Jorge Rodriguez

CYBERSEC CONTRACT AUDIT REPORT
XDEF Finance - CTDSEC.com

Introduction

During January of 2021, XDEF Finance engaged CTDSec to audit smart contracts that they created. The engagement was technical in nature and focused on identifying security flaws in the design and implementation of the contracts. XDEF Finance provided CTDSec with access to their code repository and whitepaper.

Disclaimer

The audit makes no statements or warranties about utility of the code, safety of the code, suitability of the business model, regulatory regime for the business model, or any other statements about fitness of the contracts to purpose, or their bugfree status. The audit documentation is for discussion purposes only.

I always recommend having a bug bounty program opened to detect future bugs.

Coverage

Target Code and Revision

For this audit, we performed research, investigation, and review of the XDEF Finance contract followed by issue reporting, along with mitigation and remediation instructions outlined in this report. The following code files are considered in-scope for the review:
Github: https://github.com/combinefinance

- ERC20UpgradeSafe.sol
- ERC677Token.sol
- LPToken.sol
- TokenPriceOracle.sol
- TvlOracle.sol
- XdefToken.sol
- XdefTokenMonetaryPolicy.sol
- XdefTokenOrchestrator.sol
- TokenPool.sol
- TokenGeyser.sol
- IStaking.sol

Attacks made to the contract

In order to check for the security of the contract, we tested several attacks in order to make sure that the contract is secure and follows best practices.

Correctness of the protocol implementation [Result OK]

User funds are secure on the blockchain and cannot be transferred without user permission [Result OK]

Vulnerabilities within each component as well as secure interaction between the network components [Result OK]

Correctly passing requests to the network core [Result OK]

Data privacy, data leaking, and information integrity [Result OK]

Susceptible to reentrancy attack [Result OK]

Key management implementation: secure private key storage and proper management of encryption and signing keys [Result OK]

Handling large volumes of network traffic [Result OK]

Resistance to DDoS and similar attacks [SOLVED BY DEV TEAM]

Aligning incentives with the rest of the network [Result OK]

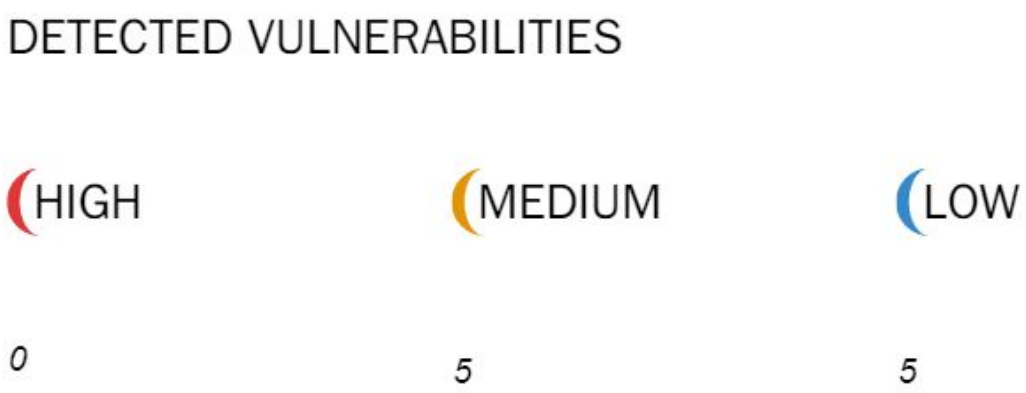
Any attack that impacts funds, such as draining or manipulating of funds [Result OK]

Mismanagement of funds via transactions [Result OK]

Inappropriate permissions and excess authority [Result OK]

Special token issuance model [Result OK]

Vulnerabilities



ISSUES

MEDIUM SWC-128

Loop over unbounded data structure.

Gas consumption in function "applyCharity" in contract "XdefTokenMonetaryPolicy" depends on the size of data structures or values that may grow unboundedly. If the data structure grows too large, the gas required to execute the code will exceed the block gas limit, effectively causing a denial-of-service condition. Consider that an attacker might attempt to cause this condition on purpose.

Locations

MEDIUM SWC-128

Loop over unbounded data structure.

Gas consumption in function "rebase" in contract "XdefTokenOrchestrator" depends on the size of data structures or values that may grow unboundedly. If the data structure grows too large, the gas required to execute the code will exceed the block gas limit, effectively causing a denial-of-service condition. Consider that an attacker might attempt to cause this condition on purpose.

Locations

```
2457  policy.rebase();
2458
2459  for (uint i = 0; i < transactionEnabled.length; i++) {
2460  // Transaction storage t = transactions[i];
2461  if (transactionEnabled[i]) {
```

MEDIUM SWC-128

Implicit loop over unbounded data structure.

Gas consumption in function "rebase" in contract "XdefTokenOrchestrator" depends on the size of data structures that may grow unboundedly. The highlighted statement involves copying the array "transactionData[i]" from "storage" to "memory". When copying arrays from "storage" to "memory" the Solidity compiler emits an implicit loop.If the array grows too large, the gas required to execute the code will exceed the block gas limit, effectively causing a denial-of-service condition. Consider that an attacker might attempt to cause this condition on purpose.

Locations

```
2460  // Transaction storage t = transactions[i];
2461  if (transactionEnabled[i]) {
2462  bool result = externalCall(transactionDestination[i], transactionData[i]);
2463  if (!result) {
2464  emit TransactionFailed(transactionDestination[i], i, transactionData[i]);
```

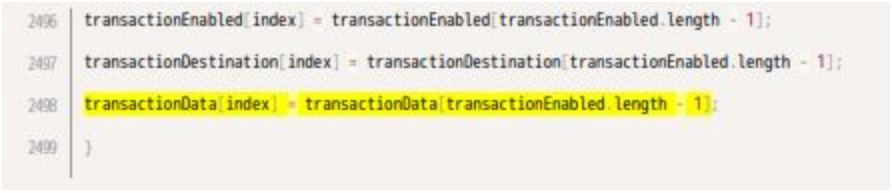
MEDIUM SWC-128

Implicit loop over unbounded data structure.

Gas consumption in function "removeTransaction" in contract "XdefTokenOrchestrator" depends on the size of data structures that may grow unboundedly. The highlighted assignment overwrites or deletes a state variable that contains an array. When assigning to or deleting storage arrays, the Solidity compiler emits an implicit clearing loop.If the array grows too large, the gas required to execute the code will exceed the block gas

limit, effectively causing a denial-of-service condition. Consider that an attacker might attempt to cause this condition on purpose.

Locations



```
2496  transactionEnabled[index] = transactionEnabled[transactionEnabled.length - 1];
2497  transactionDestination[index] = transactionDestination[transactionEnabled.length - 1];
2498  transactionData[index] = transactionData[transactionEnabled.length - 1];
2499  }
```

MEDIUM SWC-000

Function could be marked as external.

There are functions that are marked as ‘public’ that are never directly called by another function in the same contract or any of its descendants. To avoid large transaction costs we recommend to set them as ‘External’.

Location Lines:
2095, 1785, 1769, 1700, 1699, 1490, 1481

LOW - SWC-103

A floating pragma is set.

There are pragma solidity directive that aren't fixed to a specified compiler version. We recommend fixing the versions to ensure that the bytecode produced does not vary between builds. This is especially important if you rely on bytecode-level verification of the code.

Location lines:

5, 71, 112, 191, 345, 795, 1218, 1420

LOW - SWC-116

A control flow decision is made based on The block.timestamp environment variable.

The block.timestamp environment variable is used to determine a control flow decision. Note that the values of variables like coinbase, gaslimit, block number and timestamp are predictable and can be manipulated by a malicious miner. Also keep in mind that attackers know hashes of earlier blocks. Don't use any of those environment variables as sources of randomness and be aware that use of these variables introduces

a certain level of trust into miners.

Location lines:
2357, 2111, 2114, 246, 218

LOW - SWC-131

Unused function parameter "from".

The value of the function parameter "from" for the function "_beforeTokenTransfer" of contract "ERC20UpgradeSafe" does not seem to be used anywhere in "_beforeTokenTransfer".

Locations

```
717 * To learn more about hooks, head to xref:ROOT:extending-contracts.adoc#using-hooks[Using Hooks].
718 */
719 function _beforeTokenTransfer(address from, address to, uint256 amount) internal virtual { }
720
721 uint256[44] private __gap;
```

LOW - SWC-131

Unused function parameter "to".

The value of the function parameter "to" for the function "_beforeTokenTransfer" of contract "ERC20UpgradeSafe" does not seem to be used anywhere in "_beforeTokenTransfer".

Locations

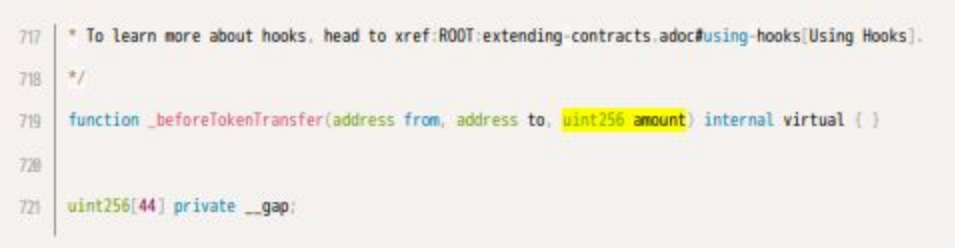
```
717 * To learn more about hooks, head to xref:ROOT:extending-contracts.adoc#using-hooks[Using Hooks].
718 */
719 function _beforeTokenTransfer(address from, address to, uint256 amount) internal virtual { }
720
721 uint256[44] private __gap;
```

LOW - SWC-131

Unused function parameter "amount".

The value of the function parameter "amount" for the function "_beforeTokenTransfer" of contract "ERC20UpgradeSafe" does not seem to be used anywhere in "_beforeTokenTransfer".

Locations



LOW - SWC-131

Unused local variable "_timestamp".

The local variable "_timestamp" is declared within the function "getData" of contract "TvlOracle" but its value does not seem to be used anywhere in "getData".

Locations

Summary of the Audit

After an exhaustive review with the development team, both security and development fixes have been applied that have improved the performance and integrity of the contract. After reviewing the modified version of the contract it is safe to deploy.