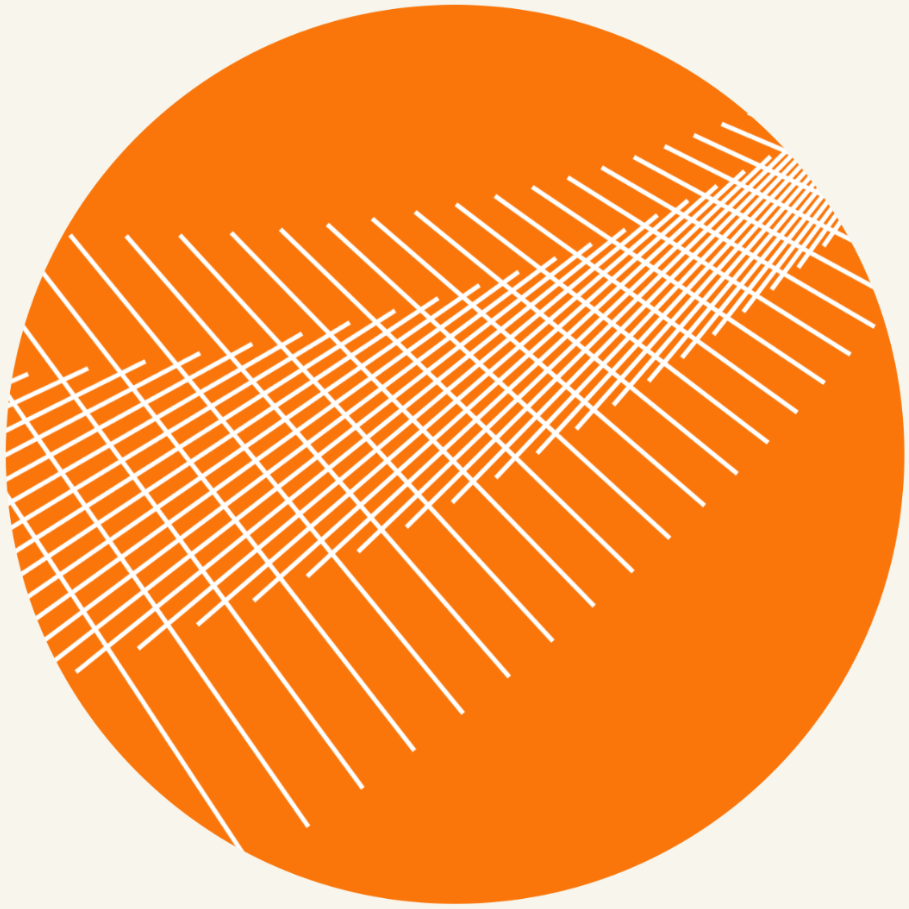
CTDSEC

YOU ARE PROTECTED



Smart contract security audit

Zbyte - Dplat Token

Table of Contents

1.0 Introduction

1.1 Project engagement

During November of 2023, Zbyte team 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. Zbyte provided CTDSec with access to their code repository and whitepaper.

1.2 Disclaimer

It should be noted that this audit is not an endorsement of the reliability or effectiveness of the contract, rather limited to an assessment of the logic and implementation. In order to ensure a secure contract that's able to withstand the network's fast-paced and rapidly changing environment, we at CTDSec recommend that Zbyte team put in place a bug bounty program to encourage further and active analysis of the smart contract.

2.0 Coverage

2.1 Target Code and Revision

For this audit, we performed research, investigation, and review of the Zbyte contracts 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:

Source file:

ERC20-main.zip: SHA256 -

773dd28e4790d83625a9242425f26235565e356c7962a5e6479f88553b56695a deployed at

https://polygonscan.com/address/0xbe7702Dc9c3fB80B96B7AD894cC012f6405E54ce#code

2.2 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.

№	Issue description.	Checking status
1	Compiler warnings.	PASSED
2	Race conditions and Reentrancy. Cross-function race conditions.	PASSED
3	Possible delays in data delivery.	PASSED
4	Oracle calls.	PASSED
5	Front running.	PASSED
6	Timestamp dependence.	PASSED
7	Integer Overflow and Underflow.	PASSED
8	DoS with Revert.	PASSED
9	DoS with block gas limit.	PASSED
10	Methods execution permissions.	PASSED
11	Economy model. If application logic is based on an incorrect economic model, the application would not function correctly and participants would incur financial losses. This type of issue is most often found in bonus rewards systems, Staking and Farming contracts, Vault and Vesting contracts, etc.	PASSED
12	The impact of the exchange rate on the logic.	PASSED
13	Private user data leaks.	PASSED
14	Malicious Event log.	PASSED
15	Scoping and Declarations.	PASSED
16	Uninitialized storage pointers.	PASSED

17	Arithmetic accuracy.	PASSED
18	Design Logic.	PASSED
19	Cross-function race conditions.	PASSED
20	Safe Zeppelin module.	PASSED
21	Fallback function security.	PASSED
22	Overpowered functions / Owner privileges	PASSED

3.0 Security Issues

3.1 High severity issues [0]

No high severity issues found.

3.2 Medium severity issues [0]

No medium severity issues found.

3.3 Low severity issues [0]

No low severity issues found.

3.4 Informational severity issues [1]

1. Block.timestamp manipulation

Block.timestamp could be manipulated to avoid waiting time for the next request at the send() function.

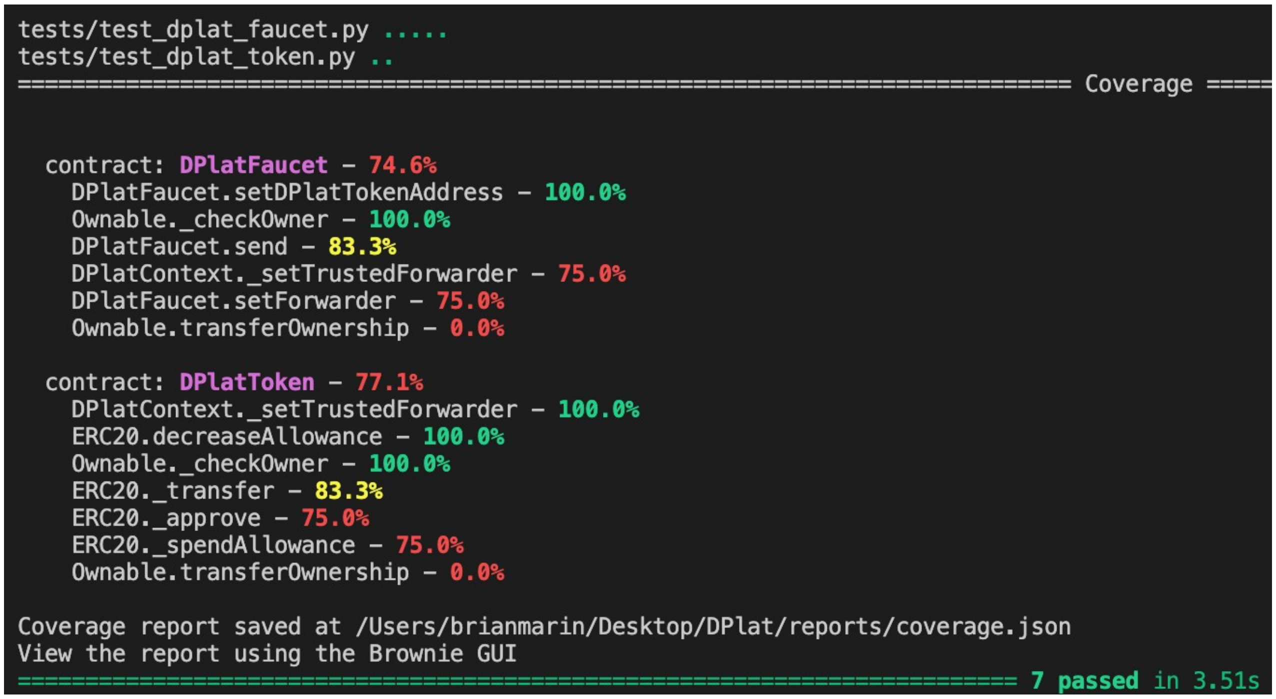
Solution:

No fix required because it is not used for a source of entropy and random number.

4.0 Testing coverage - python

During the testing phase, custom use cases were written to cover all the logic of contracts in python language. *Check "5 Annexes" to see the testing code.*

Dplat Faucet/Token tests

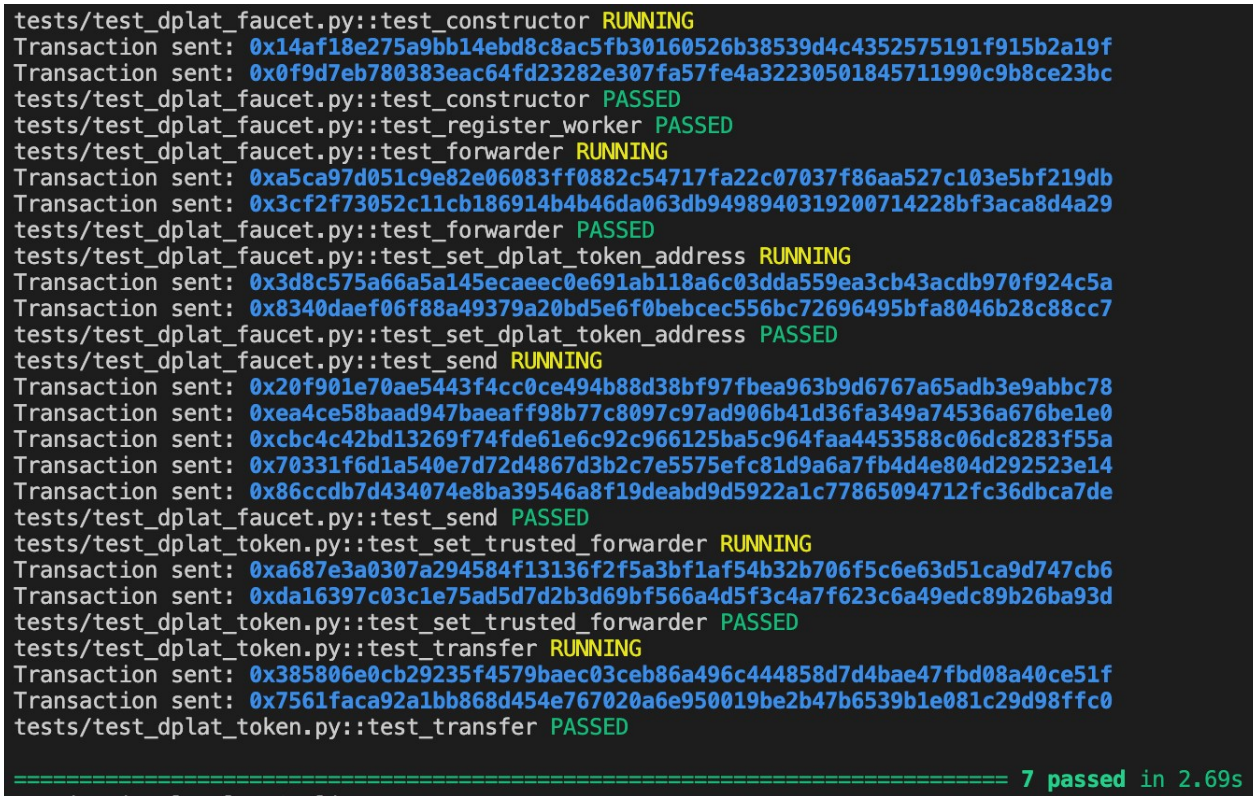


```
tests/test_dplat_faucet.py .....
tests/test_dplat_token.py ..
============================================================================== Coverage =====


  contract: DPlatFaucet - 74.6%
    DPlatFaucet.setDPlatTokenAddress - 100.0%
    Ownable._checkOwner - 100.0%
    DPlatFaucet.send - 83.3%
    DPlatContext._setTrustedForwarder - 75.0%
    DPlatFaucet.setForwarder - 75.0%
    Ownable.transferOwnership - 0.0%

  contract: DPlatToken - 77.1%
    DPlatContext._setTrustedForwarder - 100.0%
    ERC20.decreaseAllowance - 100.0%
    Ownable._checkOwner - 100.0%
    ERC20._transfer - 83.3%
    ERC20._approve - 75.0%
    ERC20._spendAllowance - 75.0%
    Ownable.transferOwnership - 0.0%

Coverage report saved at /Users/brianmarin/Desktop/DPlat/reports/coverage.json
View the report using the Brownie GUI
========================================================================== 7 passed in 3.51s
```



```
tests/test_dplat_faucet.py::test_constructor RUNNING
Transaction sent: 0x14af18e275a9bb14ebd8c8ac5fb30160526b38539d4c4352575191f915b2a19f
Transaction sent: 0x0f9d7eb780383eac64fd23282e307fa57fe4a32230501845711990c9b8ce23bc
tests/test_dplat_faucet.py::test_constructor PASSED
tests/test_dplat_faucet.py::test_register_worker PASSED
tests/test_dplat_faucet.py::test_forwarder RUNNING
Transaction sent: 0xa5ca97d051c9e82e06083ff0882c54717fa22c07037f86aa527c103e5bf219db
Transaction sent: 0x3cf2f73052c11cb186914b4b46da063db949894031920071422 8bf3aca8d4a29
tests/test_dplat_faucet.py::test_forwarder PASSED
tests/test_dplat_faucet.py::test_set_dplat_token_address RUNNING
Transaction sent: 0x3d8c575a66a5a145ecaeec0e691ab118a6c03dda559ea3cb43acdb970f924c5a
Transaction sent: 0x8340daef06f88a49379a20bd5e6f0bebcec556bc72696495bfa8046b28c88cc7
tests/test_dplat_faucet.py::test_set_dplat_token_address PASSED
tests/test_dplat_faucet.py::test_send RUNNING
Transaction sent: 0x20f901e70ae5443f4cc0ce494b88d38bf97fbea963b9d6767a65adb3e9abbc78
Transaction sent: 0xea4ce58baad947baeaff98b77c8097c97ad906b41d36fa349a74536a676be1e0
Transaction sent: 0xcbc4c42bd13269f74fde61e6c92c966125ba5c964faa4453588c06dc8283f55a
Transaction sent: 0x70331f6d1a540e7d72d4867d3b2c7e5575efc81d9a6a7fb4d4e804d292523e14
Transaction sent: 0x86ccdb7d434074e8ba39546a8f19deabd9d5922a1c77865094712fc36dbca7de
tests/test_dplat_faucet.py::test_send PASSED
tests/test_dplat_token.py::test_set_trusted_forwarder RUNNING
Transaction sent: 0xa687e3a0307a294584f13136f2f5a3bf1af54b32b706f5c6e63d51ca9d747cb6
Transaction sent: 0xda16397c03c1e75ad5d7d2b3d69bf566a4d5f3c4a7f623c6a49edc89b26ba93d
tests/test_dplat_token.py::test_set_trusted_forwarder PASSED
tests/test_dplat_token.py::test_transfer RUNNING
Transaction sent: 0x385806e0cb29235f4579baec03ceb86a496c444858d7d4bae47fbd08a40ce51f
Transaction sent: 0x7561faca92a1bb868d454e767020a6e950019be2b47b6539b1e081c29d98ffc0
tests/test_dplat_token.py::test_transfer PASSED

============================================================================ 7 passed in 2.69s
```

5.0 Annexes

Testing code:

Dplat Faucet:

```
from brownie import (
    reverts,
    DPlatFaucet
)

from scripts.helpful_scripts import (
    ZERO_ADDRESS,
    get_account,
    evm_increase_time,
    get_timestamp,
    get_custom_error_hex
)

from scripts.deploy import (
    deploy_dplat_faucet,
    deploy_dplat_token
)

def test_constructor(only_local):
    #arrange
    owner = get_account(0)
    forwarder = get_account(1)
    token = deploy_dplat_token(forwarder, owner)

    # assert
    with reverts('typed error: ' + get_custom_error_hex("ZeroAddress()")):
        DPlatFaucet.deploy(ZERO_ADDRESS, token.address, {"from": owner})
    with reverts('typed error: ' + get_custom_error_hex("ZeroAddress()")):
        DPlatFaucet.deploy(forwarder, ZERO_ADDRESS, {"from": owner})

    faucet = DPlatFaucet.deploy(forwarder, token.address, {"from": owner})

def test_register_worker():
    pass

def test_forwarder(only_local):
    #arrange
    owner = get_account(0)
    forwarder = get_account(1)
    not_owner = get_account(2)
    new_forwarder = get_account(3)
    dplat_faucet = deploy_dplat_faucet(owner, forwarder)

    # assert
    with reverts():
        dplat_faucet.setForwarder(new_forwarder, {"from": not_owner})
    with reverts('typed error: ' + get_custom_error_hex("ZeroAddress()")):
        dplat_faucet.setForwarder(ZERO_ADDRESS, {"from": owner})

    assert dplat_faucet.isTrustedForwarder(forwarder) == True
    dplat_faucet.setTrustedForwarder(new_forwarder, {"from": owner})
    assert dplat_faucet.isTrustedForwarder(forwarder) == False

def test_set_dplat_token_address(only_local):
    #arrange
    owner = get_account(0)
    forwarder = get_account(1)
    not_owner = get_account(2)
    new_token = deploy_dplat_token(owner, forwarder)
    token = deploy_dplat_token(owner, forwarder)
    dplat_faucet = DPlatFaucet.deploy(forwarder, token.address, {"from":
owner})

    # assert
    with reverts():
        dplat_faucet.setDPlatTokenAddress(new_token.address, {"from":
not_owner})
    with reverts('typed error: ' + get_custom_error_hex("ZeroAddress()")):
        dplat_faucet.setDPlatTokenAddress(ZERO_ADDRESS, {"from": owner})
    tx = dplat_faucet.setDPlatTokenAddress(new_token.address, {"from":
owner})

    assert tx.events["DPlatTokenAddressSet"] is not None
    assert tx.events["DPlatTokenAddressSet"].items()[0][1] ==
new_token.address
```

```
def test_send(only_local):
    #arrange
    owner = get_account(0)
    forwarder = get_account(1)
    worker = get_account(2)
    other = get_account(3)
    token = deploy_dplat_token(owner, forwarder)
    dplat_faucet = DPlatFaucet.deploy(forwarder, token.address, {"from":
owner})

    # assert
    with reverts(): # no worker
        dplat_faucet.send(other, {"from": owner})
    with reverts(): # no owner
        dplat_faucet.registerWorker(worker, True, {"from": other})
    dplat_faucet.registerWorker(worker, True, {"from": owner})
    with reverts('typed error:
0xb4fdb9e30000000000000000000000000000000000000000000000000000000000000000'
): # Faucet low balance
        dplat_faucet.send(other, {"from": worker})

    with reverts(): # no owner
        dplat_faucet.setFaucetTokenDripAmount(1e18, {"from": other})
    dplat_faucet.setFaucetTokenDripAmount(1e18, {"from": owner})
    # Send some tokens to contract DPlatFaucet
    token.transfer(dplat_faucet.address, 2e18, {"from": owner})
    assert token.balanceOf(other) == 0
    dplat_faucet.send(other, {"from": worker})
    assert token.balanceOf(other) == 1e18

    with reverts(): # TokenRequestTooEarly
        dplat_faucet.send(other, {"from": worker})

    evm_increase_time(60 * 5)
    token.transfer(dplat_faucet.address, 1e18, {"from": owner})

    assert token.balanceOf(other) == 1e18
    dplat_faucet.send(other, {"from": worker})
    assert token.balanceOf(other) == 2e18
```

<u>Dplat Token:</u>

```python
from brownie import (
    reverts
)

from scripts.helpful_scripts import (
    ZERO_ADDRESS,
    get_account,
    get_custom_error_hex
)

from scripts.deploy import (
    deploy_dplat_token
)

def test_set_trusted_forwarder(only_local):
    #arrange
    owner = get_account(0)
    forwarder = get_account(1)
    not_owner = get_account(2)
    new_forwarder = get_account(3)
    dplat_token = deploy_dplat_token(owner, forwarder)

    # assert
    with reverts():
        dplat_token.setTrustedForwarder(new_forwarder, {"from": not_owner})
    with reverts('typed error: ' + get_custom_error_hex("ZeroAddress()")):
        dplat_token.setTrustedForwarder(ZERO_ADDRESS, {"from": owner})

    assert dplat_token.isTrustedForwarder(forwarder) == True
    dplat_token.setTrustedForwarder(new_forwarder, {"from": owner})
    assert dplat_token.isTrustedForwarder(forwarder) == False

def test_transfer(only_local):
    #arrange
    owner = get_account(0)
    forwarder = get_account(1)
    other = get_account(2)
    account = get_account(3)
    dplat_token = deploy_dplat_token(owner, forwarder)
```

```
# assert
with reverts():
    dplat_token.transfer(owner, 1e18, {"from": other})

# test transfer
assert dplat_token.balanceOf(other) == 0
dplat_token.transfer(other, 1e18, {"from": owner})
assert dplat_token.balanceOf(other) == 1e18

# test transfer with approve
assert dplat_token.balanceOf(account) == 0
dplat_token.approve(account, 1e18, {"from": other})
dplat_token.transferFrom(other, account, 1e18, {"from": account})
assert dplat_token.balanceOf(account) == 1e18

# test decrease allowance
dplat_token.approve(account, 1e18, {"from": other})
with reverts():
    dplat_token.decreaseAllowance(account, 2e18, {"from": other})

assert dplat_token.allowance(other, account) == 1e18
dplat_token.decreaseAllowance(account, 1e18, {"from": other})
assert dplat_token.allowance(other, account) == 0
```

6.0 Summary of the audit

No high or medium criticality vulnerabilities were identified within the Zbyte Dplat contracts during the thorough security assessment