ENSO WALLET SECURITY AUDIT REPORT

February 21, 2023

MixBytes()

TABLE OF CONTENTS

1. INTRODUCTION

1.1 Disclaimer

The audit makes no statements or warranties about utility of the code, safety of the code, suitability of the business model, investment advice, endorsement of the platform or its products, regulatory regime for the business model, or any other statements about fitness of the contracts to purpose, or their bug free status. The audit documentation is for discussion purposes only. The information presented in this report is confidential and privileged. If you are reading this report, you agree to keep it confidential, not to copy, disclose or disseminate without the agreement of the Client. If you are not the intended recipient(s) of this document, please note that any disclosure, copying or dissemination of its content is strictly forbidden.

1.2 Security Assessment Methodology

A group of auditors are involved in the work on the audit. The security engineers check the provided source code independently of each other in accordance with the methodology described below:

1. Project architecture review:

- Project documentation review.
- General code review.
- Reverse research and study of the project architecture on the source code alone.

Stage goals
- Build an independent view of the project's architecture.
- Identifying logical flaws.

2. Checking the code in accordance with the vulnerabilities checklist:

- Manual code check for vulnerabilities listed on the Contractor's internal checklist. The Contractor's checklist is constantly updated based on the analysis of hacks, research, and audit of the clients' codes.
- Code check with the use of static analyzers (i.e Slither, Mythril, etc).

Stage goal

Eliminate typical vulnerabilities (e.g. reentrancy, gas limit, flash loan attacks etc.).

3. Checking the code for compliance with the desired security model:

- Detailed study of the project documentation.
- Examination of contracts tests.
- Examination of comments in code.
- Comparison of the desired model obtained during the study with the reversed view obtained during the blind audit.
- PoC development for possible exploits with the use of such programs as Brownie and Hardhat.

Stage goal

Detect inconsistencies with the desired model.

4. Consolidation of the auditors' interim reports into one:

- Cross check: each auditor reviews the reports of the others.
- Discussion of the issues found by the auditors.
- Issuance of an interim audit report.

Stage goals

- Double-check all the found issues to make sure they are relevant and the determined threat level is correct.
- Provide the Client with an interim report.

5. Bug fixing & re-audit:

- The Client either fixes the issues or provides comments on the issues found by the auditors. Feedback from the Customer must be received on every issue/bug so that the Contractor can assign them a status (either "fixed" or "acknowledged").
- Upon completion of the bug fixing, the auditors double-check each fix and assign it a specific status, providing a proof link to the fix.
- A re-audited report is issued.

Stage goals

- Verify the fixed code version with all the recommendations and its statuses.
- Provide the Client with a re-audited report.

6. Final code verification and issuance of a public audit report:

- The Customer deploys the re-audited source code on the mainnet.
- The Contractor verifies the deployed code with the re-audited version and checks them for compliance.
- If the versions of the code match, the Contractor issues a public audit report.

Stage goals

- Conduct the final check of the code deployed on the mainnet.
- Provide the Customer with a public audit report.

Finding Severity breakdown

All vulnerabilities discovered during the audit are classified based on their potential severity and have the following classification:

Severity	Description
Critical	Bugs leading to assets theft, fund access locking, or any other loss of funds.
High	Bugs that can trigger a contract failure. Further recovery is possible only by manual modification of the contract state or replacement.
Medium	Bugs that can break the intended contract logic or expose it to DoS attacks, but do not cause direct loss funds.
Low	Bugs that do not have a significant immediate impact and could be easily fixed.

Based on the feedback received from the Customer regarding the list of findings discovered by the Contractor, they are assigned the following statuses:

Status	Description
Fixed	Recommended fixes have been made to the project code and no longer affect its security.
Acknowledged	The Customer is aware of the finding. Recommendations for the finding are planned to be resolved in the future.

1.3 Project Overview

Enso Wallet is a smart wallet supporting native tokens, ERC20, ERC721, ERC1151, and signing messages by ERC1271 standart and arbitrary custom transaction.

Smart wallet shares one common implementation for all smart wallets. Every account may have several wallets with different so-called "wallet labels."

`EnsoBeacon` is a contract to govern the wallet implementation and the wallet factory implementation.

Centralization warning: Until renounced, the project administration can replace the initial wallet implementation with an insecure/unaudited one.

`EnsoWallet` is a smart wallet implementation to allow arbitrary calls to external contracts.

`EnsoWalletFactory` is a factory contract to deploy smart wallet instances.

1.4 Project Dashboard

Project Summary

Title	Description
Client	Enso Finance
Project name	Enso Wallet
Timeline	21 Nov 2022 - 20 Feb 2023

Title	Description
Number of Auditors	3

Project Log

Date	Commit Hash	Note
21.11.2022	4902e55608f975f73772310955444110b1cfc4fc	Initial commit for Enso Wallet
21.11.2022	7770653e40bec8206337bd4e08b5e3e7ef72c0c3	Previous audit of Enso Weiroll by MixBytes
21.11.2022	0d658b5a6432d849c92c1ef3bcb9710b0004292e	Initial commit (enso-weiroll)
12.12.2022	9f65d93f409034a95f3cbac8cf4cbf7108fd29b4	Code with fixes for reaudit (enso-wallet)
12.12.2022	ff226659bb3e04ebbf43e1043898180d424c9d63	Code with fixes for reaudit (enso-weiroll)
13.12.2022	56836bf1127df19af81aca58b7220199c9288907	Code with the final fixes (enso-wallet)
26.01.2023	951a4a247165f4209bceb6deb628a4970bf3f6da	Code with additional fixes and improvements (enso-wallet)
26.01.2023	900250114203727ff236d3f6313673c17c2d90dd	Code with additional fixes and improvements (enso-weiroll)

Project Scope

The audit covered the following files:

File name	Link
CommandBuilder.sol	CommandBuilder.sol

File name	Link
VM.sol	VM.sol
EnsoBeacon.sol	EnsoBeacon.sol
EnsoWalletFactory.sol	EnsoWalletFactory.sol
EnsoWallet.sol	EnsoWallet.sol
AccessController.sol	AccessController.sol
ACL.sol	ACL.sol
Ownable.sol	Ownable.sol
Roles.sol	Roles.sol
EnsoShortcutsHelpers.sol	EnsoShortcutsHelpers.sol
MathHelpers.sol	MathHelpers.sol
SignedMathHelpers.sol	SignedMathHelpers.sol
TupleHelpers.sol	TupleHelpers.sol
BeaconClones.sol	BeaconClones.sol
StorageAPI.sol	StorageAPI.sol
UpgradeableProxy.sol	UpgradeableProxy.sol
ERC1271.sol	ERC1271.sol
MinimalWallet.sol	MinimalWallet.sol
Timelock.sol	Timelock.sol

Deployments

1. EnsoBeacon

Address: 0x277D98D33b7F44921d4230697DeF8d1D56aBAa62

Verified file	Source file	Comments
File 1 of 5: EnsoBeacon.sol	EnsoBeacon.sol	audited
File 2 of 5: Timelock.sol	Timelock.sol	audited
File 3 of 5: IBeacon.sol	IBeacon.sol	reviewed
File 4 of 5: IOwnable.sol	IOwnable.sol	reviewed
File 5 of 5: IUUPS.sol	IUUPS.sol	reviewed

2. EnsoWallet

Address: 0xb6Bc9B50b4AC1397AB03d8a24d8fa529a5070ff0

Verified file	Source file	Comments
File 1 of 29: CommandBuilder.sol	CommandBuilder.sol	audited
File 2 of 29: VM.sol	VM.sol	audited
File 3 of 29: IERC1155.sol	IERC1155.sol	trusted as part of OpenZeppelin
File 4 of 29: IERC1155Receiver.sol	IERC1155Receiver.sol	trusted as part of OpenZeppelin
File 5 of 29: ERC1155Holder.sol	ERC1155Holder.sol	trusted as part of OpenZeppelin
File 6 of 29: ERC1155Receiver.sol	ERC1155Receiver.sol	trusted as part of OpenZeppelin
File 7 of 29: IERC20.sol	IERC20.sol	trusted as part of OpenZeppelin
File 8 of 29: draft-IERC20Permit.sol	draft-IERC20Permit.sol	trusted as part of OpenZeppelin

Verified file	Source file	Comments
File 9 of 29: SafeERC20.sol	SafeERC20.sol	trusted as part of OpenZeppelin*
File 10 of 29: IERC721.sol	IERC721.sol	trusted as part of OpenZeppelin*
File 11 of 29: IERC721Receiver.sol	IERC721Receiver.sol	trusted as part of OpenZeppelin
File 12 of 29: ERC721Holder.sol	ERC721Holder.sol	trusted as part of OpenZeppelin
File 13 of 29: Address.sol	Address.sol	trusted as part of OpenZeppelin*
File 14 of 29: Strings.sol	Strings.sol	trusted as part of OpenZeppelin*
File 15 of 29: ECDSA.sol	ECDSA.sol	trusted as part of OpenZeppelin*
File 16 of 29: ERC165.sol	ERC165.sol	trusted as part of OpenZeppelin
File 17 of 29: IERC165.sol	IERC165.sol	trusted as part of OpenZeppelin
File 18 of 29: Math.sol	Math.sol	trusted as part of OpenZeppelin*
File 19 of 29: EnsoWallet.sol	EnsoWallet.sol	audited
File 20 of 29: ACL.sol	ACL.sol	audited
File 21 of 29: AccessController.sol	AccessController.sol	audited
File 22 of 29: Roles.sol	Roles.sol	audited
File 23 of 29: IERC1271.sol	IERC1271.sol	reviewed
File 24 of 29: IEnsoWallet.sol	IEnsoWallet.sol	reviewed
File 25 of 29: IModuleManager.sol	IModuleManager.sol	reviewed
File 26 of 29: StorageAPI.sol	StorageAPI.sol	audited
File 27 of 29: ERC1271.sol	ERC1271.sol	audited
File 28 of 29: MinimalWallet.sol	MinimalWallet.sol	audited

Verified file	Source file	Comments
File 29 of 29: ModuleManager.sol	ModuleManager.sol	audited

Although OpenZeppelin version 4.7.3 is specified at the package.json file, some files marked with an asterisk are probably imported from version 4.8.0.

3. EnsoWalletFactory

Address: 0x66fc62c1748E45435b06cF8dD105B73E9855F93E

Verified file	Source file	Comments
File 1 of 11: draft-IERC1822.sol	draft-IERC1822.sol	trusted as part of OpenZeppelin
File 2 of 11: ERC1967Upgrade.sol	ERC1967Upgrade.sol	trusted as part of OpenZeppelin
File 3 of 11: IBeacon.sol	IBeacon.sol	trusted as part of OpenZeppelin
File 4 of 11: UUPSUpgradeable.sol	UUPSUpgradeable.sol	trusted as part of OpenZeppelin*
File 5 of 11: Address.sol	Address.sol	trusted as part of OpenZeppelin*
File 6 of 11: StorageSlot.sol	StorageSlot.sol	trusted as part of OpenZeppelin
File 7 of 11: EnsoWalletFactory.sol	EnsoWalletFactory.sol	audited
File 8 of 11: Ownable.sol	Ownable.sol	audited
File 9 of 11: IEnsoWallet.sol	IEnsoWallet.sol	reviewed
File 10 of 11: BeaconClones.sol	BeaconClones.sol	audited
File 11 of 11: StorageAPI.sol	StorageAPI.sol	audited

Although OpenZeppelin version 4.7.3 is specified at the package.json file, some files marked with an asterisk are probably imported from version 4.8.0.

1.5 Summary of findings

Severity	# of Findings
Critical	2
High	0
Medium	3
Low	8

ID	Name	Severity	Status
C-1	Destruction of the EnsoWallet implementation contract	Critical	Fixed
C-2	Front-run attack on the deployment of EnsoWalletFactory	Critical	Fixed
M-1	EXECUTOR has a full write access to the wallet storage	Medium	Fixed
M-2	Admin can bypass the upgrade delay by `setDelay`	Medium	Fixed
M-3	Admin can bypass the upgrade delay by `delegate` and `emergencyUpgrade`	Medium	Fixed
L-1	Conflicting flow of pending `upgradeFactory` and `setFactory`	Low	Fixed
L-2	Insufficient event emitting	Low	Fixed
L-3	Potential hash collisions for constants	Low	Fixed
L-4	Null checks	Low	Fixed
L-5	Using `memory` instead of `calldata`	Low	Fixed

L-6	Spelling mistakes	Low	Fixed
L-7	Unchecked timelock delay	Low	Fixed
L-8	Passing the `return data` by the EVM state	Low	Acknowledged

2. FINDINGS REPORT

2.1 Critical

C-1	Destruction of the EnsoWallet implementation contract
Severity	Critical
Status	Fixed in a5a4045a

Description

An attacker can make a direct call (not via proxy) to EnsoWallet.sol#L24 and execute the SELFDESTRUCT opcode or specify themself as EXECUTOR and gain the ability to execute SELFDESTRUCT later. Consequently, the current implementation contract will be destroyed, and all users' wallet functionality will be inaccessible until the core upgrade. The worst case occurs if an attack happens after EnsoBeacon.renounceAdministration(), and all users' funds will be frozen.

Recommendation

We recommend disallowing direct calls to EnsoWallet.initialize().

C-2	Front-run attack on the deployment of EnsoWalletFactory
Severity	Critical
Status	Fixed in f6c0a5a4

Description

An attacker can place their transactions between the deployment of the EnsoWalletFactory implementation and EnsoWalletFactory.sol#L26 to specify themself as the contract owner and make an upgrade of the EnsoWalletFactory to the modified one. The modified factory contract may implement backdoor functionality to gain control of the deployed user wallets.

Recommendation

We recommend improving the code of the upgradeable proxy to disallow the gain of ownership by arbitrary accounts or at least improve the deployment process in order to implement deployment and initialization in a single transaction.

2.2 High

Not Found

2.3 Medium

M-1	EXECUTOR has a full write access to the wallet storage
Severity	Medium
Status	Fixed in ff226659

Description

EXECUTOR has a write access to any storage slots vie the executeShortcut function and delegatecall to a specially crafted library. This allows to trigger
transfer/renounce of the OWNER address and other unintended actions.
EnsoWallet.sol#L57

Recommendation

We recommend implementing an access control for the DELEGATECALL, i.e. a whitelist of permitted libraries.

M-2	Admin can bypass the upgrade delay by `setDelay`
Severity	Medium
Status	Fixed in 1ece0b19

Description

The contract implements a special flow to upgrade the wallet core with delay, but the admin can force an immediate upgrade by reducing the delay using setDelay().
EnsoBeacon.sol#L208

This will deprive users of the ability to check implementations before applying.

Recommendation

We recommend improving the upgrade delay flow.

M-3	Admin can bypass the upgrade delay by `delegate` and `emergencyUpgrade`
Severity	Medium
Status	Fixed in 1ece0b19

Description

The admin can bypass the upgrade delay flow and immediately change the address of the wallet core by following the sequence:

- `transferDelegation` (to a new `delegate` account, controlled by the admin)
- `acceptDelegation` (from the `delegate` account)
- `upgradeFallback` (from the admin account)
- `emegrencyUpgrade` (from the `delegate` account)

Recommendation

We recommend improving the upgrade delay flow.

2.4 Low

L-1	Conflicting flow of pending `upgradeFactory` and `setFactory`
Severity	Low
Status	Fixed in 60780568

Description

Calling EnsoBeacon.sol#L74 after `setFactory` can cause unexpected behavior.

Recommendation

We recommend improving the upgrade delayflow.

L-2	Insufficient event emitting
Severity	Low
Status	Fixed in 70e778d2

Description

In some edge cases it may be not so easy to obtain the wallet owner address and address of the factory that created it.

Recommendation

We recommend adding the wallet owner and the factory address to the EnsoWalletFactory.sol#L90.

L-3	Potential hash collisions for constants
Severity	Low
Status	Fixed in e348cc18

Description

Roles.sol#L5
are vulnerable to potential hash collisions.

Recommendation

Although, we currently can't find any attack vector for this issue, we recommend improving the security of the constants by decreasing it by `-1` just like it is Ownable.sol#L11.

L-4	Null checks
Severity	Low
Status	Fixed in 2280305c

Description

Some parameters have no null checks:

- EnsoBeacon.sol#L201
- EnsoBeacon.sol#L123
- AccessController.sol#L21

Recommendation

We recommend adding a null check for `newFactory`, `newImplementation` and `account`.

L-5	Using `memory` instead of `calldata`
Severity	Low
Status	Fixed in 5145c45a

Description

Using `memory` instead of `calldata` for input arrays in external functions:
MinimalWallet.sol#L45
MinimalWallet.sol#L84
MinimalWallet.sol#L101
MinimalWallet.sol#L112
MinimalWallet.sol#L121
MinimalWallet.sol#L145
MinimalWallet.sol#L155
MinimalWallet.sol#L165

Recommendation

We recommend replacing `memory` by `calldata`.

L-6	Spelling mistakes
Severity	Low
Status	Fixed in 814f0e5f

Description

Some texts have spelling mistakes:

1. `balanceAdderess` -> `balanceAddress`
   EnsoShortcutsHelpers.sol#L14
2. `renounes` -> `renouns`
   EnsoBeacon.sol#L147
3. `alway` -> `always`
   EnsoBeacon.sol#L179
4. `indetify` -> `identify`
   EnsoWalletFactory.sol#L43
   EnsoWalletFactory.sol#L79
5. `implemenation` -> `implementation`
   EnsoBeacon.sol#L67

Recommendation

We recommend correcting the mistakes.

L-7	Unchecked timelock delay
Severity	Low
Status	Fixed in 1ece0b19

Description

The administrator can inadvertently (i.e. using millisecond timestamp notation) setup a timelock delay that is too long. This may cause an unacceptably long delay (i.e. a two-year delay instead of a seven-day delay).

Recommendation

We recommend disallowing unintended long delays by limiting the maximum delay of the timelock with some reasonable value.

L-8	Passing the `return data` by the EVM state
Severity	Low
Status	Acknowledged

Description

ModuleManager.sol#L45 is using `returndatasize` and `returndatacopy` in the assumption that virtual function `_executeCall` is preserving the result of the corresponding external call at the EVM state. As for the audited commit, this assumption is correct. However, the implementation of the virtual `_executeCall` function may become more complex during a past codebase development, and this assumption may break the contract logic later.

Recommendation

It is recommended to avoid using the EVM state as a method of passing values outside of a single solidity function, especially if the values are passed between functions located in different source files.

2.5 Appendix

1 Monitoring Recommendation

The project contains smart contracts that require monitoring. For these purposes, it is recommended to proceed with developing new monitoring events based on Forta (https://forta.org) with which you can track the following exemplary incidents:

- core upgrades and pending core and factory contract upgrades
- administration transfers

3. ABOUT MIXBYTES

MixBytes is a team of blockchain developers, auditors and analysts keen on decentralized systems. We build opensource solutions, smart contracts and blockchain protocols, perform security audits, work on benchmarking and software testing solutions, do research and tech consultancy.

Contacts


https://github.com/mixbytes/audits_public


https://mixbytes.io/


hello@mixbytes.io


https://twitter.com/mixbytes