BootNode xERC20

September 2024

Made with ❤ by the following Creed authors: Valentin Quelquejay

Table of Contents

Executive Summary

This report presents the results of our engagement with BootNode to review the xERC20 smart contracts.

The review was conducted over one week, from September 2, 2024 to September 6, 2024 by Valentin Quelquejay. A total of 5 person-days were spent.

The review focused exclusively on the differential changes between the BootNode xERC20 implementation and the reference xERC20 implementation previously reviewed by Creed.

Overall, one major finding related to the design of the system was noted, and a few minor and informational findings were filed to enforce adherence to security best practices. The changes from the reference xERC20 implementation, which has been scrutinized by Creed in the past, are minimal.

The Bootnode team addressed all the issues in parallel to the engagement. The fixes were reviewed by Creed as part of the engagement.

Scope and Objectives

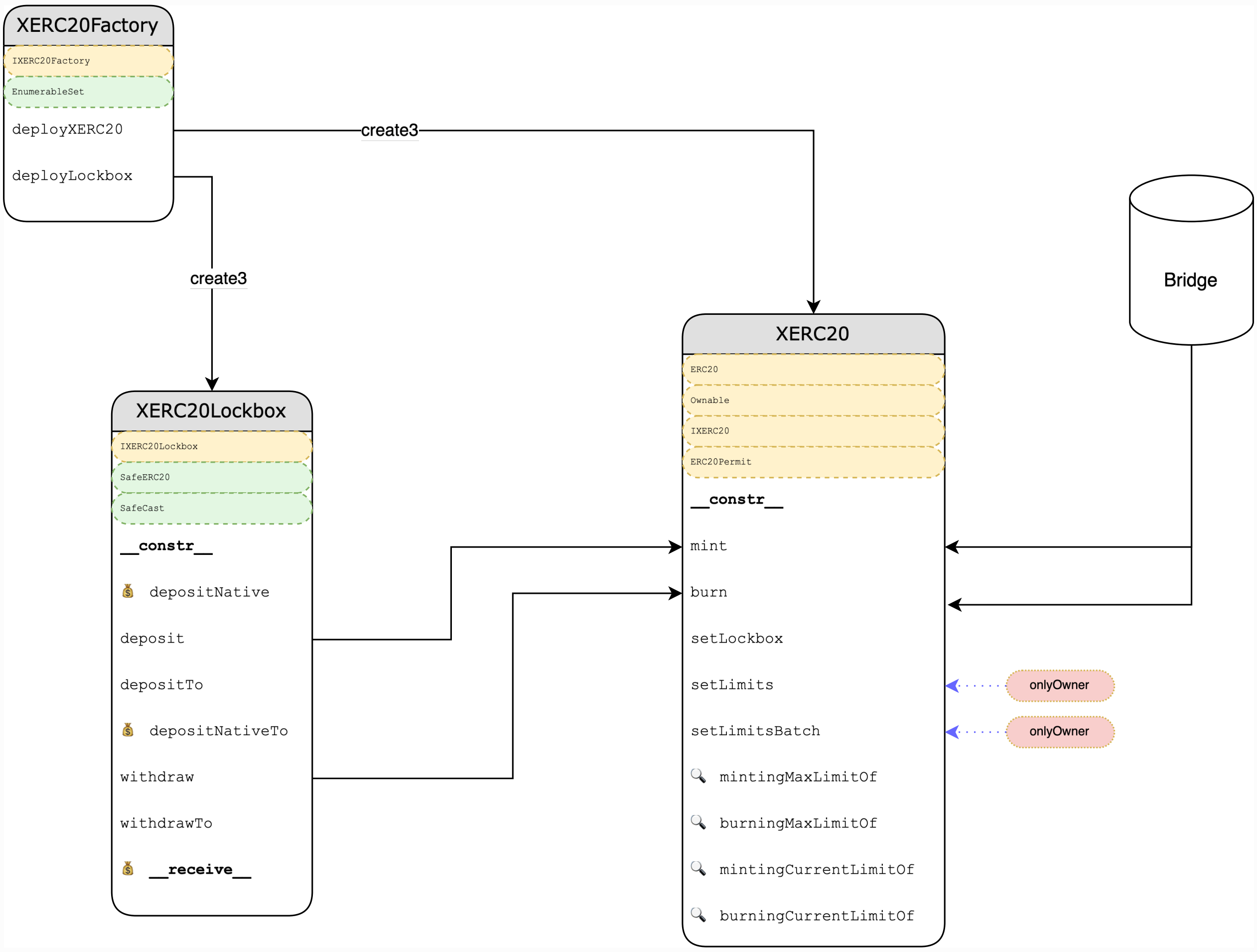
Our review focused on the commit hash 79a21e7df49c7ad93bb537410cbc8ad2280eb3f0 and more precisely on the changes introduced with regards to the reference xERC20 implementation.

Together with the BootNode team, we identified the following priorities for our review:

- Ensure the modifications to the xERC20 reference implementation do not introduce vulnerabilities.
- Ensure that the system is implemented consistently with the intended functionality, and without unintended edge cases.
- Identify known vulnerabilities particular to smart contract systems, as outlined in our Smart Contract Security Field Guide, and the ones outlined in the EEA EthTrust Security Levels Specification.

Audit Artifacts

Below is an overview of the system's smart contract architecture.



Findings

Major Decimals Handling System Design

Fixed

> Fixed via commit ef18c643df80271a40f628a4ffefffdf42ab6495. Note that the system only supports native tokens with 18 decimals.

The reference `xERC20` token contract uses 18 decimals by default. However, ERC-20 tokens may have a different number of decimals. With the reference implementation, deploying an xERC20 with a Lockbox for tokens that do not have 18 decimals is impossible.

To address this, the BootNode team modified the Lockbox implementation to handle tokens with different decimal places, allowing for decimal conversions during token deposits and redemptions. However, this introduces several concerns, such as precision loss due to decimal conversions and limited support for tokens that require upscaling.

xERC20/solidity/contracts/XERC20Lockbox.sol

```
normalizedAmount = _normalizeAmount(_amount, erc20Decimals, 18);
```

xERC20/solidity/contracts/XERC20Lockbox.sol

```
ERC20.safeTransfer(_to, _normalizeAmount(_amount, 18, erc20Decimals));
```

Originally, the decimals variable in ERC20 was intended solely for off-chain computations. The xERC20 specifications do not specify that xERC20 tokens must have 18 decimals.

Instead of handling decimal conversions in the lockbox, we recommend deploying xERC20 tokens with a matching number of decimals to avoid unneeded conversions when depositing and redeeming tokens to/from the lockbox.

Minor Streamline _deployXERC20 in XERC20Factory

Fixed

> Fixed via commits 998bb98d385e973b653df490d7083766f8f0744d and b787544eef1758f3095b917eb08c92f596696511

The `_deployXERC20` function in the `XERC20Factory` contract deploys an XERC20 token contract and pre-mints an initial supply if the `_initialSupply` parameter is set. The tokens are first minted to the factory and then transferred to the transaction sender in a subsequent step.

Similarly, ownership of the contract is initially assigned to the factory and then transferred to the `msg.sender`.

xERC20/solidity/contracts/XERC20Factory.sol

```
function _deployXERC20(
  string memory _name,
  string memory _symbol,
  uint256[] memory _minterLimits,
  uint256[] memory _burnerLimits,
  address[] memory _bridges,
  uint256 _initialSupply,
  address _owner
) internal returns (address _xerc20) {
```

xERC20/solidity/contracts/XERC20Factory.sol

```
if (_initialSupply > 0) {
  XERC20(_xerc20).transfer(msg.sender, _initialSupply);
}

XERC20(_xerc20).transferOwnership(_owner != address(0) ? _owner : msg.sender);
```

For clarity and optimization, we recommend adding additional address parameters to the XERC20 contract constructor to both mint initial tokens and transfer ownership in a single step.

Minor Refactor _deployXERC20 to Use setLimitsBatch for Improved Clarity

Fixed

> Fixed via commit b787544eef1758f3095b917eb08c92f596696511

In the `_deployXERC20` function of the `XERC20Factory` contract:

1. It verifies that the lengths of `_minterLimits`, `_burnerLimits`, and `_bridges` arrays are equal.
2. It invokes `setLimits` on the XERC20 contract with the respective parameters.

Since a `setLimitsBatch` function was introduced in the `XERC20` contract to handle these exact tasks, we recommend using it instead for clarity.

xERC20/solidity/contracts/XERC20Factory.sol

```
uint256 _bridgesLength = _bridges.length;
if (_minterLimits.length != _bridgesLength || _burnerLimits.length != _bridgesLe…
  revert IXERC20Factory_InvalidLength();
}
```

xERC20/solidity/contracts/XERC20Factory.sol

```
for (uint256 _i; _i < _bridgesLength; ++_i) {
  XERC20(_xerc20).setLimits(_bridges[_i], _minterLimits[_i], _burnerLimits[_i]);
}
```

None Fix the Documentation

Fixed

The NatSpec comment of the constructor of the `XERC20` contract contains a typo: the `_factory` parameter is defined twice.

```
xERC20/solidity/contracts/XERC20.sol

* @param _factory The factory which deployed this contract
* @param _factory The factory which deployed this contract
```

File Hashes

- `solidity/contracts/XERC20Factory.sol`
  - `f1329e52a90478bf75c585d4a6161b24b887cba7`
- `solidity/contracts/XERC20Lockbox.sol`
  - `0130caa6e77de820d5c423f52d0fe81cf9e8a059`
- `solidity/contracts/XERC20.sol`
  - `e6360ad3fd60f13267cd20e087ba33a1562ae3db`

Disclaimer

Creed ("CD") typically receives compensation from one or more clients (the "Clients") for performing the analysis contained in these reports (the "Reports"). The Reports may be distributed through other means, including via CD publications and other distributions.

The Reports are not an endorsement or indictment of any particular project or team, and the Reports do not guarantee the security of any particular project. This Report does not consider, and should not be interpreted as considering or having any bearing on, the potential economics of a token, token sale or any other product, service or other asset. Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty. No Report provides any warranty or representation to any Third-Party in any respect, including regarding the bugfree nature of code, the business model or proprietors of any such business model, and the legal compliance of any such business. No third party should rely on the Reports in any way, including for the purpose of making any decisions to buy or sell any token, product, service or other asset. Specifically, for the avoidance of doubt, this Report does not constitute investment advice, is not intended to be relied upon as investment advice, is not an endorsement of this project or team, and it is not a guarantee as to the absolute security of the project. CD owes no duty to any Third-Party by virtue of publishing these Reports.

PURPOSE OF REPORTS The Reports and the analysis described therein are created solely for Clients and published with their consent. The scope of our review is limited to a review of code and only the code we note as being within the scope of our review within this report. Any Solidity code itself presents unique and unquantifiable risks as the Solidity language itself remains under development and is subject to unknown risks and flaws. The review does not extend to the compiler layer, or any other areas beyond specified code that could present security risks. Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty. In some instances, we may perform penetration testing or infrastructure assessments depending on the scope of the particular engagement.

CD makes the Reports available to parties other than the Clients (i.e., "third parties") – on its website. CD hopes that by making these analyses publicly available, it can help the blockchain ecosystem develop technical best practices in this rapidly evolving area of innovation.

LINKS TO OTHER WEB SITES FROM THIS WEB SITE You may, through hypertext or other computer links, gain access to web sites operated by persons other than CD. Such hyperlinks are provided for your reference and convenience only, and are the exclusive responsibility of such web sites' owners. You agree that CD are not responsible for the content or operation of such Web sites, and that CD shall have no liability to you or any other person or entity for the use of third party Web sites. Except as described below, a hyperlink from this web Site to another web site does not imply or mean that CD endorses the content on that Web site or the operator or operations of that site. You are solely responsible for determining the extent to which you may use any content at any other web sites to which you link from the Reports. CD assumes no responsibility for the use of third party software on the Web Site and shall have no liability whatsoever to any person or entity for the accuracy or completeness of any outcome generated by such software.

TIMELINESS OF CONTENT The content contained in the Reports is current as of the date appearing on the Report and is subject to change without notice. Unless indicated otherwise, by CD.