



# AUDIT REPORT

---

August , 2025

For

# Table of Content

Table of Content	02
Executive Summary	03
Number of Issues per Severity	05
Checked Vulnerabilities	06
Techniques & Methods	08
Types of Severity	10
Types of Issues	11
<b>■ Informational Severity Issues</b>	12
1. Inappropriate Type Casting in clock() Function	12
Functional Tests	13
Closing Summary & Disclaimer	14

# Executive Summary

<b>Project name</b>	X-Verse (HTT)
<b>Overview</b>	<p>Name: X-Verse Symbol: HTT Initial Supply: 18 billion tokens (18,000,000,000) Decimals: 18 (standard)</p> <p>X-Verse (HTT) is an upgradeable ERC-20 governance token with 18 billion initial supply that supports voting, gasless transactions via permits, and emergency pause functionality. The contract uses OpenZeppelin's UUPS proxy pattern for upgrades and AccessManager for role-based permissions. Administrative functions include minting new tokens, pausing/unpausing transfers, and authorizing contract upgrades - all restricted to authorized addresses.</p>
<b>Audit Scope</b>	<a href="https://github.com/bcharityfi/realwillingalliance/blob/main/contracts/htt/Htt.sol">https://github.com/bcharityfi/realwillingalliance/blob/main/contracts/htt/Htt.sol</a>
<b>Branch</b>	main
<b>Contracts in Scope</b>	HTT
<b>Commit Hash</b>	bd30eba6e11ba2de37a2b7bba62bbafb73936116
<b>Language</b>	Solidity
<b>Blockchain</b>	EVM
<b>Method</b>	Manual Analysis, Functional Testing, Automated Testing
<b>Review 1</b>	18th August 2025

**Updated Code Received** NA

**Review 2** NA

**Fixed In** NA

# Number of Issues per Severity



High	0 (0.00%)
Medium	0 (0.00%)
Low	0 (0.00%)
Informational	1(100.00%)

Issues	Severity			
	High	Medium	Low	Informational
Open	0	0	0	0
Resolved	0	0	0	0
Acknowledged	0	0	0	1
Partially Resolved	0	0	0	0

# Checked Vulnerabilities

<input checked="" type="checkbox"/> Access Management	<input checked="" type="checkbox"/> Compiler version not fixed
<input checked="" type="checkbox"/> Arbitrary write to storage	<input checked="" type="checkbox"/> Address hardcoded
<input checked="" type="checkbox"/> Centralization of control	<input checked="" type="checkbox"/> Divide before multiply
<input checked="" type="checkbox"/> Ether theft	<input checked="" type="checkbox"/> Integer overflow/underflow
<input checked="" type="checkbox"/> Improper or missing events	<input checked="" type="checkbox"/> ERC's conformance
<input checked="" type="checkbox"/> Logical issues and flaws	<input checked="" type="checkbox"/> Dangerous strict equalities
<input checked="" type="checkbox"/> Arithmetic Computations Correctness	<input checked="" type="checkbox"/> Tautology or contradiction
<input checked="" type="checkbox"/> Race conditions/front running	<input checked="" type="checkbox"/> Return values of low-level calls
<input checked="" type="checkbox"/> SWC Registry	<input checked="" type="checkbox"/> Missing Zero Address Validation
<input checked="" type="checkbox"/> Re-entrancy	<input checked="" type="checkbox"/> Private modifier
<input checked="" type="checkbox"/> Timestamp Dependence	<input checked="" type="checkbox"/> Revert/require functions
<input checked="" type="checkbox"/> Gas Limit and Loops	<input checked="" type="checkbox"/> Multiple Sends
<input checked="" type="checkbox"/> Exception Disorder	<input checked="" type="checkbox"/> Using suicide
<input checked="" type="checkbox"/> Gasless Send	<input checked="" type="checkbox"/> Using delegatecall
<input checked="" type="checkbox"/> Use of tx.origin	<input checked="" type="checkbox"/> Upgradeable safety
<input checked="" type="checkbox"/> Malicious libraries	<input checked="" type="checkbox"/> Using throw

Using inline assembly

Unsafe type inference

Style guide violation

Implicit visibility level

# Techniques and Methods

Throughout the audit of smart contracts, care was taken to ensure:

- The overall quality of code.
- Use of best practices.
- Code documentation and comments, match logic and expected behavior.
- Token distribution and calculations are as per the intended behavior mentioned in the whitepaper.
- Implementation of ERC standards.
- Efficient use of gas.
- Code is safe from re-entrancy and other vulnerabilities.

**The following techniques, methods, and tools were used to review all the smart contracts.**

## Structural Analysis

In this step, we have analyzed the design patterns and structure of smart contracts. A thorough check was done to ensure the smart contract is structured in a way that will not result in future problems.

## Static Analysis

A static Analysis of Smart Contracts was done to identify contract vulnerabilities. In this step, a series of automated tools are used to test the security of smart contracts.

**Code Review / Manual Analysis**

Manual Analysis or review of code was done to identify new vulnerabilities or verify the vulnerabilities found during the static analysis. Contracts were completely manually analyzed, their logic was checked and compared with the one described in the whitepaper. Besides, the results of the automated analysis were manually verified.

**Gas Consumption**

In this step, we have checked the behavior of smart contracts in production. Checks were done to know how much gas gets consumed and the possibilities of optimization of code to reduce gas consumption.

**Tools And Platforms Used For Audit**

Remix IDE, Foundry, Solhint, Mythril, Slither, Solidity statistical analysis.

# Types of Severity

Every issue in this report has been assigned to a severity level. There are four levels of severity, and each of them has been explained below

## ● High Severity Issues

A high severity issue or vulnerability means that your smart contract can be exploited. Issues on this level are critical to the smart contract's performance or functionality, and we recommend these issues be fixed before moving to a live environment.

## ■ Medium Severity Issues

The issues marked as medium severity usually arise because of errors and deficiencies in the smart contract code. Issues on this level could potentially bring problems, and they should still be fixed.

## ● Low Severity Issues

Low-level severity issues can cause minor impact and are just warnings that can remain unfixed for now. It would be better to fix these issues at some point in the future.

## ■ Informational Issues

These are four severity issues that indicate an improvement request, a general question, a cosmetic or documentation error, or a request for information. There is low-to-no impact.

# Types of Issues

<b>Open</b>  Security vulnerabilities identified that must be resolved and are currently unresolved.	<b>Resolved</b>  Security vulnerabilities identified that must be resolved and are currently unresolved.
<b>Acknowledged</b>  Vulnerabilities which have been acknowledged but are yet to be resolved.	<b>Partially Resolved</b>  Considerable efforts have been invested to reduce the risk/ impact of the security issue, but are not completely resolved.

# Informational Severity Issues

## Inappropriate Type Casting in `clock()` Function

Acknowledged

**Path**

Rwa.sol

**Function**

`clock()`

**Description**

-

**Recommendation**

Use `uint64` instead of `uint48`

# Functional Tests

**Some of the tests performed are mentioned below:**

- ✓ Upgradability is working as expected
- ✓ Transfers are restricted when paused
- ✓ Tested minting with proper authorization
- ✓ Tested pause/unpause functionality with proper permissions
- ✓ Tested unauthorized access rejection

# Automated Tests

No major issues were found. Some false positive errors were reported by the tools. All the other issues have been categorized above according to their level of severity.

# Closing Summary

In this report, we have considered the security of X-Verse ( HTT). We performed our audit according to the procedure described above.

One issue of Informational severity was found, X-Verse (HTT) team acknowledged that issue

# Disclaimer

At QuillAudits, we have spent years helping projects strengthen their smart contract security. However, security is not a one-time event—threats evolve, and so do attack vectors. Our audit provides a security assessment based on the best industry practices at the time of review, identifying known vulnerabilities in the received smart contract source code.

This report does not serve as a security guarantee, investment advice, or an endorsement of any platform. It reflects our findings based on the provided code at the time of analysis and may no longer be relevant after any modifications. The presence of an audit does not imply that the contract is free of vulnerabilities or fully secure.

While we have conducted a thorough review, security is an ongoing process. We strongly recommend multiple independent audits, continuous monitoring, and a public bug bounty program to enhance resilience against emerging threats.

Stay proactive. Stay secure.



# About QuillAudits

QuillAudits is a leading name in Web3 security, offering top-notch solutions to safeguard projects across DeFi, GameFi, NFT gaming, and all blockchain layers. With seven years of expertise, we've secured over 1400 projects globally, averting over \$3 billion in losses. Our specialists rigorously audit smart contracts and ensure DApp safety on major platforms like Ethereum, BSC, Arbitrum, Algorand, Tron, Polygon, Polkadot, Fantom, NEAR, Solana, and others, guaranteeing your project's security with cutting-edge practices.



Follow Our Journey



# AUDIT REPORT

---

August , 2025

For



Canada, India, Singapore, UAE, UK

[www.quillaudits.com](http://www.quillaudits.com)    [audits@quillaudits.com](mailto:audits@quillaudits.com)