POA Bridge

Security Assessment

Cris Neckar
SECUREWARE.IO

Executive Summary

The engagement team performed a limited scope, source code assessment of POA Network's POA Bridge. The purpose of this assessment was to identify vulnerabilities within the POA Bridge that could allow an attacker to negatively impact the bridge or associated contracts. The primary goal of the assessment team was to identify vulnerabilities that could contribute to any of the following:

- The compromise of one or more bridge nodes
- Compromising Proof of Authority from the context of a malicious bridge
- Causing a denial of service condition, preventing bridges from relaying or approving transactions
- Locking value in either the Home or Foreign Bridge contracts
- Causing the unintended transfer of Ether or POA tokens from the Home or Foreign Bridge contracts

No serious vulnerabilities were identified in this assessment. However, several deployment recommendations are discussed in this report that are critical to the security of a validator Bridge and to maintaining the protections offered from Proof of Authority consensus.

This assessment was not intended as a comprehensive security assessment of the POA Bridge Ethereum contracts. Due to the limited scope of the assessment, the findings presented in this report should not be considered to be exhaustive.

Threat Model

Publicly Exposed Attack Surface

The primary attack surface of the POA Bridge takes the form of Ethereum log entries passed to the Bridge via the configured Ethereum nodes on the Home and Foreign Ethereum networks. These log entries are generated, by the Home and Foreign Bridge contracts on their respective networks, in response to user-controlled transactions (specifically any payable transaction to the Home Bridge contract or a user-triggered call to onTokenTransfer() via the configured ERC677 on the Foreign Bridge contract). The user-controllable fields of these log entries are limited to the originating address and the value of the transaction in the case of the Home Bridge Deposit() event, and the ERC677 proxied address "from" and uint256 "value" in the case of the Foreign Bridge Withdraw() event. Since these inputs are simple and well defined, the publicly exposed attack surface of the POA Bridge is extremely limited. No vulnerabilities were identified that could be exploited via publicly accessible attack surface.

Additional Attack Surface

The POA Bridge exposes additional attack surface to the configured Ethereum Home and Foreign nodes, as well as to any attacker who is able to compromise these nodes' connections to the Home or Foreign Ethereum networks or the communication channel between the POA Bridge and either of these nodes. An attacker could gain access to this additional attack surface via any of the following attacks:

- Compromising one or both of the configured nodes
- DNS cache poisoning attacks (causing the Bridge to communicate directly with an attacker-controlled node)
- MITM network-based attacks (allowing the attacker to view or modify communication between the Home or Foreign node and the Bridge)

Discussion of Identified Risks

Malicious Transaction - Home Network

As discussed previously, the attack surface exposed by the Home Contract (and by extension, the Bridge) is extremely minimal. Specifically, only two external methods may be called by an arbitrary address.

The first is the payable fallback method. This method triggers a Deposit event that is parsed within deposit_relay_payload() of the Bridge. The low-level parsing of raw log entries occurs in Parity's ethabi library in the function parse_log(). An audit of this portion of the ethabi codebase did not uncover any vulnerabilities. The parsed values are then used to create a Transaction using the ethcore_transaction library and sent to the Foreign node via rust-web3. No vulnerabilities were identified in this control flow.

The second method that can be called by an arbitrary account is withdraw(). This method is intended to be called by the validator Bridge responsible for relaying a withdrawal to the Home network. Although no vulnerabilities were identified within this method, it is recommended that access be limited to only validators. This could be trivially achieved by performing the following check:

```
require(validatorContract().isValidator(msg.sender));
```

Malicious Transaction - Foreign Network

The attack surface exposed by the Foreign Contract (and by extension, the Bridge) is even more limited than that of the Home contract. The contract does not expose any external methods directly to arbitrary accounts. Transactions are instead generated through the ERC677

contract's call to onTokenTransfer(). This call contains three user-controlled values including the address, value to be transferred, and arbitrary user-supplied bytes of context. The arbitrary context bytes are ignored and only the address and value are included in the generated Withdraw() event. These values are then parsed and included in subsequent transactions in the same way described previously. No vulnerabilities were identified in the code that handles these values.

Compromised Ethereum Node - Home Network

Assuming that an attacker is able to compromise the configured Home network Ethereum node, the attack surface of the Bridge increases significantly. Specifically, the compromised node can directly attack the rust-web3 RPC channel, JSON parser, parsing of RawLog entries, or simply send invalid log entries to a Bridge that represent transactions that have not actually occurred on the Home network.

A comprehensive assessment of the third-party attack surface exposed to a compromised node could not be performed in the limited scope of this assessment. However, it is recommended that controls be implemented to minimize the risk of such an attack. Specifically, all nodes configured within the validator Bridges should use a secure communication channel (TLS). It is recommended that the Bridge enforce the use of HTTPS protocol or only allow insecure transport when compiled with a flag for testing purposes.

The Proof of Authority consensus algorithm was found to be robust against attempted abuses by a single compromised Ethereum node. However, it is critical that a quorum of Bridges do not connect to the same node. This would create a single point of failure allowing the Proof of Authority consensus to be undermined.

Compromised Ethereum Node - Foreign Network

The risks associated with the compromise of a configured Foreign network Ethereum node are similar to those of a Home network node. Key recommendations are the enforced use of a secure transport (TLS) and ensuring that validator Bridges do not use the same Ethereum node.

Denial of Service Attacks on Ethereum Node or Bridge

A number of scenarios exist in which an attacker could prevent one or more validator Bridges from receiving the event logs related to deposits or withdrawals. Although it is likely impossible to fully mitigate the risk of such a scenario, the impact can be greatly decreased through the implementation of a few simple recommendations. Similar to issues surrounding a compromised Ethereum node, it is critical that a single point of failure be avoided. The various validator Bridges should use a disparate set of Ethereum nodes on both the Home and Foreign

networks. No quorum of validators should ever use the same Ethereum node on either the Home or Foreign networks.

Compromise of a Validator Bridge

The purpose of the Proof of Authority consensus mechanism is to ensure that no single malicious or compromised validator can single-handedly misappropriate value. Although, the critical consensus checks are performed within the Home and Foreign contracts, which were not in scope for this testing, these checks were reviewed and no vulnerabilities were identified that would undermine the POA consensus.

Third-Party Certificate Validation

One of the key recommendations of this report is that a secure transport be used between the POA Bridges and the various Ethereum nodes on each network. In order for this measure to be effective, it is critical that the library responsible for this communication implement robust certificate validation.

The POA Bridge currently uses the rust-web3 library for communication between the Bridge and Ethereum nodes. This library, in turn, uses the hyper_tls library for SSL support. The hyper_tls library was tested and was found to perform adequate certificate validation. The certificate is verified to chain to a valid CA and the hostname is validated.

Use of Shared Validator Account on Home and Foreign Networks

One of the requests provided by POA Networks for this assessment was to analyze the risk associated with using a single Ethereum account (private key and address) as the validator account on both the Home and Foreign networks. Using a single account decreases the complexity of the Bridge and simplifies configuration.

No significant risks were identified with this proposal.

Discussion of Recommendations

Require Secure RPC Transport

In order to prevent the hijacking of communication between the Bridge and the configured Ethereum nodes it is critical that all communication is performed via a secure transport

mechanism. The Bridge should enforce a policy that requires Ethereum nodes for both the Home and Foreign networks to use the HTTPS protocol.

Require Use of Different Ethereum Nodes by Validator Bridges

The Proof of Authority consensus algorithm implemented by the POA Bridge relies on the avoidance of a single point of failure. It is critical that the various validator Bridges use a variety of different Ethereum nodes on both the Home and Foreign networks. It is especially important that no quorum (the number sufficient to approve a deposit or withdrawal) of validators ever use the same Ethereum node on either network.

Require Hardening and Segmentation of Validator Bridges

It is critical that the compromise of a single validator Bridge does not immediately lead to the compromise of other Bridges. This is especially true if the total number of Bridges is relatively low as it is likely to be shortly after launch. No two bridges should reside on the same network or in the same location. Bridges should be run on dedicated systems and should never be run alongside other client-facing applications. In cases where two or more Bridges are managed by the same individuals it should be verified that Bridges do not share the same system accounts and passwords or accept the same shared keys for authentication.