVA-07 WVA-04 | Floating Pragma

Category	Severity	Location	Status
Logical Issue	Minor	RentalWorkflow.sol WorkflowBase.sol	Resolved

Description

The current pragma Solidity directive is ""^0.8.0"". It is recommended to specify a fixed compiler version to ensure that the bytecode produced does not vary between builds. This is especially important if you rely on bytecode-level verification of the code.

Recommendation

Use a new specific stable Solidity version e.g 0.8.18, although avoid using the latest version avoiding any potential yet undiscovered bugs.

Revision

The ToolBlox team fixed the contracts' Pragma version to 0.8.19

GVA-01 | Gas Optimization (Modifiers)

Category	Severity	Location	Status
Logical Issue	Minor	RentawlWorkflow.sol	Resolved

Description

RentalWorkflow.sol relies on dedfined private functions to enforce some conditions e.g:

```
30    function _assertOnlyRenter(Rental memory item) private view {
31        address renter = item.renter;
32        if (renter != address(0))
33        {
34            require(_msgSender() == renter, "Invalid Renter");
35            return;
36        }
37        item.renter = _msgSender();
38    }
```

Recommendation

For the sole purpose of providing more optimal gas usage, it is recommended to specify the conditional functions and use them as modifiers (e.g use onlyOwner modfier instead of '_checkOwner()'

Revision

The ToolBlox team implemented the onlyOwner modifier for owner() access control, but left _assertOnlyRenter() as it is because it is intended to access and modify state.

RVA-06 WVA-03 Lack of documentation

Categor y	Severity	Location	Status
Logical Issue	Informational	RentalWorkflow.sol WorkflowBase.sol	Resolved

Description

Althought the TOOLBLOX contract set features some documentation on the main contract RentawlWorkflow.sol, it is recommended to fully document all the written code for better future communication between developers, auditors and any other involved party.

Revision

The ToolBlox team successfully provided detailed and tailored documentation within their smart contracts rendering them more readable, understandable and easier to be communicated.

DISCLAIMER | VIBRANIUM AUDITS

This report is subject to the terms and conditions (including without limitation description of services confidentiality disclaimer and limitation of liability) set forth in the Services Agreement or the scope of services and terms and conditions provided to you (“Customer” or the “Company”) in connection with the Agreement This report provided in connection with the Services set forth in the Agreement shall be used by the Company only to the extent permitted under the terms and conditions set forth in the Agreement This report may not be transmitted disclosed referred to or relied upon by any person for any purposes nor may copies be delivered to any other person other than the Company without Vibranium Audits prior written consent in each instance

This report is not nor should be considered an “endorsement” or “disapproval” of any particular project or team This report is not nor should be considered an indication of the economics or value of any “product” or “asset”created by any team or project that contracts Vibranium Audits to perform a security assessment This report does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed nor do they provide any indication of the technologies proprietors business business model or legal compliance

This report should not be used in any way to make decisions around investment or involvement with any particular project This report in no way provides investment advice nor should be leveraged as investment advice of any sort This report represents an extensive assessing process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology

Blockchain technology and cryptographic assets present a high level of ongoing risk Vibranium Audits position is that each company and individual are responsible for their own due diligence and continuous security Vibranium Audits goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies and in no way claims any guarantee of security or functionality of the technology we agree to analyze

The assessment services provided by Vibranium Audits 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 Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty The assessment reports could include false positives false negatives and other unpredictable results The services may access and depend upon multiple layers of third-parties

ALL SERVICES THE LABELS THE ASSESSMENT REPORT WORK PRODUCT OR OTHER MATERIALS OR ANY PRODUCTS OR RESULTS OF THE USE THEREOF ARE PROVIDED “AS IS” AND “AS AVAILABLE” AND WITH ALL FAULTS AND DEFECTS WITHOUT WARRANTY OF ANY KIND TO THE MAXIMUM EXTENT PERMITTED UNDER APPLICABLE LAW VIBRANIUM AUDITS HEREBY DISCLAIMS ALL WARRANTIES WHETHER EXPRESS IMPLIED STATUTORY OR OTHERWISE WITH RESPECT TO THE SERVICES ASSESSMENT REPORT OR OTHER MATERIALS WITHOUT LIMITING THE FOREGOING VIBRANIUM AUDITS SPECIFICALLY DISCLAIMS ALL IMPLIED WARRANTIES OF MERCHANTABILITY FITNESS FOR A PARTICULAR PURPOSE TITLE AND NON-INFRINGEMENT AND ALL WARRANTIES ARISING FROM COURSE OF DEALING USAGE OR TRADE PRACTICE WITHOUT LIMITING THE FOREGOING VIBRANIUM AUDITS MAKES NO WARRANTY OF ANY KIND THAT THE SERVICES THE LABELS THE ASSESSMENT REPORT WORK PRODUCT OR OTHER MATERIALS OR ANY PRODUCTS OR RESULTS OF THE USE THEREOF WILL MEET CUSTOMER’S OR ANOTHER PERSON’S REQUIREMENTS ACHIEVE ANY INTENDED RESULT BE COMPATIBLE OR WORK WITH ANY SOFTWARE SYSTEM OR OTHER SERVICES OR BE SECURE ACCURATE COMPLETE FREE OF HARMFUL CODE

OR ERROR-FREE WITHOUT LIMITATION TO THE FORGOING, VIBRANIUM AUDITS PROVIDES NO WARRANTY OR UNDERTAKING, AND MAKES NO REPRESENTATION OF ANY KINDTHAT WILL MEET CUSTOMER' S REQUIREMENTS ACHIEVE ANY INTENDED RESULTS BE COMPATIBLE OR WORK WITH ANY OTHER SOFTWARE APPLICATIONS SYSTEMS OR SERVICES OPERATE WITHOUT INTERRUPTION MEET ANY PERFORMANCE OR RELIABILITY STANDARDS OR BE ERROR FREE OR THAT ANY ERRORS OR DEFECTS CAN OR WILL BE CORRECTED

WITHOUT LIMITING THE FOREGOING NEITHER VIBRANIUM AUDITS NOR ANY OF VIBRANIUM AUDITS AGENTS MAKES ANY REPRESENTATION OR WARRANTY OF ANY KIND EXPRESS OR IMPLIED AS TO THE ACCURACY RELIABILITY ORCURRENCY OF ANY INFORMATION OR CONTENT PROVIDED THROUGH THE SERVICE VIBRANIUM AUDITS WILL ASSUME NO LIABILITY OR RESPONSIBILITY FOR (I) ANY ERRORS MISTAKES OR INACCURACIES OF CONTENT AND MATERIALS OR FOR ANY LOSS OR DAMAGE OF ANY KIND INCURRED AS A RESULT OF THE USE OF ANY CONTENT OR (II) ANY PERSONAL INJURY OR PROPERTY DAMAGE OF ANY NATURE WHATSOEVER RESULTING FROM CUSTOMER'S ACCESS TO OR USE OF THE SERVICES ASSESSMENT REPORT OR OTHER MATERIALS

ALL THIRD-PARTY MATERIALS ARE PROVIDED "AS IS" AND ANY REPRESENTATION OR WARRANTY OF OR CONCERNING ANY THIRD-PARTY MATERIALS IS STRICTLY BETWEEN CUSTOMER AND THE THIRD-PARTY OWNEROR DISTRIBUTOR OF THE THIRD-PARTY MATERIALS

THE SERVICES ASSESSMENT REPORT AND ANY OTHER MATERIALS HERE UNDER ARE SOLELY PROVIDED TO CUSTOMER AND MAY NOT BE RELIED ON BY ANY OTHER PERSON OR FOR ANY PURPOSE NOT SPECIFICALLY IDENTIFIED IN THIS AGREEMENT NOR MAY COPIES BE DELIVERED TO ANY OTHER PERSON WITHOUT VIBRANIUM AUDITS PRIOR WRITTEN CONSENT IN EACH INSTANCE

NO THIRD PARTY OR ANYONE ACTING ON BEHALF OF ANY THEREOF SHALL BE A THIRD PARTY OR OTHER BENEFICIARY OF SUCH SERVICES ASSESSMENT REPORT AND ANY ACCOMPANYING MATERIALS AND NO SUCH THIRD PARTY SHALL HAVE ANY RIGHTS OF CONTRIBUTION AGAINST VIBRANIUM AUDITS WITH RESPECT TO SUCH SERVICES ASSESSMENT REPORT AND ANY ACCOMPANYING MATERIALS

THE REPRESENTATIONS AND WARRANTIES OF VIBRANIUM AUDITS CONTAINED IN THIS AGREEMENT ARE SOLELY FOR THE BENEFIT OF CUSTOMER ACCORDINGLY NO THIRD PARTY OR ANYONE ACTING ON BEHALF OF ANY THEREOF SHALL BE A THIRD PARTY OR OTHER BENEFICIARY OF SUCH REPRESENTATIONS AND WARRANTIES AND NO SUCH THIRD PARTY SHALL HAVE ANY RIGHTS OF CONTRIBUTION AGAINST VIBRANIUM AUDITS WITH RESPECT TO SUCH REPRESENTATIONS OR WARRANTIES OR ANY MATTER SUBJECT TO OR RESULTING IN INDEMNIFICATION UNDER THIS AGREEMENT OR OTHERWISE

FOR AVOIDANCE OF DOUBT THE SERVICES INCLUDING ANY ASSOCIATED ASSESSMENT REPORTS ORMATERIALS SHALL NOT BE CONSIDERED OR RELIED UPON AS ANY FORM OF FINANCIAL, TAX, LEGAL, REGULATORY, OR OTHER ADVICE.

Vibranium | Securing the Web3 World

Vibranium Audits is a blockchain security company that was founded in 2021 by professors from the University of Greenwich and cyber-security engineers from ITI Capital. As pioneers in the field, Vibranium Audits utilizes best-in-class Formal Verification and AI technology to secure and monitor blockchains, smart contracts, and Web3 apps.