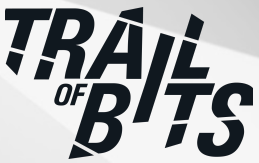

Hopper (XYZ) token contract

Security Assessment (Summary Report)

January 18, 2023

Prepared for:
Liz Rodan
Paxos

Prepared by: **Alexander Remie, Tarun Bansal, and Kurt Willis**

About Trail of Bits

Founded in 2012 and headquartered in New York, Trail of Bits provides technical security assessment and advisory services to some of the world's most targeted organizations. We combine high-end security research with a real-world attacker mentality to reduce risk and fortify code. With 100+ employees around the globe, we've helped secure critical software elements that support billions of end users, including Kubernetes and the Linux kernel.

We maintain an exhaustive list of publications at https://github.com/trailofbits/publications, with links to papers, presentations, public audit reports, and podcast appearances.

In recent years, Trail of Bits consultants have showcased cutting-edge research through presentations at CanSecWest, HCSS, Devcon, Empire Hacking, GrrCon, LangSec, NorthSec, the O'Reilly Security Conference, PyCon, REcon, Security BSides, and SummerCon.

We specialize in software testing and code review projects, supporting client organizations in the technology, defense, and finance industries, as well as government entities. Notable clients include HashiCorp, Google, Microsoft, Western Digital, and Zoom.

Trail of Bits also operates a center of excellence with regard to blockchain security. Notable projects include audits of Algorand, Bitcoin SV, Chainlink, Compound, Ethereum 2.0, MakerDAO, Matic, Uniswap, Web3, and Zcash.

To keep up to date with our latest news and announcements, please follow @trailofbits on Twitter and explore our public repositories at https://github.com/trailofbits. To engage us directly, visit our "Contact" page at https://www.trailofbits.com/contact, or email us at info@trailofbits.com.

Trail of Bits, Inc.
228 Park Ave S #80688
New York, NY 10003
https://www.trailofbits.com
info@trailofbits.com

Notices and Remarks

Copyright and Distribution

Test Coverage Disclaimer

All activities undertaken by Trail of Bits in association with this project were performed in accordance with a statement of work and agreed upon project plan.

Security assessment projects are time-boxed and often reliant on information that may be provided by a client, its affiliates, or its partners. As a result, the findings documented in this report should not be considered a comprehensive list of security issues, flaws, or defects in the target system or codebase.

Trail of Bits uses automated testing techniques to rapidly test the controls and security properties of software. These techniques augment our manual security review work, but each has its limitations: for example, a tool may not generate a random edge case that violates a property or may not fully complete its analysis during the allotted time. Their use is also limited by the time and resource constraints of a project.

Table of Contents

Executive Summary

Engagement Overview

Paxos engaged Trail of Bits to review the security of its Hopper (XYZ) token contract. From December 12 to December 16, 2022, a team of three consultants conducted a security review of the client-provided source code, with one person-week of effort. Details of the project's timeline, test targets, and coverage are provided in subsequent sections of this report.

Project Scope

Our testing efforts were focused on the identification of flaws that could result in a compromise of confidentiality, integrity, or availability of the target system. We conducted this audit with full knowledge of the target system, including access to the source code and documentation. We performed static and dynamic automated and manual testing of the target system.

Summary of Findings

The audit did not uncover any significant flaws or defects that could impact system confidentiality, integrity, or availability.

EXPOSURE ANALYSIS

Severity	*Count*
Medium	1
Low	4
Informational	4
Undetermined	1

CATEGORY BREAKDOWN

Category	*Count*
Access Controls	3
Authentication	1
Configuration	1
Data Validation	1
Timing	3
Undefined Behavior	1

Project Summary

Contact Information

The following managers were associated with this project:

Dan Guido, Account Manager
dan@trailofbits.com

Anne Marie Barry, Project Manager
annemarie.barry@trailofbits.com

The following engineers were associated with this project:

Alexander Remie, Consultant
alexander.remie@trailofbits.com

Tarun Bansal, Consultant
tarun.bansal@trailofbits.com

Kurt Willis, Consultant
kurt.willis@trailofbits.com

Project Timeline

The significant events and milestones of the project are listed below.

Date	Event
December 9, 2022	Pre-project kickoff call
December 19, 2022	Delivery of report draft; report readout meeting
January 18, 2023	Delivery of final report and fix review

Project Goals

The engagement was scoped to provide a security assessment of the Paxos Hopper (XYZ) token. Specifically, we sought to answer the following non-exhaustive list of questions:

- Does the token adhere to the ERC20 standard?
- Does the token have a mechanism for mitigating the race condition risk caused by the ERC20 `approve` function?
- Are there any reentrancy vulnerabilities in the contract?
- Are access controls applied correctly to all functions?
- Does the use of the `delegatecall` proxy pattern as an upgradeability mechanism cause any problems?
- Are all inputs validated?
- Are events emitted for all updates to important contract variables?
- Are there any front-running vulnerabilities in the contract?
- Does the reassignment of a privileged role cause any problems?
- Are there any vulnerabilities in the signature schema used in the mechanism for delegating a transfer?
- Is the contract deployment / initialization mechanism sound?

Project Targets

The engagement involved a review and testing of the following target.

Hopper (XYZ)

Repository	https://github.com/paxosglobal/xyz-contracts
Version	`503c871f87d257d43ee14c373f7e52ff3f3d0c4b`
Type	Solidity
Platform	Ethereum

Project Coverage

This section provides an overview of the analysis coverage of the review, as determined by our high-level engagement goals. Our approaches and their results include the following:

- **Slither.** We ran Slither, our static analysis tool, over the codebase and triaged the findings. No major issues were identified.
- **Echidna.** We used Echidna, our smart contract fuzzer, to check ERC20-related invariants of the token contract. All of the invariants held.
- **Manual review.** We manually reviewed all contracts in the repository specified in the Project Targets section.

Summary of Findings

The table below summarizes the findings of the review, including type and severity details.

ID	Title	Type	Severity
1	Supply controller can mint and burn tokens while the contract is paused	Access Controls	Informational
2	Calls to initialize() can be front-run	Timing	Low
3	initializeDomainSeparator() can be used to disrupt ownership transfers	Access Controls	Low
4	Solidity compiler optimizations can be problematic	Undefined Behavior	Undetermined
5	The ERC20 approve function can create a race condition that enables multiple transfers from a single wallet	Timing	Medium
6	Missing zero address checks in setAssetProtectionRole() and setBetaDelegateWhitelister()	Data Validation	Informational
7	Former supply controller's token balance is not transferred to the new supply controller	Access Controls	Low
8	Lack of chainID validation in the transfer-delegation functionality allows reuse of signatures across forks	Authentication	Informational
9	A failure to call initializeDomainSeparator() could prevent users from signing the data required for the delegation of transfers	Configuration	Informational
10	Malicious users can front-run betaDelegatedTransfer() calls to collect fees	Timing	Low

A. Vulnerability Categories

The following tables describe the vulnerability categories, severity levels, and difficulty levels used in this document.

Vulnerability Categories	
Category	**Description**
Access Controls	Insufficient authorization or assessment of rights
Auditing and Logging	Insufficient auditing of actions or logging of problems
Authentication	Improper identification of users
Configuration	Misconfigured servers, devices, or software components
Cryptography	A breach of system confidentiality or integrity
Data Exposure	Exposure of sensitive information
Data Validation	Improper reliance on the structure or values of data
Denial of Service	A system failure with an availability impact
Error Reporting	Insecure or insufficient reporting of error conditions
Patching	Use of an outdated software package or library
Session Management	Improper identification of authenticated users
Testing	Insufficient test methodology or test coverage
Timing	Race conditions or other order-of-operations flaws
Undefined Behavior	Undefined behavior triggered within the system

Severity Levels	
Severity	**Description**
Informational	The issue does not pose an immediate risk but is relevant to security best practices.
Undetermined	The extent of the risk was not determined during this engagement.
Low	The risk is small or is not one the client has indicated is important.
Medium	User information is at risk; exploitation could pose reputational, legal, or moderate financial risks.
High	The flaw could affect numerous users and have serious reputational, legal, or financial implications.

Difficulty Levels	
Difficulty	**Description**
Undetermined	The difficulty of exploitation was not determined during this engagement.
Low	The flaw is well known; public tools for its exploitation exist or can be scripted.
Medium	An attacker must write an exploit or will need in-depth knowledge of the system.
High	An attacker must have privileged access to the system, may need to know complex technical details, or must discover other weaknesses to exploit this issue.

B. Status Categories

The following table describes the statuses used to indicate whether an issue has been sufficiently addressed.

Fix Status	
Status	**Description**
Undetermined	The status of the issue was not determined during this engagement.
Unresolved	The issue persists and has not been resolved.
Partially Resolved	The issue persists but has been partially resolved.
Resolved	The issue has been sufficiently resolved.

C. Fix Review Results

On January 13, 2023, Trail of Bits reviewed the fixes and mitigations implemented by the Paxos team for issues identified in this report.

Paxos resolved five of the issues described in the report and did not resolve the other five.

We reviewed each of the fixes to ensure that the proposed remediation would be effective. For additional information, see the Detailed Fix Log.

ID	Title	Severity	Status
1	Supply controller can mint and burn tokens while the contract is paused	Informational	Resolved
2	Calls to initialize() can be front-run	Low	Resolved
3	initializeDomainSeparator() can be used to disrupt ownership transfers	Low	Resolved
4	Solidity compiler optimizations can be problematic	Undetermined	Unresolved
5	The ERC20 approve function can create a race condition that enables multiple transfers from a single wallet	Medium	Resolved
6	Missing zero address checks in setAssetProtectionRole() and setBetaDelegateWhitelister()	Informational	Unresolved
7	Former supply controller's token balance is not transferred to the new supply controller	Low	Unresolved
8	Lack of chainID validation in the transfer-delegation functionality allows reuse of signatures across forks	Informational	Unresolved

9	A failure to call initializeDomainSeparator() could prevent users from signing the data required for the delegation of transfers	Informational	Resolved
10	Malicious users can front-run betaDelegatedTransfer() calls to collect fees	Low	Unresolved

Detailed Fix Log

1: Supply controller can mint and burn tokens in a paused state (PR15)
Resolved. The README has been updated and now explains that minting and burning is still possible when the contract is in a paused state. The Paxos team provided the following additional context:

> *In the extreme case that the supply controller has adjusted the supply against the wishes of the owner, the tokens will not be transferable in the paused state and the owner will change the supply controller, thus seizing the supply controller role. Note that the supply controller is a trusted address in the system so compromise is highly unlikely. Any loss of the supply controller will result in the immediate incident response of changing the supply controller address.*

2: initialize() can be front-run (PR14)
Resolved. The deployment script has been updated and will throw an error recommending verification of the initialization process if the `initialize()` call does not succeed. Additionally, the Paxos team updated the internal deployment documentation to require verification of the initialization's execution.

3: initializeDomainSeparator() can be used to grief ownership transfers (PR16)
Resolved. The implementation of the `initializeDomainSeparator()` function has been updated and no longer resets `proposedOwner` to the zero address.

4: Solidity compiler optimizations can be problematic
Unresolved. Paxos provided the following additional context:

> *It is safer to leave the optimizations on given that we have enabled optimizations for this solidity version for all Paxos stablecoins for the last four years. Paxos will revisit whether or not to include optimizations when using a newer version of Solidity.*

5: Multiple calls to ERC20 approve function may create race condition that could lead to increased token transfer from a single wallet (PR17)
Resolved. The implementation has been updated with `increaseApproval` and `decreaseApproval` functions. Additionally, the README has been updated to encourage the use of the two new functions when appropriate.

6: Missing zero address checks in setAssetProtectionRole() and setBetaDelegateWhitelister()
Unresolved. Paxos provided the following additional context:

> *This was by design as Paxos wants to keep the ability to unset the role. Since Paxos has the ability to set the role to a non-zero address at any time, it will never be stuck as the zero address. Paxos does not plan to remove the owner's privilege of assigning these roles.*

7: Revoked supply controller's token balance not transferred to new supply controller
Unresolved. Paxos provided the following additional context:

> *Paxos prefers the flexibility of being able to move the revoked supply controller's token balance in a second transaction. There are use cases where Paxos may want to keep tokens in the revoked supply controller. If there is an exploit where the supply controller is compromised, Paxos still has the ability to transfer the tokens to the new supply controller.*

8: Lack of chainID validation allows re-using delegated transfer signatures across forks
Unresolved. Paxos provided the following additional context:

> *This is not a concern for Paxos because this token is asset backed and will only be supported on a single chain at once. Paxos does not plan to upgrade the solidity compiler version, given we feel more confident in this version, which has been widely used by our stablecoins over the past four years.*

9: Forgetting to call initializeDomainSeparator() could prevent users from signing the required data for the delegated transfer functions (PR12)
Resolved. The Paxos team updated the implementation of `initializeDomainSeparator()`, which is now called from the `initialize()` function. Additionally, the visibility of the `initializeDomainSeparator()` function has been changed to `private`, which prevents calls to the function after the contract's initialization.

10: betaDelegatedTransfer() can be front-run to collect fees
Unresolved. Paxos provided the following additional context:

> *This feature is for our clients to cover fees for token transfers and we will only whitelist major partners that we trust. Furthermore the incentive for one of these partners to use this exploit is extremely low.*