MixBytes()

# CURVE-DAO-VOTING-FORWARDER

SMART CONTRACT

AUDIT REPORT

JULY 13

2020

## FOREWORD TO REPORT

A small bug can cost you millions. **MixBytes** is a team of experienced blockchain engineers that reviews your codebase and helps you avoid potential heavy losses. More than 10 years of expertise in information security and high-load services and 18 000+ lines of audited code speak for themselves. This document outlines our methodology, scope of work, and results. We would like to thank **Curve Finance** for their trust and opportunity to audit their smart contracts.

## CONTENT DISCLAIMER

This report is public upon the consent of **Curve Finance**. **MixBytes** is not to be held responsible for any damage arising from or connected with the report. Smart contract security audit does not guarantee an inclusive analysis disclosing all possible errors and vulnerabilities but covers the majority of issues that represent threat to smart contract operation, have been overlooked or should be fixed.

# TABLE OF CONTENTS

# 01 INTRODUCTION TO THE AUDIT

## GENERAL PROVISIONS

**Curve Finance** is a project that uses liquidity pools and bonding curves to provide high-efficiency stablecoin trading and low-risk returns for liquidity providers. MixBytes was approached by Curve Finance to provide a security assessment of a part of their governance mechanism smart contracts.

## SCOPE OF THE AUDIT

The scope of the audit is smart contracts at **https://github.com/pengiundev/curve-dao-voting-forwarder/blob/db6a2694bdc34d68bf39435ae956dea0791291d6/contracts/BalanceTimeForwarder.sol**

Audited commit is db6a2694bdc34d68bf39435ae956dea0791291d6.

The specification of the contract is located **https://github.com/pengiundev/curve-dao-voting-forwarder/blob/5b116f454188fd6f12a5d44bbdec5cf64560f63a/SPECS.md**

# 02 SECURITY ASSESSMENT PRINCIPLES

## CLASSIFICATION OF ISSUES

**CRITICAL**

Bugs leading to Ether or token theft, fund access locking or any other loss of Ether/tokens to be transferred to any party (for example, dividends).

**MAJOR**

Bugs that can trigger a contract failure. Further recovery is possible only by manual modification of the contract state or replacement.

**WARNINGS**

Bugs that can break the intended contract logic or expose it to DoS attacks.

**COMMENTS**

Other issues and recommendations reported to/acknowledged by the team.

## SECURITY ASSESSMENT METHODOLOGY

The audit was performed with triple redundancy by three auditors.

Stages of the audit were as follows:

1. “Blind” manual check of the code and model behind the code
2. “Guided” manual check of the code
3. Check of adherence of the code to requirements of the client
4. Automated security analysis using internal solidity security checker
5. Automated security analysis using public analysers
6. Manual by-checklist inspection of the system
7. Discussion and merge of independent audit results
8. Report execution

# 03 DETECTED ISSUES

## CRITICAL

Not found.

## MAJOR

### 1. BalanceTimeForwarder.sol#L93

`msg.sender` is erroneously used instead of `_sender`. The `canForward` function can be called by other contracts or a frontend to check if a user can forward an action. In case of these calls `msg.sender` wouldn’t correspond to `_sender` yielding a logically incorrect result. We suggest replacing `msg.sender` with `_sender` on this line.

**Status:**

**FIXED** at **031d29f**

## WARNINGS

### 1. BalanceTimeForwarder.sol#L87

Possible reentrant call, since `lastVoteTimes` is modified after the call to `runScript`. We, therefore, suggest moving the state change before the runScript call.

**For example:**

```
lastVoteTimes[msg.sender] = block.timestamp;
runScript(_evmScript, input, blackList);
```

**Status:**

**FIXED** at **031d29f**

### 2. BalanceTimeForwarder.sol#L34

Input values `minTime`, `minBalance` are not checked in the `initialize` function.

We suggest adding checks in accordance to **the spec**:

```
require(minTime > 12 hours && minTime < 2 weeks);
require(minBalance > 10000 * 365 days);
```

They can be written as modifiers to avoid code duplication.

**Status:**

**FIXED** at **031d29f**

### 3. BalanceTimeForwarder.sol#L53
### BalanceTimeForwarder.sol#L61

Integer literals representing time period are not accurate.

* **1 year:** `60*60*24*365 == 31536000`
* **2 weeks:** `60*60*24*14 == 1209600`

For accuracy we suggest using **time units**, like `365 days`, `14 days` or `2 weeks`.

**Status:**

**FIXED** at **031d29f**

## COMMENTS

### 1. BalanceTimeForwarder.sol#L13-L15

Solidity constants are not optimized. They work like pure functions, executed upon each access.

We suggest using the following snippet:

```
bytes32 public constant SET_TOKEN_ROLE =
0x9b27b5f34c38cd0d691143dc6188f9aa90e75f5e87b428e9315e822224da1dd2; //
keccak256(“SET_TOKEN_ROLE”)
// ...

constructor() public {
    assert(SET_TOKEN_ROLE == keccak256(“SET_TOKEN_ROLE”));
    // ...
}
```

**Status:**

**FIXED** at **031d29f**

### 2. BalanceTimeForwarder.sol#L5

Unused dependency.

**Status:**

**FIXED** at **031d29f**

### 3. SPECS.md#L10

There is no voting power calculation directly in the `BalanceTimeForwarder` contract. We assume that it's calculated by the token contract and returned by the `balanceOf` call.

**Status:**

**ACKNOWLEDGED**

### 4. BalanceTimeForwarder.sol#L82-L85

These lines are copied from the TokenManager. The second line of the comment is incorrect. Also, make sure that the blacklist is needed.

**Status:**

**FIXED** at 031d29f

## 04 CONCLUSION AND RESULTS

Several troublesome issues were identified and properly addressed.

The **fixed contract** doesn't have any vulnerabilities according to our analysis.

## ABOUT MIXBYTES

**MixBytes** is a team of blockchain developers, auditors and analysts keen on decentralized systems. We build open-source solutions, smart contracts and blockchain protocols, perform security audits, work on benchmarking and software testing solutions, consult universities and enterprises, do research, publish articles and documentation.

**Stack**

**Blockchains**

## JOIN US