Ether Authority

www.EtherAuthority.io
audit@etherauthority.io

SMART CONTRACT

Security Audit Report

Customer:	DePay
Website:	https://depay.fi
Platform:	Binance Smart Chain
Language:	Solidity
Date:	August 27th, 2021

Table of contents

Email: audit@EtherAuthority.io

THIS IS SECURITY AUDIT REPORT DOCUMENT AND WHICH MAY CONTAIN INFORMATION WHICH IS CONFIDENTIAL. WHICH INCLUDES ANY POTENTIAL VULNERABILITIES AND MALICIOUS CODES WHICH CAN BE USED TO EXPLOIT THE SOFTWARE. THIS MUST BE REFERRED INTERNALLY AND ONLY SHOULD BE MADE AVAILABLE TO PUBLIC AFTER ISSUES ARE RESOLVED.

Email: audit@EtherAuthority.io

Introduction

EtherAuthority was contracted by the DePay team to perform the Security audit of the DePay Launchpad V1 smart contract code. The audit has been performed using manual analysis as well as using automated software tools. This report presents all the findings regarding the audit performed on August 27th, 2021.

The purpose of this audit was to address the following:

- Ensure that all claimed functions exist and function correctly.
- Identify any security vulnerabilities that may be present in the smart contract.

Project Background

DePay Launchpad V1 is a smart contract to whitelist, claim and release tokens in a launchpad fashion. This launchpad smart contract allows users to participate in a token launch.

Audit scope

Name	**Code Review and Security Analysis Report for DePay Launchpad V1 Smart Contract**
Platform	**BSC / Solidity**
File	DePayLaunchpadV1.sol
Smart Contract Online Code	https://github.com/DePayFi/depay-evm-launchpad/blob/master/contracts/DePayLaunchpadV1.sol
Branch	Master
Git Commit	996f7f6db52fb5f1dc216c8484b016cffa3d7482
Audit Date	August 27th, 2021

Claimed Smart Contract Features

Claimed Feature Detail	Our Observation
Launchpad Platform: Binance Smart Chain Payment Tokens: will be set at launchpad initialization Launchpad Tokens: will be set at launchpad initialization Split release amount: will be set at launchpad initialization Split release address: will be set at launchpad initialization Launchpad End time: will be set while starting the launchpad Token price: will be set while starting the launchpad	**YES, This is valid.**
Owner's functions: • init: This initiates the launchpad • start: Owner can start the launchpad by providing the launchpad end time and price of the token. • whitelistAddress: Owner can set whitelisted addresses, which can participate in the launchpad once started. • whitelistAddresses: Owner can whitelist many wallets. • releasePayments: Owner can release payment to himself once launchpad is over. • releaseUnclaimed: Owner can release unclaimed tokens to himself once launchpad is over.	**YES, This is valid. But please keep the private key of the owner wallet secure, because if that is compromised, then it would create problems.**

Audit Summary

According to the standard audit assessment, Customer`s solidity smart contracts are **"Secured"**. These contracts also have owner functions (described in the centralization section below), which does not make everything 100% decentralized. Thus, the owner must execute those smart contract functions as per the business plan.

We used various tools like MythX, Slither and Remix IDE. At the same time this finding is based on critical analysis of the manual audit.
All issues found during automated analysis were manually reviewed and applicable vulnerabilities are presented in the Audit overview section. General overview is presented in AS-IS section and all identified issues can be found in the Audit overview section.

We found 0 critical, 0 high, 1 medium and 2 low and some very low level issues.

Investors Advice: Technical audit of the smart contract does not guarantee the ethical nature of the project. Any owner controlled functions should be executed by the owner with responsibility. All investors/users are advised to do their due diligence before investing in the project.

Technical Quick Stats

Main Category	Subcategory	Result
Security Attack vectors	Security of user funds	Passed
	Front-running	Passed
	DOS	Passed
	Integer overflow/underflow	Passed
	Reentrancy	Passed
	Timestamp Dependence	Passed
	Owner trust mitigation	Passed
Contract Programming	Solidity version too old	Moderated
	Function input parameters lack of check	Passed
	Function input parameters check bypass	Passed
	Function access control lacks management	Passed
	Critical operation lacks event log	Moderated
	Random number generation/use vulnerability	Passed
	Fallback function misuse	Passed
	Race condition	Passed
	Logical vulnerability	Passed
	Other programming issues	Passed
Code Specification	Function visibility not explicitly declared	Passed
	Var. storage location not explicitly declared	Passed
	Use keywords/functions to be deprecated	Passed
	Other code specification issues	Passed
Gas Optimization	“Out of Gas” Issue	Passed
	High consumption ‘for/while’ loop	Moderated
	High consumption ‘storage’ storage	Passed
	Assert() misuse	Passed
Business Risk	The maximum limit for mintage not set	Passed
	“Short Address” Attack	Passed
	“Double Spend” Attack	Passed

Overall Audit Result: **PASSED**

Code Quality

This audit scope has 1 smart contract. This smart contract also contains Libraries, Smart contracts inherits and Interfaces. These are compact and well written contracts. The imported contracts are from third party provider OpenZeppelin, which are out of the audit scope and thus are **not** audited.

The libraries in DePay launchpad V1 are part of its logical algorithm. Once deployed on the blockchain (only once), it is assigned a specific address and its properties / methods can be reused many times by other contracts.

The DePay Token team has provided scenario and unit test scripts, which have helped to determine the integrity of the code in an automated way.

Documentation

We were given a DePay Launchpad V1 smart contract code in the form of a public github repository. The weblink and commit of that code is mentioned above in the table.

As mentioned above, some code parts are **well** commented. So it is easy to quickly understand the programming flow as well as complex code logic. Comments are very helpful in understanding the overall architecture of the protocol.

Another source of information was its official website https://depay.fi/ which provided rich information about the project architecture and tokenomics.

Use of Dependencies

As per our observation, the libraries are used in this smart contract infrastructure that are based on well known industry standard open source projects. And their core code blocks are written well.

Apart from libraries, its functions are used in external smart contract calls.

AS-IS overview

(1) imports

(a) Ownable

(b) ReentrancyGuard

(c) ERC20

(d) SafeERC20

(e) SafeMath

(2) Inherited contracts

(a) Ownable

(b) ReentrancyGuard

(3) Usages

(a) using SafeMath for uint

(b) using SafeERC20 for ERC20

(4) Functions

Sl.	Functions	Type	Observation	Conclusion
1	init	write	Owner function	No Issue
2	start	write	Owner function	No Issue
3	_whitelistAddress	internal	Passed	No Issue
4	whitelistAddress	write	Owner function	No Issue
5	whitelistAddresses	write	Infinite loop possibility	Refer audit findings section below
6	claim	write	Passed	No Issue
7	_release	internal	Passed	No Issue
8	release	write	Passed	No Issue
9	multiRelease	write	Infinite loop possibility	Refer audit findings section below
10	releasePayments	write	Owner function	No Issue
11	releaseUnclaimed	write	Owner function	No Issue

Severity Definitions

Risk Level	Description
Critical	Critical vulnerabilities are usually straightforward to exploit and can lead to token loss etc.
High	High-level vulnerabilities are difficult to exploit; however, they also have significant impact on smart contract execution, e.g. public access to crucial
Medium	Medium-level vulnerabilities are important to fix; however, they can't lead to tokens lose
Low	Low-level vulnerabilities are mostly related to outdated, unused etc. code snippets, that can't have significant impact on execution
Lowest / Code Style / Best Practice	Lowest-level vulnerabilities, code style violations and info statements can't affect smart contract execution and can be ignored.

Audit Findings

Critical

No Critical severity vulnerabilities were found.

High

No High severity vulnerabilities were found.

Medium

(1) Multiplication before division

```
function claim(
  address forAddress,
  uint256 claimedAmount,
  bool splitRelease
) external onlyInProgress nonReentrant returns(bool) {
  require(whitelist[forAddress], 'Address has not been whitelisted for this launch!');
  if(splitRelease && splitReleases[forAddress] == false){ require(claimedAmount > splitReleas
  uint256 payedAmount = claimedAmount.div(10**ERC20(paymentToken).decimals()).mul(price);
  ERC20(paymentToken).safeTransferFrom(msg.sender, address(this), payedAmount);
  claims[forAddress] = claims[forAddress].add(claimedAmount);
  splitReleases[forAddress] = splitRelease;
  totalClaimed = totalClaimed.add(claimedAmount);
  require(totalClaimed <= totalClaimable, 'Claiming attempt exceeds totalClaimable amount!');
  return true;
```

The Claim function has a formula where there is multiplication before division. It will cause unexpected behavior due to solidity's resource constraints. We recommend placing all the multiplication first and then dividing it by any values.

Low

(1) Infinite loop possibility

```
function whitelistAddresses(
  address[] memory _addresses,
  bool status
) external onlyOwner returns(bool) {
  for (uint256 i = 0; i < _addresses.length; i++) {
    require(_whitelistAddress(_addresses[i], status));
  }
```

The functions whitelistAddresses and multiRelease have a loop, which can go infinite if the addresses are too many. Although these are owner functions so the impact is minor, we recommend placing some limit on the number of addresses provided in function input.

(2) Critical functions lack events

```
function claim(
  address forAddress,
  uint256 claimedAmount,
  bool splitRelease
```

All state changing functions should emit an event to properly notify the clients. We recommend emitting an event inside the following functions:

- init
- start
- _whitelistAddress
- claim
- _release
- releasePayments
- releaseUnclaimed

Very Low / Informational / Best practices:

(1) Make variables constant

```
address public launchedToken;

address public paymentToken;
```

Some variable's values will be unchanged. So, please make it constant. It will save some gas. Just put a constant keyword. Following variables can be made constant.

- launchedToken
- paymentToken
- splitReleaseAddress
- splitReleaseAmount

PS: init function code must be placed inside the constructor in order to make all above variables constant.

(2) Avoid using the SafeMath library, since the solidity version above 0.8.0 is used. Because it has in-build support for overflow/underflow prevention. This will save gas.

(3) Use the latest solidity version while deploying

```
pragma solidity >=0.8.6 <0.9.0;
```

We recommend using the latest solidity compiler when deploying the contract, which is 0.8.7 at the time of this audit.

(4) The variable totalClaimable should be zero

```
function releaseUnclaimed() external onlyEnded onlyOwner nonReentrant
  uint256 unclaimed = totalClaimable-totalClaimed;
  ERC20(launchedToken).safeTransfer(owner(), unclaimed);
  totalClaimable = totalClaimable.sub(unclaimed);
  return true;
```

Here, totalClaimable variable can be simply zero. because all claimed and unclaimed tokens are out of this contract.

(5) Unnecessary condition

```
modifier onlyInProgress() {
  require(
    endTime > 0,
    "Launchpad has not been started yet!"
  );
  require(
    endTime > block.timestamp,
    "Launchpad has been finished!"
  );
```

The second condition endTime > block.time will always dominate the first condition endTime > 0. So, the first condition can be safely removed. It will save little gas.

But, on the other hand, this condition gives a more specific reason to revert. So, in that case, this is useful.

So, if this is part of the plan, then this point can be safely ignored.

Conclusion

We were given a contract code. And we have used all possible tests based on given objects as files. We observed some issues in the smart contracts and those issues are not critical ones. So, **it's good to go to production**.

Since possible test cases can be unlimited for such smart contracts protocol, we provide no such guarantee of future outcomes. We have used all the latest static tools and manual observations to cover maximum possible test cases to scan everything.

Smart contracts within the scope were manually reviewed and analyzed with static analysis tools. Smart Contract's high level description of functionality was presented in As-is overview section of the report.

Audit report contains all found security vulnerabilities and other issues in the reviewed code.

Security state of the reviewed contract, based on standard audit procedure scope, is **"Secured".**

Our Methodology

We like to work with a transparent process and make our reviews a collaborative effort. The goals of our security audits are to improve the quality of systems we review and aim for sufficient remediation to help protect users. The following is the methodology we use in our security audit process.

Manual Code Review:

In manually reviewing all of the code, we look for any potential issues with code logic, error handling, protocol and header parsing, cryptographic errors, and random number generators. We also watch for areas where more defensive programming could reduce the risk of future mistakes and speed up future audits. Although our primary focus is on the in-scope code, we examine dependency code and behavior when it is relevant to a particular line of investigation.

Vulnerability Analysis:

Our audit techniques included manual code analysis, user interface interaction, and whitebox penetration testing. We look at the project's web site to get a high level understanding of what functionality the software under review provides. We then meet with the developers to gain an appreciation of their vision of the software. We install and use the relevant software, exploring the user interactions and roles. While we do this, we brainstorm threat models and attack surfaces. We read design documentation, review other audit results, search for similar projects, examine source code dependencies, skim open issue tickets, and generally investigate details other than the implementation.

Documenting Results:

We follow a conservative, transparent process for analyzing potential security vulnerabilities and seeing them through successful remediation. Whenever a potential issue is discovered, we immediately create an Issue entry for it in this document, even though we have not yet verified the feasibility and impact of the issue. This process is conservative because we document our suspicions early even if they are later shown to not represent exploitable vulnerabilities. We generally follow a process of first documenting the suspicion with unresolved questions, then confirming the issue through code analysis, live experimentation, or automated tests. Code analysis is the most tentative, and we strive to provide test code, log captures, or screenshots demonstrating our confirmation. After this we analyze the feasibility of an attack in a live system.

Suggested Solutions:

We search for immediate mitigations that live deployments can take, and finally we suggest the requirements for remediation engineering for future releases. The mitigation and remediation recommendations should be scrutinized by the developers and deployment engineers, and successful mitigation and remediation is an ongoing collaborative process after we deliver our report, and before the details are made public.

Disclaimers

EtherAuthority.io Disclaimer

EtherAuthority team has analyzed this smart contract in accordance with the best industry practices at the date of this report, in relation to: cybersecurity vulnerabilities and issues in smart contract source code, the details of which are disclosed in this report, (Source Code); the Source Code compilation, deployment and functionality (performing the intended functions).

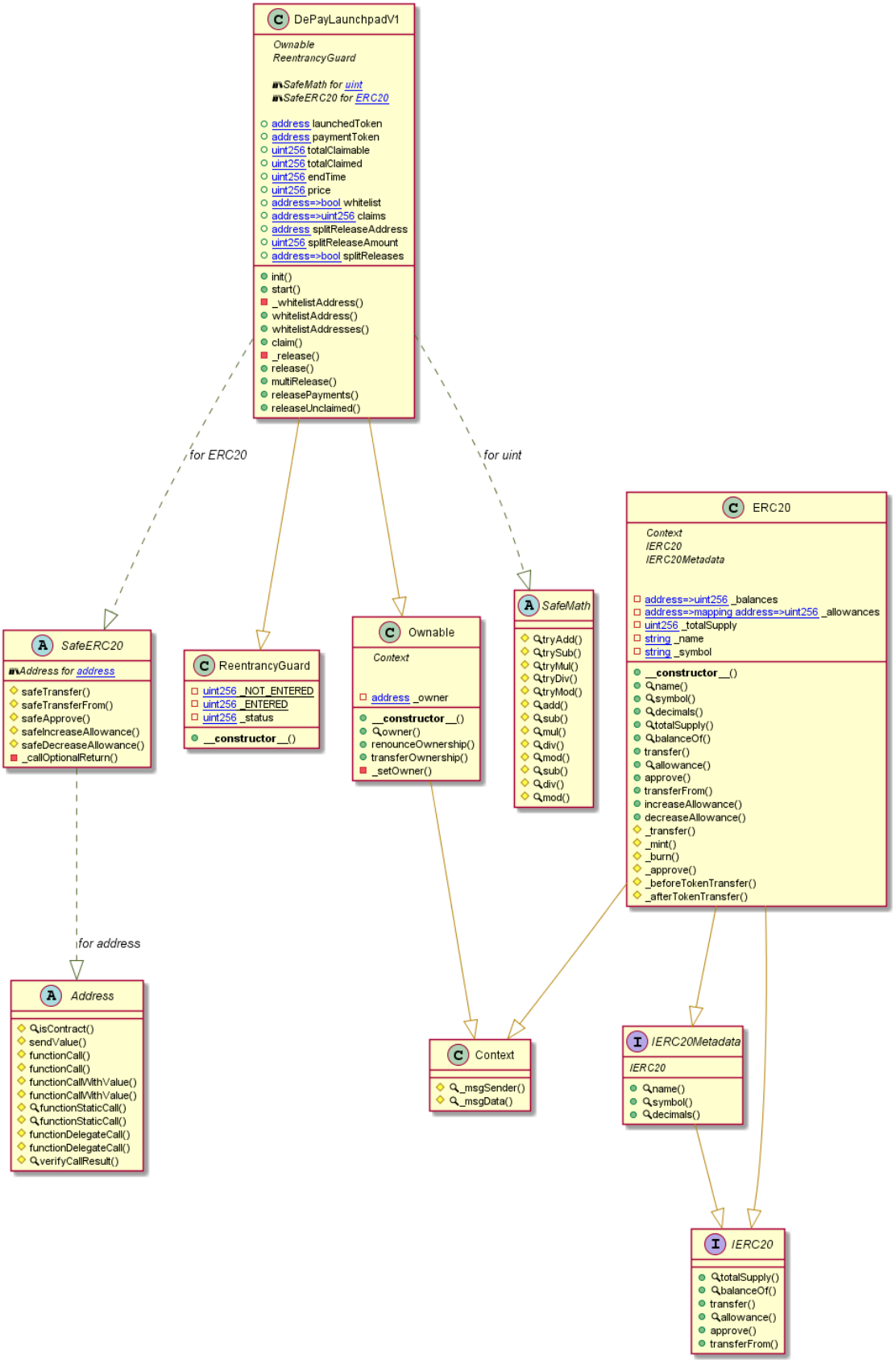
Due to the fact that the total number of test cases are unlimited, the audit makes no statements or warranties on security of the code. It also cannot be considered as a sufficient assessment regarding the utility and safety of the code, bugfree status or any other statements of the contract. While we have done our best in conducting the analysis and producing this report, it is important to note that you should not rely on this report only. We also suggest conducting a bug bounty program to confirm the high level of security of this smart contract.

Technical Disclaimer

Smart contracts are deployed and executed on the blockchain platform. The platform, its programming language, and other software related to the smart contract can have their own vulnerabilities that can lead to hacks. Thus, the audit can’t guarantee explicit security of the audited smart contracts.

Appendix

Code Flow Diagram - DePay Launchpad V1



Slither Results Log

Slither log >> DePayLaunchpadV1.sol

```
INFO:Detectors:
DePayLaunchpadV1.releasePayments() (../chetan/gaza/mycontracts/DePayLaunchpadV1.sol#1410-1413) ignores return value by ERC20(paymentToken
).transfer(owner(),ERC20(paymentToken).balanceOf(address(this))) (../chetan/gaza/mycontracts/DePayLaunchpadV1.sol#1411)
DePayLaunchpadV1.releaseUnclaimed() (../chetan/gaza/mycontracts/DePayLaunchpadV1.sol#1416-1421) ignores return value by ERC20(launchedTok
en).transfer(owner(),unclaimed) (../chetan/gaza/mycontracts/DePayLaunchpadV1.sol#1418)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unchecked-transfer
INFO:Detectors:
DePayLaunchpadV1.claim(address,uint256,bool) (../chetan/gaza/mycontracts/DePayLaunchpadV1.sol#1344-1359) performs a multiplication on the
 result of a division:
        -payedAmount = claimedAmount.div(10 ** ERC20(paymentToken).decimals()).mul(price) (../chetan/gaza/mycontracts/DePayLaunchpadV1.so
l#1352)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply
INFO:Detectors:
Reentrancy in DePayLaunchpadV1._release(address) (../chetan/gaza/mycontracts/DePayLaunchpadV1.sol#1376-1389):
        External calls:
        - ERC20(launchedToken).safeTransfer(splitReleaseAddress,splitReleaseAmount) (../chetan/gaza/mycontracts/DePayLaunchpadV1.sol#1382
)
        - ERC20(launchedToken).safeTransfer(forAddress,claimedAmount.sub(splitReleaseAmount)) (../chetan/gaza/mycontracts/DePayLaunchpadV
1.sol#1383)
        - ERC20(launchedToken).safeTransfer(forAddress,claimedAmount) (../chetan/gaza/mycontracts/DePayLaunchpadV1.sol#1385)
        State variables written after the call(s):
        - claims[forAddress] = 0 (../chetan/gaza/mycontracts/DePayLaunchpadV1.sol#1387)
Reentrancy in DePayLaunchpadV1.claim(address,uint256,bool) (../chetan/gaza/mycontracts/DePayLaunchpadV1.sol#1344-1359):
        External calls:
        - ERC20(paymentToken).safeTransferFrom(msg.sender,address(this),payedAmount) (../chetan/gaza/mycontracts/DePayLaunchpadV1.sol#135
3)
        State variables written after the call(s):
        - splitReleases[forAddress] = splitRelease (../chetan/gaza/mycontracts/DePayLaunchpadV1.sol#1355)
Reentrancy in DePayLaunchpadV1.releaseUnclaimed() (../chetan/gaza/mycontracts/DePayLaunchpadV1.sol#1416-1421):
        External calls:
        - ERC20(launchedToken).transfer(owner(),unclaimed) (../chetan/gaza/mycontracts/DePayLaunchpadV1.sol#1418)
        State variables written after the call(s):
        - totalClaimable = totalClaimable.sub(unclaimed) (../chetan/gaza/mycontracts/DePayLaunchpadV1.sol#1419)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1
```

```
INFO:Detectors:
DePayLaunchpadV1.init(address,address,uint256,address)._launchedToken (../chetan/gaza/mycontracts/DePayLaunchpadV1.sol#1253) lacks a zero
-check on :
                - launchedToken = _launchedToken (../chetan/gaza/mycontracts/DePayLaunchpadV1.sol#1258)
DePayLaunchpadV1.init(address,address,uint256,address)._paymentToken (../chetan/gaza/mycontracts/DePayLaunchpadV1.sol#1254) lacks a zero-
check on :
                - paymentToken = _paymentToken (../chetan/gaza/mycontracts/DePayLaunchpadV1.sol#1259)
DePayLaunchpadV1.init(address,address,uint256,address)._splitReleaseAddress (../chetan/gaza/mycontracts/DePayLaunchpadV1.sol#1256) lacks
a zero-check on :
                - splitReleaseAddress = _splitReleaseAddress (../chetan/gaza/mycontracts/DePayLaunchpadV1.sol#1260)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
Reentrancy in DePayLaunchpadV1.claim(address,uint256,bool) (../chetan/gaza/mycontracts/DePayLaunchpadV1.sol#1344-1359):
        External calls:
        - ERC20(paymentToken).safeTransferFrom(msg.sender,address(this),payedAmount) (../chetan/gaza/mycontracts/DePayLaunchpadV1.sol#135
3)
        State variables written after the call(s):
        - claims[forAddress] = claims[forAddress].add(claimedAmount) (../chetan/gaza/mycontracts/DePayLaunchpadV1.sol#1354)
        - totalClaimed = totalClaimed.add(claimedAmount) (../chetan/gaza/mycontracts/DePayLaunchpadV1.sol#1356)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-2
INFO:Detectors:
DePayLaunchpadV1.start(uint256,uint256) (../chetan/gaza/mycontracts/DePayLaunchpadV1.sol#1288-1297) uses timestamp for comparisons
        Dangerous comparisons:
        - require(bool,string)(_endTime > block.timestamp,endTime needs to be in the future!) (../chetan/gaza/mycontracts/DePayLaunchpadV
1.sol#1292)
        - require(bool,string)(_endTime < (block.timestamp + 7257600),endTime needs to be less than 12 weeks in the future!) (../chetan/g
aza/mycontracts/DePayLaunchpadV1.sol#1293)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
INFO:Detectors:
Address.isContract(address) (../chetan/gaza/mycontracts/DePayLaunchpadV1.sol#669-679) uses assembly
        - INLINE ASM (../chetan/gaza/mycontracts/DePayLaunchpadV1.sol#675-677)
Address.verifyCallResult(bool,bytes,string) (../chetan/gaza/mycontracts/DePayLaunchpadV1.sol#838-858) uses assembly
        - INLINE ASM (../chetan/gaza/mycontracts/DePayLaunchpadV1.sol#850-853)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
DePayLaunchpadV1.claim(address,uint256,bool) (../chetan/gaza/mycontracts/DePayLaunchpadV1.sol#1344-1359) compares to a boolean constant:
        -splitRelease == false (../chetan/gaza/mycontracts/DePayLaunchpadV1.sol#1351)
```

```
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
DePayLaunchpadV1.claim(address,uint256,bool) (../chetan/gaza/mycontracts/DePayLaunchpadV1.sol#1344-1359) compares to a boolean constant:
        -splitRelease == false (../chetan/gaza/mycontracts/DePayLaunchpadV1.sol#1351)
DePayLaunchpadV1.claim(address,uint256,bool) (../chetan/gaza/mycontracts/DePayLaunchpadV1.sol#1344-1359) compares to a boolean constant:
        -require(bool,string)(splitReleases[forAddress] == false,You cannot change splitRelease once set!) (../chetan/gaza/mycontracts/De
PayLaunchpadV1.sol#1351)
DePayLaunchpadV1.claim(address,uint256,bool) (../chetan/gaza/mycontracts/DePayLaunchpadV1.sol#1344-1359) compares to a boolean constant:
        -splitRelease && splitReleases[forAddress] == false (../chetan/gaza/mycontracts/DePayLaunchpadV1.sol#1350)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#boolean-equality
INFO:Detectors:
Address.functionCall(address,bytes) (../chetan/gaza/mycontracts/DePayLaunchpadV1.sol#722-724) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256) (../chetan/gaza/mycontracts/DePayLaunchpadV1.sol#751-757) is never used and should b
e removed
Address.functionDelegateCall(address,bytes) (../chetan/gaza/mycontracts/DePayLaunchpadV1.sol#811-813) is never used and should be removed
Address.functionDelegateCall(address,bytes,string) (../chetan/gaza/mycontracts/DePayLaunchpadV1.sol#821-830) is never used and should be
removed
Address.functionStaticCall(address,bytes) (../chetan/gaza/mycontracts/DePayLaunchpadV1.sol#784-786) is never used and should be removed
Address.functionStaticCall(address,bytes,string) (../chetan/gaza/mycontracts/DePayLaunchpadV1.sol#794-803) is never used and should be re
moved
Address.sendValue(address,uint256) (../chetan/gaza/mycontracts/DePayLaunchpadV1.sol#697-702) is never used and should be removed
Context._msgData() (../chetan/gaza/mycontracts/DePayLaunchpadV1.sol#26-28) is never used and should be removed
ERC20._burn(address,uint256) (../chetan/gaza/mycontracts/DePayLaunchpadV1.sol#559-574) is never used and should be removed
ERC20._mint(address,uint256) (../chetan/gaza/mycontracts/DePayLaunchpadV1.sol#536-546) is never used and should be removed
SafeERC20.safeApprove(IERC20,address,uint256) (../chetan/gaza/mycontracts/DePayLaunchpadV1.sol#906-919) is never used and should be remov
ed
SafeERC20.safeDecreaseAllowance(IERC20,address,uint256) (../chetan/gaza/mycontracts/DePayLaunchpadV1.sol#930-941) is never used and shoul
d be removed
SafeERC20.safeIncreaseAllowance(IERC20,address,uint256) (../chetan/gaza/mycontracts/DePayLaunchpadV1.sol#921-928) is never used and shoul
d be removed
SafeMath.div(uint256,uint256,string) (../chetan/gaza/mycontracts/DePayLaunchpadV1.sol#1153-1162) is never used and should be removed
SafeMath.mod(uint256,uint256) (../chetan/gaza/mycontracts/DePayLaunchpadV1.sol#1113-1115) is never used and should be removed
SafeMath.mod(uint256,uint256,string) (../chetan/gaza/mycontracts/DePayLaunchpadV1.sol#1179-1188) is never used and should be removed
SafeMath.sub(uint256,uint256,string) (../chetan/gaza/mycontracts/DePayLaunchpadV1.sol#1130-1139) is never used and should be removed
SafeMath.tryAdd(uint256,uint256) (../chetan/gaza/mycontracts/DePayLaunchpadV1.sol#984-990) is never used and should be removed
SafeMath.tryDiv(uint256,uint256) (../chetan/gaza/mycontracts/DePayLaunchpadV1.sol#1026-1031) is never used and should be removed
SafeMath.tryMod(uint256,uint256) (../chetan/gaza/mycontracts/DePayLaunchpadV1.sol#1038-1043) is never used and should be removed
```

```
SafeMath.mod(uint256,uint256,string) (../chetan/gaza/mycontracts/DePayLaunchpadV1.sol#1179-1188) is never used and should be removed
SafeMath.sub(uint256,uint256,string) (../chetan/gaza/mycontracts/DePayLaunchpadV1.sol#1130-1139) is never used and should be removed
SafeMath.tryAdd(uint256,uint256) (../chetan/gaza/mycontracts/DePayLaunchpadV1.sol#984-990) is never used and should be removed
SafeMath.tryDiv(uint256,uint256) (../chetan/gaza/mycontracts/DePayLaunchpadV1.sol#1026-1031) is never used and should be removed
SafeMath.tryMod(uint256,uint256) (../chetan/gaza/mycontracts/DePayLaunchpadV1.sol#1038-1043) is never used and should be removed
SafeMath.tryMul(uint256,uint256) (../chetan/gaza/mycontracts/DePayLaunchpadV1.sol#1009-1019) is never used and should be removed
SafeMath.trySub(uint256,uint256) (../chetan/gaza/mycontracts/DePayLaunchpadV1.sol#997-1002) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
solc-0.8.0 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Low level call in Address.sendValue(address,uint256) (../chetan/gaza/mycontracts/DePayLaunchpadV1.sol#697-702):
        - (success) = recipient.call{value: amount}() (../chetan/gaza/mycontracts/DePayLaunchpadV1.sol#700)
Low level call in Address.functionCallWithValue(address,bytes,uint256,string) (../chetan/gaza/mycontracts/DePayLaunchpadV1.sol#765-776):
        - (success,returndata) = target.call{value: value}(data) (../chetan/gaza/mycontracts/DePayLaunchpadV1.sol#774)
Low level call in Address.functionStaticCall(address,bytes,string) (../chetan/gaza/mycontracts/DePayLaunchpadV1.sol#794-803):
        - (success,returndata) = target.staticcall(data) (../chetan/gaza/mycontracts/DePayLaunchpadV1.sol#801)
Low level call in Address.functionDelegateCall(address,bytes,string) (../chetan/gaza/mycontracts/DePayLaunchpadV1.sol#821-830):
        - (success,returndata) = target.delegatecall(data) (../chetan/gaza/mycontracts/DePayLaunchpadV1.sol#828)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
Parameter DePayLaunchpadV1.init(address,address,uint256,address)._launchedToken (../chetan/gaza/mycontracts/DePayLaunchpadV1.sol#1253) is
 not in mixedCase
Parameter DePayLaunchpadV1.init(address,address,uint256,address)._paymentToken (../chetan/gaza/mycontracts/DePayLaunchpadV1.sol#1254) is
not in mixedCase
Parameter DePayLaunchpadV1.init(address,address,uint256,address)._splitReleaseAmount (../chetan/gaza/mycontracts/DePayLaunchpadV1.sol#125
5) is not in mixedCase
Parameter DePayLaunchpadV1.init(address,address,uint256,address)._splitReleaseAddress (../chetan/gaza/mycontracts/DePayLaunchpadV1.sol#12
56) is not in mixedCase
Parameter DePayLaunchpadV1.start(uint256,uint256)._endTime (../chetan/gaza/mycontracts/DePayLaunchpadV1.sol#1289) is not in mixedCase
Parameter DePayLaunchpadV1.start(uint256,uint256)._price (../chetan/gaza/mycontracts/DePayLaunchpadV1.sol#1290) is not in mixedCase
Parameter DePayLaunchpadV1.whitelistAddress(address,bool)._address (../chetan/gaza/mycontracts/DePayLaunchpadV1.sol#1310) is not in mixed
Case
Parameter DePayLaunchpadV1.whitelistAddresses(address[],bool)._addresses (../chetan/gaza/mycontracts/DePayLaunchpadV1.sol#1319) is not in
 mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
```

```
INFO:Detectors:
renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (../chetan/gaza/mycontracts/DePayLaunchpadV1.sol#85-87)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (../chetan/gaza/mycontracts/DePayLaunchpadV1.sol#93-96)
name() should be declared external:
        - ERC20.name() (../chetan/gaza/mycontracts/DePayLaunchpadV1.sol#346-348)
symbol() should be declared external:
        - ERC20.symbol() (../chetan/gaza/mycontracts/DePayLaunchpadV1.sol#354-356)
decimals() should be declared external:
        - ERC20.decimals() (../chetan/gaza/mycontracts/DePayLaunchpadV1.sol#371-373)
totalSupply() should be declared external:
        - ERC20.totalSupply() (../chetan/gaza/mycontracts/DePayLaunchpadV1.sol#378-380)
balanceOf(address) should be declared external:
        - ERC20.balanceOf(address) (../chetan/gaza/mycontracts/DePayLaunchpadV1.sol#385-387)
transfer(address,uint256) should be declared external:
        - ERC20.transfer(address,uint256) (../chetan/gaza/mycontracts/DePayLaunchpadV1.sol#397-400)
allowance(address,address) should be declared external:
        - ERC20.allowance(address,address) (../chetan/gaza/mycontracts/DePayLaunchpadV1.sol#405-407)
approve(address,uint256) should be declared external:
        - ERC20.approve(address,uint256) (../chetan/gaza/mycontracts/DePayLaunchpadV1.sol#416-419)
transferFrom(address,address,uint256) should be declared external:
        - ERC20.transferFrom(address,address,uint256) (../chetan/gaza/mycontracts/DePayLaunchpadV1.sol#434-448)
increaseAllowance(address,uint256) should be declared external:
        - ERC20.increaseAllowance(address,uint256) (../chetan/gaza/mycontracts/DePayLaunchpadV1.sol#462-465)
decreaseAllowance(address,uint256) should be declared external:
        - ERC20.decreaseAllowance(address,uint256) (../chetan/gaza/mycontracts/DePayLaunchpadV1.sol#481-489)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
```

Solidity static analysis

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in Address.functionCallWithValue(address,bytes,uint256,string): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 765:4:

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in SafeERC20.safeApprove(contract IERC20,address,uint256): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 906:4:

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in SafeERC20.safeIncreaseAllowance(contract IERC20,address,uint256): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 921:4:

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in SafeERC20.safeDecreaseAllowance(contract IERC20,address,uint256): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 930:4:

Inline assembly:

The Contract uses inline assembly, this is only advised in rare cases.
Additionally static analysis modules do not parse inline Assembly, this can lead to wrong analysis results.
more
Pos: 675:8:

Inline assembly:

The Contract uses inline assembly, this is only advised in rare cases.
Additionally static analysis modules do not parse inline Assembly, this can lead to wrong analysis results.
more
Pos: 850:16:

Block timestamp:

Use of "block.timestamp": "block.timestamp" can be influenced by miners to a certain degree.
That means that a miner can "choose" the block.timestamp, to a certain degree, to change the outcome of a transaction in the mined block.
more
Pos: 1292:23:

Block timestamp:

Use of "block.timestamp": "block.timestamp" can be influenced by miners to a certain degree.
That means that a miner can "choose" the block.timestamp, to a certain degree, to change the outcome of a transaction in the mined block.
more
Pos: 1293:24:

Low level calls:

Use of "call": should be avoided whenever possible.
It can lead to unexpected behavior if return value is not handled properly.
Please use Direct Calls via specifying the called contract's interface.
more
Pos: 700:27:

Low level calls:

Use of "call": should be avoided whenever possible.
It can lead to unexpected behavior if return value is not handled properly.
Please use Direct Calls via specifying the called contract's interface.
more
Pos: 774:50:

Low level calls:

Use of "delegatecall": should be avoided whenever possible.
External code, that is called can change the state of the calling contract and send ether from the caller's balance.
If this is wanted behaviour, use the Solidity library feature if possible.
more
Pos: 828:50:

Gas costs:

Gas requirement of function ERC20.name is infinite:
If the gas requirement of a function is higher than the block gas limit, it cannot be executed.
Please avoid loops in your functions or actions that modify large areas of storage
(this includes clearing or copying arrays in storage)
Pos: 346:4:

Gas costs:

Gas requirement of function ERC20.symbol is infinite:
If the gas requirement of a function is higher than the block gas limit, it cannot be executed.
Please avoid loops in your functions or actions that modify large areas of storage
(this includes clearing or copying arrays in storage)
Pos: 354:4:

Gas costs:

Gas requirement of function ERC20.transfer is infinite:
If the gas requirement of a function is higher than the block gas limit, it cannot be executed.
Please avoid loops in your functions or actions that modify large areas of storage
(this includes clearing or copying arrays in storage)
Pos: 397:4:

Gas costs:

Gas requirement of function DePayLaunchpadV1.releaseUnclaimed is infinite:
If the gas requirement of a function is higher than the block gas limit, it cannot be executed.
Please avoid loops in your functions or actions that modify large areas of storage
(this includes clearing or copying arrays in storage)
Pos: 1416:2:

For loop over dynamic array:

Loops that do not have a fixed number of iterations, for example, loops that depend on storage values, have to be used carefully. Due to the block gas limit, transactions can only consume a certain amount of gas. The number of iterations in a loop can grow beyond the block gas limit which can cause the complete contract to be stalled at a certain point.
Additionally, using unbounded loops incurs in a lot of avoidable gas costs. Carefully test how many items at maximum you can pass to such functions to make it successful.
more
Pos: 1322:4:

For loop over dynamic array:

Loops that do not have a fixed number of iterations, for example, loops that depend on storage values, have to be used carefully. Due to the block gas limit, transactions can only consume a certain amount of gas. The number of iterations in a loop can grow beyond the block gas limit which can cause the complete contract to be stalled at a certain point.
Additionally, using unbounded loops incurs in a lot of avoidable gas costs. Carefully test how many items at maximum you can pass to such functions to make it successful.
more
Pos: 1403:4:

Constant/View/Pure functions:

IERC20.transfer(address,uint256) : Potentially should be constant/view/pure but is not. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 197:4:

Constant/View/Pure functions:

IERC20.approve(address,uint256) : Potentially should be constant/view/pure but is not. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 222:4:

Constant/View/Pure functions:

IERC20.transferFrom(address,address,uint256) : Potentially should be constant/view/pure but is not. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 233:4:

Constant/View/Pure functions:

ERC20._beforeTokenTransfer(address,address,uint256) : Potentially should be constant/view/pure but is not. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 615:4:

Constant/View/Pure functions:

ERC20._afterTokenTransfer(address,address,uint256) : Potentially should be constant/view/pure but is not. Note: Modifiers are currently not considered by this static analysis.
more
Pos: 635:4:

Similar variable names:

ERC20._burn(address,uint256) : Variables have very similar names "account" and "amount". Note: Modifiers are currently not considered by this static analysis.
Pos: 567:22:

Similar variable names:

ERC20._burn(address,uint256) : Variables have very similar names "account" and "amount". Note: Modifiers are currently not considered by this static analysis.
Pos: 567:50:

Similar variable names:

ERC20._burn(address,uint256) : Variables have very similar names "account" and "amount". Note: Modifiers are currently not considered by this static analysis.
Pos: 569:24:

Similar variable names:

ERC20._burn(address,uint256) : Variables have very similar names "account" and "amount". Note: Modifiers are currently not considered by this static analysis.
Pos: 571:22:

Similar variable names:

ERC20._burn(address,uint256) : Variables have very similar names "account" and "amount". Note: Modifiers are currently not considered by this static analysis.
Pos: 571:43:

Similar variable names:

ERC20._burn(address,uint256) : Variables have very similar names "account" and "amount". Note: Modifiers are currently not considered by this static analysis.
Pos: 573:28:

No return:

IERC20.totalSupply(): Defines a return type but never explicitly returns a value.
Pos: 183:4:

No return:

IERC20.balanceOf(address): Defines a return type but never explicitly returns a value.
Pos: 188:4:

No return:

IERC20.transfer(address,uint256): Defines a return type but never explicitly returns a value.
Pos: 197:4:

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 74:8:

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 94:8:

Guard conditions:

Use "assert(x)" if you never ever want x to be false, not in any circumstance (apart from a bug in your code). Use "require(x)" if x can be false, due to e.g. invalid input or a failing external component.
more
Pos: 157:8:

Data truncated:

Division of integer values yields an integer value again. That means e.g. 10 / 100 = 0 instead of 0.1 since the result is an integer again. This does not hold for division of (only) literal values since those yield rational constants.
Pos: 1016:16:

Data truncated:

Division of integer values yields an integer value again. That means e.g. 10 / 100 = 0 instead of 0.1 since the result is an integer again. This does not hold for division of (only) literal values since those yield rational constants.
Pos: 1029:26:

Data truncated:

Division of integer values yields an integer value again. That means e.g. 10 / 100 = 0 instead of 0.1 since the result is an integer again. This does not hold for division of (only) literal values since those yield rational constants.
Pos: 1098:15:

Solhint Linter

```
contracts/DePayLaunchpadV1.sol:443:18: Error: Parse error: missing ';'
at '{'

contracts/DePayLaunchpadV1.sol:484:18: Error: Parse error: missing ';'
at '{'

contracts/DePayLaunchpadV1.sol:517:18: Error: Parse error: missing ';'
at '{'

contracts/DePayLaunchpadV1.sol:566:18: Error: Parse error: missing ';'
at '{'

contracts/DePayLaunchpadV1.sol:935:18: Error: Parse error: missing ';'
at '{'

contracts/DePayLaunchpadV1.sol:985:18: Error: Parse error: missing ';'
at '{'

contracts/DePayLaunchpadV1.sol:998:18: Error: Parse error: missing ';'
at '{'

contracts/DePayLaunchpadV1.sol:1010:18: Error: Parse error: missing ';'
at '{'

contracts/DePayLaunchpadV1.sol:1027:18: Error: Parse error: missing ';'
at '{'

contracts/DePayLaunchpadV1.sol:1039:18: Error: Parse error: missing ';'
at '{'

contracts/DePayLaunchpadV1.sol:1135:18: Error: Parse error: missing ';'
at '{'

contracts/DePayLaunchpadV1.sol:1158:18: Error: Parse error: missing ';'
at '{'

contracts/DePayLaunchpadV1.sol:1184:18: Error: Parse error: missing ';'
at '{'
```

Email: audit@EtherAuthority.io