Security Audit – Wormhole v2

Neodyme AG

January 10, 2022

Contents

Introduction

Neodyme was engaged to do a security review of the Wormhole Bridge. The review took place in the months leading up to the Wormhole v2 launch, beginning in July 2021 and ending in September 2021. Wormhole v2 was launched after the review concluded on the 17th of September 2021.

The review revealed three significant vulnerabilities as well as some medium and low-priority findings, all of which were resolved before the bridge went live.

In this report, we outline the most relevant findings of our initial reviews. Neodyme is currently performing another comprehensive audit of all Wormhole components and has also been contracted to conduct ongoing audits on a regular basis.

Project Overview

> The Wormhole Token Bridge allows you to seamlessly transfer tokenized assets across Solana, Ethereum, BSC, Terra, Polygon, Avalanche, and Oasis.

At its core, Wormhole v2 allows passing messages across chains. This requires two things: a Wormhole smart contract on each supported chain and a network of guardians that provide attestations for all Wormhole messages. These attestations can be verified on any supported chain, minimizing the required trust in any single entity. Each guardian is semi-trusted and is responsible for watching for messages from the corresponding on-chain Wormhole contracts. When it sees one, it signs it. There is a separate p2p network used to exchange and gather signatures of multiple guardians.

On top of this data-transfer layer, token-bridges are built. Token bridges are separate contracts and also have to exist on each supported chain. To transfer assets between a native and a target chain, Wormhole locks the assets on the native chain inside the token-bridge and generates a transfer message. With an attestation of this message, a user can cause the token-bridge on the target chain to mint the corresponding wrapped tokens. The holder of the wrapped assets can, at any time, transfer them back to the bridge and thus obtain the corresponding amount of locked tokens on the native chain.

At launch, Wormhole supported Solana, Terra, Ethereum, and BSC, where the latter two are both EVM-based and use the same contracts. Taking everything into account, there were seven separate components relevant for security at launch: the main bridge and token-bridge on each of the three supported chain types, and the guardian infrastructure.

More detailed information on Wormhole can be found in its documentation.

Scope

As the initial review was done as a part of the Solana peer review process, the focus was placed on the Solana contracts. But Wormhole's other parts were also inspected. Neodyme therefore reviewed the Solana, Ethereum, and Terra bridges and token-bridges, as well as the guardian implementation. All code is contained in the Wormhole monorepo: github.com/certusone/wormhole. As parts of the bridge were still under development during our reviews, and the Wormhole team quickly resolved any found issues, there is no single commit where all described bugs are present. The relevant code was contained in the following folders:

- `/solana/solitaire/`: a Solana smart-contract framework, which is used in all Solana Wormhole contracts
- `/solana/bridge/`: the main Wormhole bridge on Solana
- `/solana/modules/token_bridge/`: the Wormhole token-bridge on Solana
- `/ethereum/contracts/`: all Ethereum contracts
- `/terra/contracts/wormhole/`: main Terra bridge contract
- `/terra/contracts/token-bridge/`: Terra token bridge contract
- `/node/`: go implementation of a guardian node

Methodology

Neodyme's audit team, which consists of security engineers with extensive experience in Solana smart contract security, reviewed and tested the code of the seven on- and off-chain components, paying particular attention to the following:

- Ruling out common classes of Solana contract vulnerabilities, such as:
  - Missing ownership checks
  - Missing signer checks
  - Signed invocation of unverified programs
  - Solana account confusions
  - Re-initiation with cross-instance confusion
  - Missing freeze authority checks
  - Insufficient SPL token account verification
  - Missing rent exemption assertion
  - Casting truncation
  - Arithmetic over- or underflows
  - Numerical precision errors
- Checking for unsafe design, which might lead to common vulnerabilities being introduced in the future
- Checking for any other, as-of-yet unknown classes of vulnerabilities arising from the structure of the Solana blockchain
- Ensuring that the contract logic correctly implements the project specifications
- Examining the code in detail for contract-specific low-level vulnerabilities
- Ruling out denial-of-service attacks
- Ruling out economic attacks
- Checking for instructions that allow front-running or sandwiching attacks
- Checking for rug-pull mechanisms or hidden backdoors
- Checking for replay protection

Findings

All findings are classified in one of four severity levels:

- **Critical**: Bugs that will likely cause loss of funds. This means that an attacker can, with little or no preparation, trigger them, or they are expected to happen accidentally. Effects are difficult to undo after they are detected.
- **High**: Bugs, which can be used to set up a loss of funds in a more limited capacity, or to render the contract unusable.
- **Medium**: Bugs that do not cause direct loss of funds but lead to other exploitable mechanisms.
- **Low**: Bugs that do not have a significant immediate impact and could be fixed easily after detection.

Name	Severity	Status
Solana – Solitaire not Executing Verification Methods	Critical	Resolved
Terra – Replay Attack	Critical	Resolved
Solana Token Bridge – Target not Checked	High	Resolved
Solana Bridge – Removed Guardians can DoS	Medium	Resolved
Solana Bridge – Missing Governance Checks	Medium	Resolved
Terra Token Bridge – Missing Chain Registration Checks	Low	Resolved

Solana – Solitaire not Executing Verification Methods (Critical; Resolved)

Severity	Impact	Affected Component	Status
Critical	Likely Loss of Funds	Solana Contracts	Resolved

The two Solana contracts (bridge and token-bridge) use the custom Solitaire framework. The framework supports user-defined account types. Runtime checks on these accounts were intended to be placed inside a user-defined `verify()` function. These functions were intended to be automatically called when an account was used. The code generated via Solitaire did not execute these verify functions. Thus, many account-constraint checks Solitaire should be doing in the bridges were omitted, which could likely be used to cause a loss of funds.

The root cause for this bug was erroneous automatic code generation by Solitaire. As a first fix, the verify functions were called correctly. Later, the framework was restructured, removing the verify functions in favor of explicit account checks inside the contracts' handlers. Neodyme verified the fix.

Terra – Replay Attack (Critical; Resolved)

Severity	Impact	Affected Component	Status
Critical	Loss of Funds	Terra Bridges	Resolved

The Terra bridge implementations were missing replay protection. A transfer message from another chain could be executed arbitrarily often.

The mechanism for replay protection was implemented, just not called correctly. A first change adding the relevant call to vaa_archive_add was incomplete since it only applied to non-governance messages, which are special-cased very early in the code. This was fixed, and all messages are now replay protected.

Solana Token Bridge – Target not Checked (High; Resolved)

Severity	Impact	Affected Component	Status
High	In-flight transfers can be redirected	Solana Token Bridge	Resolved

The target validation in the Solana token bridge was insufficient. A token bridge message includes both a `to` address, as well as a field specifying on which chain the receiver is. Both were unchecked, leading to two issues:

First, the missing receiver-address check allowed an attacker to redeem an in-flight token transfer to his own wallet instead of the specified receiver. Since each message can only be redeemed once, this only affected in-flight transfers.

Second, the missing check on `vaa.to_chain` made it possible to redeem messages meant for other chains on Solana. This bug is not obvious to exploit since the receiving account must be a valid SPL-Token account for the correct token mint. However, other chains might be more lenient with invalid transfer targets. An attacker could send a transfer targeting such a lenient chain at a known-good Solana address. The same transfer would be redeemable on two chains, once on Solana with a good address, so the attacker actually gets his tokens, and second on the lenient chain. At the very least, this would skew the token supply.

Both issues were fixed by adding the appropriate checks, which Neodyme verified.

Solana Bridge – Removed Guardians can DoS (Medium; Resolved)

Severity	Impact	Affected Component	Status
Medium	Old, potentially compromised guardians can freeze transactions	Solana Bridge	Resolved

The signature verification inside the Solana bridge in `verify_signatures` was vulnerable to a Denial of Service (DoS) attack from old, potentially compromised, and removed guardians. The signature verification is built to gather all guardian signatures into a single account since they are too large to fit in a single transaction. A message can only be executed once such a signature set is complete. In an early version of Wormhole v2, there could only exist one such signature set for each message. An old guardian could block these signature set accounts, which would make it impossible to execute new messages.

All current guardians are stored in a guardian-set, and old guardian-sets also persist on-chain. When starting signature verification, the user specifies a guardian-set to use. At least one valid signature for any of the guardians in the set is required. Thus, any past guardian can, at a time of their choosing, create a signature that can be used for a new signature account with an old guardian set. This is an issue as old guardian sets are not accepted when executing a message. Since the created account has the hash of the message as account-seeds, no other signature set for a newer guardian-set could ever be created. The past guardian could now front-run all legit signature accounts and block their creation with his own old signature account.

The issue was fixed, and it is now possible to have a signature account at arbitrary addresses a user can sign with, so no blocking is possible. Neodyme verified the fix.

Solana Bridge – Missing Governance Checks (Medium; Resolved)

Severity	Impact	Affected Component	Status
Medium	One governance action might be used for another.	Solana Bridge	Resolved

In Wormhole, all governance actions are done through messages from a white-listed governance address. In this message, the governance action is specified in a header, followed by the governance parameters. Most governance functions in the Solana bridge did not check this governance header correctly. Thus one governance message can be used with a different endpoint to make something else happen, like using an “Upgrade” message to change the fees. But since a governance message still has to come from the correct sender and then could not be used again for the intended action, such an attack would likely have been noticed quickly.

All governance functions now correctly use check_governance_header. In addition, the payload lengths are now checked. Neodyme verified the fix.

Terra Token Bridge – Missing Chain Registration Checks (Low; Resolved)

Severity	Impact	Affected Component	Status
Low	Invalid Token Metadata	Terra Token Bridge - Registration	Resolved

The Wormhole token bridges must register all tokens before they are first used to avoid passing token metadata on each transfer. The chain a token resides on attests the metadata like decimals, and all other chains can then create their according wrapped token. The Terra implementation insufficiently checked these token registration messages:

Only the transfer messages checked that the sender is one of the expected foreign chain token-bridge contracts. The token registration was missing this check. Thus anybody could register made-up tokens or tokens with invalid decimals by crafting the corresponding attestation message themselves. The issue was fixed by checking the message sender, which Neodyme verified.

Neodyme AG

Dirnismaning 55
Halle 13
85748 Garching
E-Mail: contact@neodyme.io

https://neodyme.io