20/02/2023

C6Digital TeamToken smart contract

Security audit report

AUTHOR : oualid_pro
WEBSITE : https://getsecureworld.com

Summary

Introduction

We were retained by C6Digital to conduct a security audit on the C6Digital TeamToken smart contract in order to determine the security issues that could lead to some serious financial losses. All activities were conducted in a manner that simulated a malicious actor engaged in a targeted attack against it with the goals of:

- Identifying if a remote attacker could exploit a vulnerability in the contract to steal funds
- Determining the impact of a security issue on the business
- Check for any business logic vulnerabilities

We have performed a combination of manual review of the code and automated security testing to balance efficiency, timeliness, practicality, and accuracy in regard to the scope of the smart contract audit. While manual testing is recommended to uncover flaws in logic, process, and implementation, automated testing techniques help enhance coverage of smart contracts and can quickly identify items that do not follow security best practices.

Scope

The security audit was mainly performed on the following smart contract:

- C6Digital TeamToken
- Burnable C6Digital TeamToken

Vulnerabilities

Title	Reentrancy
Description	Reentrancy occurs when external contract calls are allowed to make new calls to the calling contract before the initial execution is complete.
Impact	Financial loses
Severity	High
Code location	None
Status	Checked
Result	Clean

Title	Broken Access Control
Description	The broken access control vulnerability happen when a smart contract fail to check the permissions of a user before executing its instructions.
Impact	Depends on the vulnerable functionality
Severity	High
Code location	None
Status	Checked
Result	Clean

Title	Arithmetic Issues
Description	Integers are frequently used in smart contracts to carry out computations. Yet, improper handling of integer overflow and underflow can lead to unexpected behavior and significant security vulnerabilities in a contract. When an integer exceeds its maximum value or goes below its minimum value, it is said to have an underflow.
Impact	Denial of server or financial lose
Severity	High
Code location	None
Status	Checked
Result	Clean

Title	Unchecked Return Values For Low Level Calls
Description	The low-level Solidity methods call(), callcode(), delegatecall(), and transmit are among its more complex features. They behave somewhat differently from other Solidity functions in terms of error handling since faults do not spread (or bubble up) or result in a complete reversal of the present execution. Instead, they'll provide back a boolean value that is set to false, and the program will still run. Developers may be taken aback by this, and if the return value of such low-level calls is not verified, it may result in fail-opens and other undesirable results.
Impact	Denial of server or financial lose
Severity	High
Code location	None
Status	Checked
Result	Clean

Title	Denial of Service
Description	while other types of applications can eventually recover, smart contracts can be taken offline forever by just one of these attacks. Many ways lead to denials of service, including maliciously behaving when being the recipient of a transaction, artificially increasing the gas necessary to compute a function, abusing access controls to access private components of smart contracts, taking advantage of mixups and negligence, etc.
Impact	Denial of server and financial lose
Severity	High
Code location	None
Status	Checked
Result	Clean

Title	Bad Randomness
Description	With Ethereum, randomness is difficult to achieve. Although Solidity has functions and variables that can access values that appear to be difficult to anticipate, they are often either more open than they appear to be or are sensitive to the impact of miners. Some sources of randomness can be replicated by malevolent users in order to attack the function depending on its unpredictability because they are somewhat predictable.
Impact	Denial of server and financial lose
Severity	High
Code location	None
Status	Checked
Result	Clean

Title	Front-Running
Description	Users can set larger fees to have their transactions mined more rapidly since miners are constantly compensated via gas costs for running code on behalf of externally owned addresses (EOA). Everyone may view the details of other people's pending transactions since the Ethereum blockchain is open to the public. This implies that if one person discloses the answer to a riddle or other valuable secret, another user who wishes to do them harm might clone their transaction and pay greater fees to outbid the original solution. This circumstance has the potential to give rise to useful and deadly front-running assaults, therefore smart contract creators must exercise caution.
Impact	Denial of server and financial lose
Severity	High
Code location	None
Status	Checked
Result	Clean

Title	Business logic
Description	Business logic vulnerabilities are flaws in the design and implementation of a smart contract that allow an attacker to elicit unintended behavior. Most of those vulnerabilities are not discoverable with the actual stat of art tools. Therefore, a manual test was performed to detecte those vulnerabilities.
Impact	Financial loses
Severity	High
Code location	None
Status	Checked
Result	Clean

Title	Time manipulation
Description	Checking the actual time is a common practice in building apps. In smart contract those values are coming from miners. Therefore, relying on those values to check time could create a vulnerability in the system if the node is malicious.
Impact	Financial lose
Severity	Medium
Code location	None
Status	Checked
Result	Clean