PALADIN
BLOCKCHAIN SECURITY

Smart Contract Security Assessment

Final Report

For Ether.Fi (Migration)

30 September 2024

paladinsec.co

info@paladinsec.co

Table of Contents

Disclaimer

Paladin Blockchain Security ("Paladin") has conducted an independent audit to verify the integrity of and highlight any vulnerabilities or errors, intentional or unintentional, that may be present in the codes that were provided for the scope of this audit. This audit report does not constitute agreement, acceptance or advocation for the Project that was audited, and users relying on this audit report should not consider this as having any merit for financial advice in any shape, form or nature. The contracts audited do not account for any economic developments that may be pursued by the Project in question, and that the veracity of the findings thus presented in this report relate solely to the proficiency, competence, aptitude and discretion of our independent auditors, who make no guarantees nor assurance that the contracts are completely free of exploits, bugs, vulnerabilities or deprecation of technologies. Further, this audit report shall not be disclosed nor transmitted to any persons or parties on any objective, goal or justification without due written assent, acquiescence or approval by Paladin.

All information provided in this report does not constitute financial or investment advice, nor should it be used to signal that any persons reading this report should invest their funds without sufficient individual due diligence regardless of the findings presented in this report. Information is provided 'as is', and Paladin is under no covenant to the completeness, accuracy or solidity of the contracts audited. In no event will Paladin or its partners, employees, agents or parties related to the provision of this audit report be liable to any parties for, or lack thereof, decisions and/or actions with regards to the information provided in this audit report.

Cryptocurrencies and any technologies by extension directly or indirectly related to cryptocurrencies are highly volatile and speculative by nature. All reasonable due diligence and safeguards may yet be insufficient, and users should exercise considerable caution when participating in any shape or form in this nascent industry.

The audit report has made all reasonable attempts to provide clear and articulate recommendations to the Project team with respect to the rectification, amendment and/or revision of any highlighted issues, vulnerabilities or exploits within the contracts provided. It is the sole responsibility of the Project team to sufficiently test and perform checks, ensuring that the contracts are functioning as intended, specifically that the functions therein contained within said contracts have the desired intended effects, functionalities and outcomes of the Project team.

Paladin retains the right to re-use any and all knowledge and expertise gained during the audit process, including, but not limited to, vulnerabilities, bugs, or new attack vectors. Paladin is therefore allowed and expected to use this knowledge in subsequent audits and to inform any third party, who may or may not be our past or current clients, whose projects have similar vulnerabilities. Paladin is furthermore allowed to claim bug bounties from third-parties while doing so.

1 Overview

This report has been prepared for Ether.fi's migration contracts on the Ethereum network. Paladin provides a user-centred examination of the smart contracts to look for vulnerabilities, logic errors or other issues from both an internal and external perspective.

1.1 Summary

Project Name	ether.fi
URL	https://ether.fi
Platform	Ethereum
Language	Solidity
Preliminary Contracts	https://github.com/etherfi-protocol/weETH-cross-chain/pull/5/commits/e7180b7c8693d9c99eefe9c8f3b883dbd8f0b898
Resolution 1	https://github.com/etherfi-protocol/weETH-cross-chain/pull/11

1.2 Contracts Assessed

Name	Contract	Live Code Match
EtherFiOFTAdapter		
EtherFiOFTAdapterUpgradeable		
MigrationOFT		
DeployUpgradeableOFTAdapter		
DeployMigrationOFT		
GenerationMigrationTransactions		

1.3 Findings Summary

Severity	Found	Resolved	Partially Resolved	Acknowledged (no change made)
Governance	2	2	-	-
High	0	-	-	-
Medium	2	1	-	1
Low	4	2	-	2
Informational	1	1	-	-
Total	**9**	**6**	-	**3**

Classification of Issues

Severity	Description
Governance	Issues under this category are where the governance or owners of the protocol have certain privileges that users need to be aware of, some of which can result in the loss of user funds if the governance's private keys are lost or if they turn malicious, for example.
High	Exploits, vulnerabilities or errors that will certainly or probabilistically lead towards loss of funds, control, or impairment of the contract and its functions. Issues under this classification are recommended to be fixed with utmost urgency.
Medium	Bugs or issues that may be subject to exploit, though their impact is somewhat limited. Issues under this classification are recommended to be fixed as soon as possible.
Low	Effects are minimal in isolation and do not pose a significant danger to the project or its users. Issues under this classification are recommended to be fixed nonetheless.
Informational	Consistency, syntax or style best practices. Generally pose a negligible level of risk, if any.

1.3.1 EtherFiOFTAdapter

No issues found.

1.3.2 EtherFiOFTAdapterUpgradeable

ID	Severity	Summary	Status
01	GOV	The contract is upgradeable	✓ RESOLVED
02	LOW	The adapter contract might change	ACKNOWLEDGED

1.3.3 MigrationOFT

ID	Severity	Summary	Status
03	GOV	The contract can drain the entire OFT Adapter	✓ RESOLVED
04	INFO	Gas optimizations	✓ RESOLVED

1.3.4 DeployUpgradeableOFTAdapter

No issues found.

1.3.5 DeployMigrationOFT

ID	Severity	Summary	Status
05	LOW	`migrationOFT` should never receive any messages as its only purpose is to migrate funds - users may be able to bridge their `weEth` to `migrationOFT`	✓ RESOLVED
06	LOW	The delegate is not transferred to Etherfi's Gnosis multi-signature contract	✓ RESOLVED

1.3.6 GenerationMigrationTransactions

ID	Severity	Summary	Status
07	MEDIUM	The invariant used is imprecise	ACKNOWLEDGED
08	MEDIUM	Users can bridge `weEth` to `migrationOFT` as soon as step 1 is executed, which could lock their `weEth` on Arbitrum	✓ RESOLVED
09	LOW	The block confirmation might be insufficient	ACKNOWLEDGED

2 Findings

2.1 EtherFiOFTAdapter

`EtherFiOFTAdapter` is a basic OFT Adapter that allows the `weEth` token to be bridged via Layer Zero.

2.1.1 Privileged Functions

- `setMsgInspector`
- `setPeer`
- `setDelegate`
- `setPreCrime`
- `setEnforcedOptions`

2.1.2 Issues & Recommendations

No issues found.

2.2 EtherFiOFTAdapterUpgradeable

`EtherFiOFTAdapterUpgradeable` is a basic OFT Upgradeable Adapter that allows the `weEth` token to be bridged via LayerZero. This contract is an upgradeable proxy which allows the EtherFi team to update the implementation.

2.2.1 Privileged Functions

- `setMsgInspector`
- `setPeer`
- `setDelegate`
- `setPreCrime`
- `setEnforcedOptions`

2.2.2 Issues & Recommendations

Issue #01	The contract is upgradeable
Severity	GOVERNANCE
Description	The contract is upgradeable, which allows the owner of the proxy to upgrade the implementation to a new one.
Recommendation	Consider incorporating a Gnosis Multisignature contract as owner and ensuring that the Gnosis participants are trusted entities.
Resolution	RESOLVED The contract will be owned by a multi-signature wallet controlled by the ether.fi team

Issue #02	The adapter contract might change
Severity	LOW SEVERITY
Description	The migration process from the previous OFT Adapter to the new upgradeable one would not create major issues as the actual token addresses on both the L2s and L1 will not change. However, the adapter address would change, which could create issues for partners that use the adapter to bridge the token on the different chains. It would be necessary to check for partners that do so and make sure that they have sufficient time to update their codebase to use the new adapter.
Recommendation	Consider reaching out to partners that bridge `weEth` via LayerZero to give them sufficient time to update the adapter to the new one, as it could require redeployment if this address is hard coded in one of their contracts.
Resolution	ACKNOWLEDGED Etherfi will make sure to reach out to partners before the actual migration.

2.3 MigrationOFT

`MigrationOFT` is a basic OFT that allows the owner to initiate the migration from the previous `EtherFiOFTAdapter` to the new `EtherFiOFTAdapterUpgradeable`.

This contract will send a message to the Ethereum contract to send `weEth` out of the previous adapter to the new one. To prevent bridging issues and undesired edge cases, the entire bridging process of `weEth` will be paused during the migration.

2.3.1 Privileged Functions

- `setMsgInspector`
- `setPeer`
- `setDelegate`
- `setPreCrime`
- `setEnforcedOptions`
- `sendMigrationMessage`

2.3.2 Issues & Recommendations

Issue #03	The contract can drain the entire OFT Adapter
Severity	GOVERNANCE
Description	The contract allows the owner to drain the entire OFT Adapter. If the owner gets compromised, it could be used to steal the entire weEth locked in the OFT adapter contract. However, the funds will always be sent to the TARGET_OFT_ADAPTER, it is therefore very important to set this value carefully to the right address.
Recommendation	Consider incorporating a Gnosis Multisignature contract as owner and ensuring that the Gnosis participants are trusted entities.
Resolution	RESOLVED The contract will be owned by a multi-signature wallet controlled by the ether.fi team

Issue #04	Gas optimizations
Severity	INFORMATIONAL
Description	L57 `this.send{value: fee.nativeFee }(param, fee, this.owner());` As this function is guarded by an `onlyOwner` modifier, `this.owner()` can be simplified to `_msgSender()` to reduce gas cost.
Recommendation	Consider implementing the gas optimizations mentioned above.
Resolution	RESOLVED

2.4 DeployUpgradeableOFTAdapter

`DeployUpgradeableOFTAdapter` is a helper contract which is a script that deploys the `UpgradeableOFTAdapter` and initializes it to the correct values.

It will first deploy the implementation and then the proxy associated with this implementation. Then it will set the peer address of this contract to all the peer addresses of all the L2s where Etherfi is deployed. After that, it will set the mainnet DVNs to those of Nethermind's and LayerZero's and the enforced options to all the L2s which is to have at least 1M gas when bridging the `weEth` token (with or without calling a contract on the destination). Finally, it will transfer the ownership and the delegate to EtherFi's multi-signature wallet.

2.4.1 Issues & Recommendations

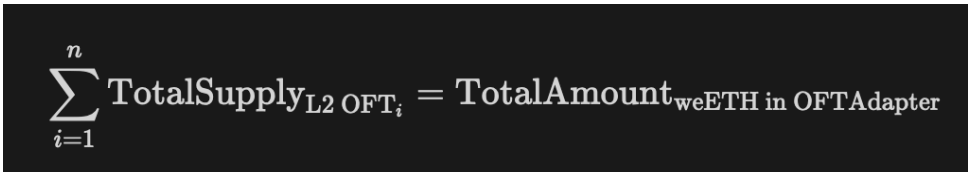
No issues found.

2.5 DeployMigrationOFT

`DeployMigrationOFT` is a helper contract which is a script that deploys `MigrationOFT` on Arbitrum and initializes it to the right values.

It will first deploy the implementation using the new OFT Adapter as the `TARGET_OFT_ADAPTER`. Then it will set the peer address of this contract to the current OFT Adapter on mainnet. After that, it will set the Arbitrum DVNs to those of Nethermind's and LayerZero's and the enforced options to mainnet which is to have at least 1M gas when bridging it (with or without calling a contract on the destination). Finally, it will transfer the ownership of the contract.

2.5.1 Issues & Recommendations

Issue #05	**`migrationOFT` should never receive any messages as its only purpose is to migrate funds – users may be able to bridge their `weEth` to `migrationOFT`**
Severity	LOW SEVERITY
Description	The script sets both the `SEND` and `RECEIVE` lib of the migration OFT, which allows this contract to send and also to receive messages. This in itself should not create any major issues, however, the Etherfi team would like to use the invariant that the sum of the L2 supply should match the balance of the OFT Adapter on mainnet.  $$\sum_{i=1}^{n} \text{TotalSupply}_{\text{L2 OFT}_i} = \text{TotalAmount}_{\text{weETH in OFTAdapter}}$$ However, `weEth` is not an OFT on Arbitrum and the invariant would not count any tokens bridged on Arbitrum using Layer Zero, but as the Receive library is set to a valid `RECEIVE` library, it is possible for any user to bridge their `weEth` from mainnet to `migrationOFT` on Arbitrum if it is added as a peer with a valid `SEND` library. This issue is only raised as low because if the `SEND` library is not set on mainnet, no messages can be sent to `migrationOFT`, however it is better to disable the receiving of messages entirely as a safety measure.
Recommendation	Consider setting the `RECEIVE` configuration to `DEAD_DVN` to prevent tokens from being bridged from mainnet to `migrationOFT` on Arbitrum.
Resolution	RESOLVED

Issue #06	The delegate is not transferred to Etherfi's Gnosis multi-signature contract
Severity	LOW SEVERITY
Description	The script only transfers the ownership to the contract, but not the delegate. If the deployer's key ever gets compromised, this could be used to change the peer address or the DVN address to prevent Etherfi from using `sendMigrationMessage`. This issue is only raised as low as the Etherfi team can always redeploy the contract and redo the entire process. However, it would be cleaner to simply update the delegate address to the Gnosis multi-signature address.
Recommendation	Consider updating the delegate address to EtherFi's Gnosis multi-signature address.
Resolution	RESOLVED The delegate is now set to EtherFi's Gnosis multi-signature address.

2.6 GenerationMigrationTransactions

`GenerationMigrationTransactions` is a helper contract which is a script that deploys the `MigrationOFT` on Arbitrum and initializes it to the correct values.

It will first deploy the implementation using the new OFT adapter as the `TARGET_OFT_ADAPTER`. Then it will set the peer address of this contract to the current OFT adapter on mainnet. After that, it will set the Arbitrum DVNs to those of Nethermind's and LayerZero's and the enforced options to mainnet which is to have at least 1M gas when bridging it (with or without calling a contract on the destination). Finally, it will transfer the ownership.

2.6.1 Issues & Recommendations

Issue #07	The invariant used is imprecise
Severity	MEDIUM SEVERITY
Description	In order to know when all inflight messages have been received, EtherFi uses this invariant: $$\sum_{i=1}^{n} \text{TotalSupply}_{\text{L2 OFT}_i} = \text{TotalAmount}_{\text{weETH in OFTAdapter}}$$ The invariant is to check that the sum of the total supply on each L2 OFT matches the total amount on the OFT adapter on mainnet. However, the OFT adapter does not store the actual number of tokens bridged, but only the total balance of the contract. If any user sends tokens directly to this contract, even 1 WEI, the invariant would never be true and the EtherFi would not be able to start the migration process. The correct amount of tokens needs to be calculated using the events emitted by the OFT Adapter: - Each time a user bridges `weEth` from mainnet to a L2, an `OFTSent` event that contains `amountSentLD` is emitted - Each time a user bridges `weEth` back to mainnet, an `OFTReceived` event that contains `amountReceivedLD` is emitted - By summing all the `amountSent` and subtracting the `amountReceivedLD`, Etherfi would be able to know the actual `totalAmount` that needs to be matched in order to validate the invariant Note that if a message is inflight to bridge back tokens to mainnet, this amount would need to be added to the `totalAmount`. This value may be harder to calculate if EtherFi ever takes a fee when someone bridges `weEth`. It does not seem to be the case here as EtherFi uses the basic OFT on all the L2s.
Recommendation	Consider using the events to calculate the `totalAmount` on the OFT Adapter instead of the balance of the contract.

Resolution

ACKNOWLEDGED

Etherfi will wait twice the maximum time a bridge transaction takes to ensure no tokens are inflight. The proposed invariant does not suffice as EtherFi may have some temporarily unbacked tokens on L2s.

Issue #08	**Users can bridge weEth to `migrationOFT` as soon as step 1 is executed, which could lock their weEth on Arbitrum**
Severity	MEDIUM SEVERITY
Description	During step 1 after the peer has been set, only the `SEND` library is set. However, when no `RECEIVE` library has been specified, Layer Zero takes the default values and Arbitrum has a default value for the `RECEIVE` library. This allows users to bridge their `weEth` tokens to Arbitrum on the Migration OFT contract, which is not ideal and will create issues when validating the invariant to check that all inflight messages have been received. As explained earlier, as the `RECEIVE` library is also set on Arbitrum during the MigrationOFT deployment, tokens can actually be sent to Arbitrum. However, if after checking that no messages are pending and the team assumes that it is safe to migrate the entire balance to the new Adapter, then the `weEth` on Arbitrum would be stuck forever unless: - the migration OFT is added as a peer of the new upgradeable OFT adapter; or - the team actually sends the locked amount on Arbitrum to the previous OFT adapter, allowing the user to bridge back to mainnet. Both cases are undesired, and could be avoided by simply preventing any tokens bridged to Arbitrum by setting the `RECEIVE` library to the `DEAD_DVN` address.
Recommendation	Consider setting the `RECEIVE` library of MigrationOFT to `DEAD_DVN` during step 1 to prevent anyone from bridging from mainnet to Arbitrum.
Resolution	RESOLVED

Issue #09	The block confirmation might be insufficient
Severity	LOW SEVERITY
Description	With the current configuration, the DVNs must always wait for at least 64 blocks before being able to confirm a message. It is important that the same configurations are used between A to B and B to A (see https://docs.layerzero.network/v2/developers/evm/protocol-gas-settings/default-config#debugging-configurations) This block confirmation is necessary to prevent issues related to chain reorgs. On mainnet, the finality is achieved after 64 blocks, but on other chains it might be greater. In the event of a reorg on a chain that reaches finality after the currently used block confirmation, Etherfi would be at risk if it happens after more than 64 blocks.
Recommendation	Consider setting block confirmation according to the chain, and not use the same block confirmation for all chains.
Resolution	ACKNOWLEDGED


PALADIN
BLOCKCHAIN SECURITY