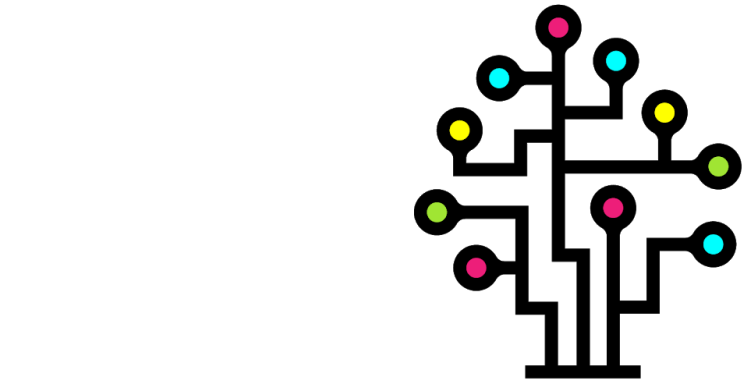


BLOCKLOGY®

LEARN LOOP LEAP

TIMEALLY SMART CONTRACT REPORT

Reported to - Eraswap Technologies

Contract name - TimeAlly.sol
(https://github.com/KMPARDS/timeally-contract/blob/75percentTimeLoan/contracts/TimeAlly.sol)

Website name - https://www.timeally.io

Date - 27.07.2019

Prepared By
Blocklogy Edutech

Contracts Audited:

1- TimeAlly.sol

2- NRTManager.sol

Website Audited:

https://www.timeally.io

Auditing Overview:

1- Overview

2- Methods Used for Auditing

3- Security Level

4- Findings: Errors, Re-entrancy Attack, Syntax Attack, Gas Estimation

5- Analysis of Security Issues

6- Automated Test Result

7- Test Report

8- Summary of Smart Contract Audit

9- TimeAlly VAPT Assessment Report

10- TimeAlly Software Testing Report

OVERVIEW:

This Audit Report highlights the overall security of TimeAlly Smart Contract where Blocklogy Edutech performed a security review of the TimeAlly Smart contract and its working according to the Era Swap Whitepaper (https://eraswaptoken.io/pdf/eraswap_whitepaper.pdf) as per the request of Eraswap Technologies.

Contract Location:

https://github.com/KMPARDS/timeally-contract/tree/75percentTimeLoan/contracts

METHODS USED FOR AUDITING:

Following steps has been performed for the Audit:

1. Understanding the contracts intended purpose by reading the whitepaper provided by Eraswap Technologies.
2. Automated scanning of the contract for security vulnerabilities and use of best practice guidelines, Manual each line analysis of source code for security vulnerabilities such as: re-entrancy analysis, race condition analysis, storage layout vulnerabilities, DOS attack, Function visibility issue, under-overflow issue, timestamp dependencies, front running issues and transaction order dependencies.

SECURITY LEVEL:

1. Level 1 : Issues are minor and warnings that can remain unfixed.
2. Level 2 : Issues could bring problem and should eventually be fixed.
3. Level 3 : Issues will bring problem and should be fixed before deployment.

FINDINGS:

Level 1:

Extra Gas Consumption -

TimeAlly.sol - Lines: 560-574, Lines: 354-368, Lines: 445-462, Lines: 262-265, Lines: 581-604, Lines: 477-510, Lines: 505-508, Lines: 421-431, Lines: 382-397, Lines: 595-599.

Issue:

State variable, .balance, or .length is used in the condition of for or while loop. In this case, every iteration of loop consumes extra gas.

Recommendation :

If state variable, .balance, or .length is used several times, holding its value in a local variable is more gas efficient.

Level 2:

Costly loop:

Lines: 354-368, Lines: 560-574, Lines: 581-604, Lines: 382-397, Lines: 262-265, Lines: 595-599, Lines: 477-510, Lines: 445-462, Lines: 421-431.

Issue:

Ethereum is a very resource-constrained environment. Prices per computational step are orders of magnitude higher than with centralized providers. Moreover, Ethereum miners impose a limit on the total number of gas consumed in a block. If array.length is large enough, the function exceeds the block gas limit, and transactions calling it will never be confirmed:

```
for (uint256 i = 0; i < array.length ; i++) {
    cosltyFunc();
}
```

This becomes a security issue, if an external actor influences array.length. E.g., if array enumerates all registered addresses, an adversary can register many addresses, causing the problem described above.

Recommendation:

Avoid loops with big or unknown number of steps.

Level 3:

There are no Level 3 severity issue in the TimeAlly Smart Contract.

ANALYSIS OF SECURITY ISSUES:

Level 1 and Level 2 issues for TimeAlly contract can be just considered as a warning, as per Era Swap Whitepaper the loops and gas consumption are common with respect to its working.

Since there are no Level 3 issue where issues such as re-entrancy, functional visibility, race condition, DOS attack and storage layout vulnerability are being checked manually and issue for the same is not found in the TimeAlly Smart Contract and also no vulnerability found in NRTManager Contract.

AUTOMATED TEST RESULT:

Mocha Script Test:

What is Mocha Test ?

Mocha is a feature-rich JavaScript test framework running on Node.js and in the browser, making asynchronous testing *simple* and *fun*. Mocha tests run serially, allowing for flexible and accurate reporting, while mapping uncaught exceptions to the correct test cases.

Mocha Test on TimeAlly.sol :

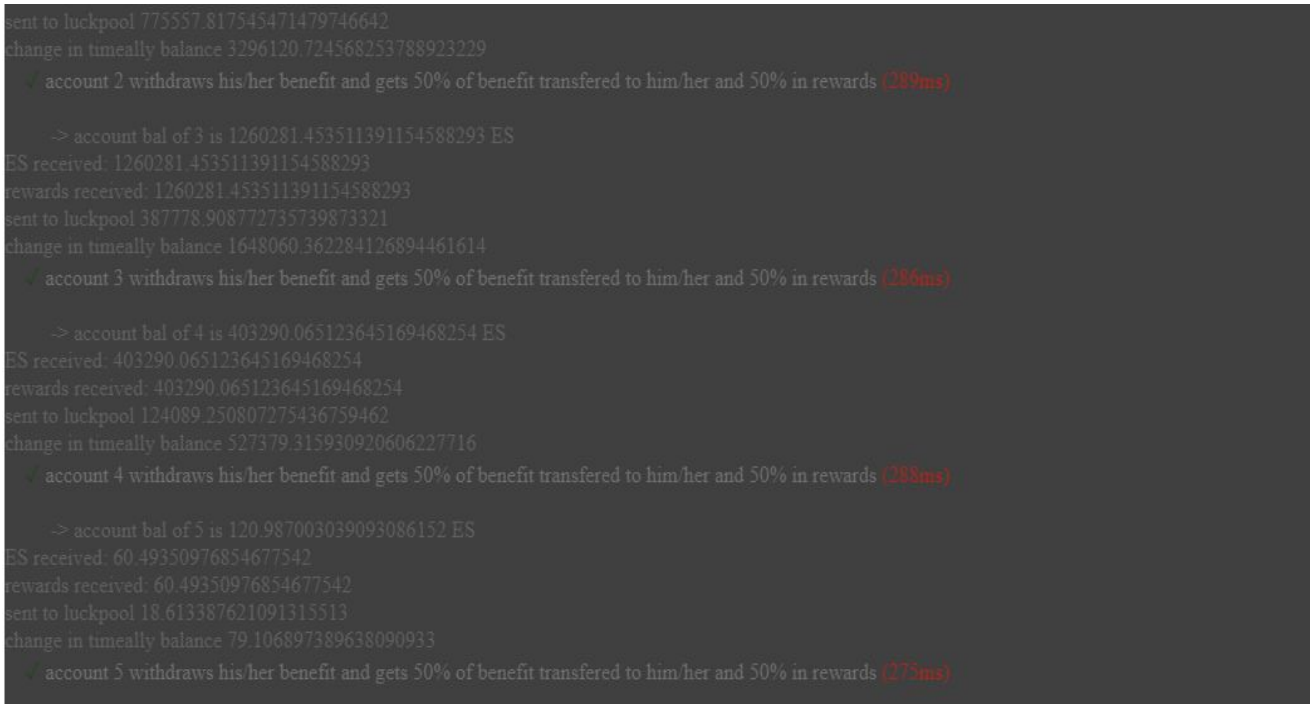
The Business logic for TimeAlly smart contract was written as Mocha test file and performed an Automated testing of the TimeAlly logic with respect to Whitepaper provided by Era Swap Technologies.

```
Sohams-MacBook-Air:timeAlly-new sohamzemse$ npm test

> timeally-new@1.0.0 test /Users/sohamzemse/soham/kmpards/timeAlly-new
> mocha --timeout 30000



  Ganache Setup
    ✓ initiates ganache and generates a bunch of demo accounts (85ms)

  Era Swap Setup
    ✓ deploys Era Swap token contract from first account (363ms)
    ✓ gives account 0 => 91,00,00,000 ES balance (69ms)
    ✓ mou() time machine is present

  NRT Manager Setup
    ✓ deploys NRT manager from the account 0 (363ms)
    ✓ invokes AddNRTManager method in the Era Swap Instance from account 0 (203ms)

  TimeAlly Setup
Gas used for TimeAlly deploy 6547838
    ✓ deploys TimeAlly from the account 0 (175ms)
    ✓ invokes Update Addresses in NRT Manager from account 0 (165ms)
    ✓ creating staking plans (with loan feature inactive) of 1 year (with benefit 13 fractionFrom15) and 2 year (15 benefit fractionFrom15) (283ms)
    ✓ creating staking plans (with loan feature active) of 1 year (with benefit 13 fractionFrom15) and 2 year (15 benefit fractionFrom15) (276ms)
```

```
  Staking
    ✓ account 0 sends 50000000 ES, 500000000 ES, 250000000 ES, 80000000 ES, 12000 ES to 1, 2, 3, 4, 5 account
31177
    ✓ account 1 gives allowance of 50000000 ES to timeAlly (714ms)
675456
    ✓ account 1 stakes 50000000 ES using timeAlly 2 year (601ms)
31241
    ✓ account 2 gives allowance of 500000000 ES to timeAlly (97ms)
219084
    ✓ account 2 stakes 500000000 ES using timeAlly 1 year (402ms)
31177
    ✓ account 3 gives allowance of 250000000 ES to timeAlly (101ms)
234084
    ✓ account 3 stakes 250000000 ES using timeAlly 2 year (403ms)
31177
    ✓ account 4 gives allowance of 80000000 ES to timeAlly (97ms)
219020
    ✓ account 4 stakes 80000000 ES using timeAlly 1 year (387ms)
    ✓ goes 10 days in future (58ms)
31177
    ✓ account 5 gives allowance of 12000 ES to timeAlly (105ms)
234084
    ✓ account 5 stakes 12000 ES using timeAlly 2 year (400ms)
    ✓ goes 10 days back in time (66ms)
```

```
 first month in TimeAlly
   ✓ time travelling to the future by 1 month and half day using mou() time machine (97ms)
timeAllyMonthlyNRTthisMonth 10237500.0
   ✓ invoking MonthlyNRTRelease in NRT contract and checking if TimeAlly gets 10237500 ES in first month nrt (676ms)
seeBenefitOfAStakingByMonths of account 1 and months: [1] 581668.204524483757039676
   ✓ account 1 tries to see benefit of staking id 0 using seeBenefitOfAStakingByMonths (43ms)
benefit BigNumber { _hex: '0x00' }
   ✓ account 5 tries to see his/her benefit but gets 0.0 ES as he/she staked 10 days later (48ms)
   ✓ account 5 tries to withdraw his/her benefit gets 0.0 ES as he/she staked 10 days later (207ms)
   ✓ goes 10 days future in time (51ms)
account 5 benefit 120.986986541092621463 ES
   ✓ account 5 tries to see his/her benefit, now should see something (53ms)
liquid transfered to account 5: 60.493493270546310732
NRT received: 18.613382544783480225 ES
   ✓ account 5 tries to withdraw his/her benefit should get some balance now (231ms)

 second month in TimeAlly
   ✓ time travelling to the future by 1 month and half day using mou() time machine (86ms)
```

```
imeAllyMonthlyNRTthisMonth 10237502.792007381717522033
   ✓ invoking MonthlyNRTRelease in NRT contract and checking if TimeAlly gets some ES in second month nrt (482ms)

      -> benefit of account 1 is 581668.363159103609809981 ES
   ✓ account 1 can see his/her benefit (to be shown in UI) (54ms)

      -> benefit of account 2 is 5041125.814045564618353174 ES
   ✓ account 2 can see his/her benefit (to be shown in UI) (49ms)

      -> benefit of account 3 is 2520562.907022782309176586 ES
   ✓ account 3 can see his/her benefit (to be shown in UI) (56ms)

      -> benefit of account 4 is 806580.130247290338936507 ES
   ✓ account 4 can see his/her benefit (to be shown in UI) (56ms)

      -> benefit of account 5 is 120.98701953709355084 ES
   ✓ account 5 can see his/her benefit (to be shown in UI) (53ms)

      -> account bal of 1 is 290834.18157955180490499 ES
ES received: 290834.18157955180490499
ewards received: 290834.18157955180490499
ent to luckpool 0.0
hange in timeally balance 290834.18157955180490499
   ✓ account 1 withdraws his/her benefit and gets 50% of benefit transfered to him/her and 50% in rewards (373ms)

      -> account bal of 2 is 2520562.907022782309176587 ES
ES received: 2520562.907022782309176587
ewards received: 2520562.907022782309176587
```


sent to luckpool 775557.8175454714797466 42
change in timeally balance 3296120.724568253788923229
✓ account 2 withdraws his/her benefit and gets 50% of benefit transfered to him/her and 50% in rewards (289ms)
-> account bal of 3 is 1260281.453511391154588293 ES
ES received: 1260281.453511391154588293
rewards received: 1260281.453511391154588293
sent to luckpool 387778.908772735739873321
change in timeally balance 1648060.362284126894461614
✓ account 3 withdraws his/her benefit and gets 50% of benefit transfered to him/her and 50% in rewards (286ms)
-> account bal of 4 is 403290.065123645169468254 ES
ES received: 403290.065123645169468254
rewards received: 403290.065123645169468254
sent to luckpool 124089.250807275436759462
change in timeally balance 527379.315930920606227716
✓ account 4 withdraws his/her benefit and gets 50% of benefit transfered to him/her and 50% in rewards (288ms)
-> account bal of 5 is 120.987003039093086152 ES
ES received: 60.49350976854677542
rewards received: 60.49350976854677542
sent to luckpool 18.613387621091315513
change in timeally balance 79.106897389638090933
✓ account 5 withdraws his/her benefit and gets 50% of benefit transfered to him/her and 50% in rewards (275ms)


timeAllyMonthlyNRTthisMonth 10430616.68857696556215424
✓ invoking MonthlyNRTRelease in NRT contract and checking if TimeAlly gets some ES in third month nrt (453ms)
-> benefit of account 1 is 592640.594024681797643341 ES
✓ account 1 can see his/her benefit (to be shown in UI) (52ms)
-> benefit of account 2 is 5136218.48154724224624229 ES
✓ account 2 can see his/her benefit (to be shown in UI) (54ms)
-> benefit of account 3 is 2568109.240773621123121145 ES
✓ account 3 can see his/her benefit (to be shown in UI) (47ms)
-> benefit of account 4 is 821794.957047558759398765 ES
✓ account 4 can see his/her benefit (to be shown in UI) (53ms)
-> benefit of account 5 is 123.269243557133813909 ES
✓ account 5 can see his/her benefit (to be shown in UI) (45ms)
-> account bal of 1 is 587154.47859189270372666 ES
ES received: 296320.29701234089882167
rewards received: 296320.29701234089882167
sent to luckpool 0.0
change in timeally balance 296320.29701234089882167
✓ account 1 withdraws his/her benefit and gets 50% of benefit transfered to him/her and 50% in rewards (288ms)

```
      -> account bal of 2 is 5088672.147796403432297732 ES
ES received: 2568109.240773621123121145
rewards received: 2568109.240773621123121145
sent to luckpool 790187.458699575730191121
change in timeally balance 3358296.699473196853312266
  ✓ account 2 withdraws his/her benefit and gets 50% of benefit transfered to him/her and 50% in rewards (283ms)

      -> account bal of 3 is 2544336.073898201716148866 ES
ES received: 1284054.620386810561560573
rewards received: 1284054.620386810561560573
sent to luckpool 395093.72934978786509556
change in timeally balance 1679148.349736598426656133
  ✓ account 3 withdraws his/her benefit and gets 50% of benefit transfered to him/her and 50% in rewards (284ms)

      -> account bal of 4 is 814187.543647424549167637 ES
ES received: 410897.478523779379699383
rewards received: 410897.478523779379699383
sent to luckpool 126429.993391932116830579
change in timeally balance 537327.471915711496529962
  ✓ account 4 withdraws his/her benefit and gets 50% of benefit transfered to him/her and 50% in rewards (274ms)

      -> account bal of 5 is 182.621624817659993107 ES
ES received: 61.634621778566906955
rewards received: 61.634621778566906955
sent to luckpool 18.964499008789817524
change in timeally balance 80.599120787356724479
  ✓ account 5 withdraws his/her benefit and gets 50% of benefit transfered to him/her and 50% in rewards (277ms)
```

```
Loan
  ✓ creating a loan plan of 1% (85ms)

    max amount can be loaned 0.0 ES
  ✓ account 1 tries to see how much loan he/she can take from his loan inavtive staking sees 0 ES as max loan possible (43ms)

    max amount can be loaned 125000000.0 ES
  ✓ account 3 tries to see how much loan he/she can take from his loan active staking sees something as max loan possible

    account 3 got credited: 3960.0 ES
  ✓ account 3 can take loan of 4000 ES (355ms)
  ✓ account 3 tries to repay loan by giving allowance of 4000 ES and invoking repay loan function with loan id 0 (287ms)

    max amount can be loaned 6000.0 ES
  ✓ account 5 tries to see how much loan he/she can take from his loan active staking sees something as max loan possible (41ms)
  ✓ account 5 tries to take loan of 6001 ES gets error (178ms)

    account 5 got credited: 5940.0 ES
  ✓ account 5 can take loan of 6000 ES (339ms)
  ✓ time travelling to the future by little more than 2 months using mou() time machine && invoking monthly NRT (993ms)
VM Exception while processing transaction: revert
```

```
  Nominee
    ✓ Account 2 can make Account 3 and Account 4 nominee of his staking id 0 (140ms)
    ✓ time travelling to the future by 1 year using mou() time machine && invoking monthly NRT every month (4885ms)
    ✓ nominee account 3 (of account 2's staking id 0) should get error while calling withdraw function (50ms)
    ✓ time travelling to the future by 1 year using mou() time machine && invoking monthly NRT every month (4917ms)
    ✓ time travelling to the future by 1 year using mou() time machine && invoking monthly NRT every month (4831ms)
252544336.073898201716148866
    ✓ nominee account 3 (of account 2's staking id 0) should get balance in liquid and reward while calling withdraw function (199ms)


  73 passing (27s)
  2 failing

  1) first month in TimeAlly
       account 5 tries to see his/her benefit but gets error as he/she staked 10 days later:

      AssertionError [ERR_ASSERTION]: should get error
      + expected - actual

      -false
      +true

      at Context.it (test/timeAlly.test.js:303:16)
      at <anonymous>
      at process._tickCallback (internal/process/next_tick.js:188:7)

  2) first month in TimeAlly
       account 5 tries to withdraw his/her benefit get error as he/she staked 10 days later:

      AssertionError [ERR_ASSERTION]: should get error
      + expected - actual

      -false
      +true

      at Context.it (test/timeAlly.test.js:320:9)
      at <anonymous>
      at process._tickCallback (internal/process/next_tick.js:188:7)
```

TEST REPORT:

Manual Testing :

Manual testing of TimeAlly contract was performed line by line checking the code and manually deploying the contract after testing the Level 1 and Level 2 severity issues were found and according to the Whitepaper's TimeAlly working the severity can be ignored as a warning since level 3 issues are not found the manual testing shows no sign of attack or vulnerabilities.

Automated Testing:

Automated testing was performed in Mocha by writing the logic of the TimeAlly with respect to Whitepaper provided by Eraswap Technologies, The testing performed was passed for each case that were written making sure that the contract is functional without any error.

SUMMARY OF AUDIT:

The contract have implemented the best security practices, The contract stores the funds safely and transact safely whenever needed. The contracts are written keeping in mind the best security practices of the solidity and it is using the pull mechanism of the funds which is the best way to avoid any attacks and the misuse of the funds by the attackers. The contract is also using the SafeMath library of the OpenZeppelin which avoids the underflow and overflow.

The Contract is free from all Security Vulnerability and external attacks, the contract is Good to Deploy over mainnet.

TIMEALLY VAPT ASSESSMENT REPORT

VAPT Performed On - https://www.timeally.io

VAPT Testing Overview:

1. Introduction
2. Purpose
3. Scope of Assessment
4. Limitations and Constraints
5. Information Security Assessment Approach
6. Information Collection Techniques

Overview:

1. Introduction:

This section presents the purpose,scope,and limitations of the vulnerability assessment.

2. Purpose:

The purpose of this security review is to identify the vulnerabilities on the publicly accessible servers within their operational environment.

3. Scope of Assessment

The scope of this information security vulnerability assessment is limited technical controls(system and application access control,identification and authentication, etc.) implemented on the two servers identified in the report.

4. Limitations and Constraints

The ability to collect necessary information and understanding of the some of the required operating system functionality on the servers. Therefore, some of the findings and recommendations may be inappropriate due to the assumptions made by the assessment team during the analysis of the raw data.

5. Information Security Assessment Approach

This Section describes the methodology employed by the Security Engineer during the conduct of this vulnerability assessment.

6. Information Collection Techniques

The VAPT Assessment Team used the following information collection techniques to gain an understanding of the database servers and identify vulnerabilities:

- OWASP-ZAP
- NMAP
- Manual inspection

SECURITY LEVEL:

1. High: If an observation or finding is evaluated as high, there is a strong need for corrective measures.

2. Medium: If an observation is rated as medium, corrective actions are needed and a plan must be developed to incorporate these actions within a reasonable period of time.

3. Low: If an observation is described as low,the system's authorizing official must determine whether corrective actions are still required or decide to accept the vulnerability.

SECURITY INFORMATION:

Vulnerabilities Found when performing VAPT

1. CSP Scanner: Wildcard Directive (2)
2. X-Frame-Options Header Not Set (1)
3. Web Browser XSS Protection Not Enabled (3)
4. X-Content-Type-Options Header Missing (6)

CSP Scanner : Wildcard Directive

Url: http://www.timeally.io/robots.txt
Risk: Medium
Confidence: Medium
Parameter: Content-Security-Policy
Attack:
Evidence: default-src 'none'
CWE ID: 16
WASC-ID: 15

Description:

The following directives either allow wildcard sources (or ancestors), are not defined, or are overly broadly defined:
Frame-ancestor

Solution:

The above mentioned Vulnerability found while performing VAPT is solved by Eraswap team by configuring the web server, application server, and load balancer and the website is safe and vulnerable free.

CSP Scanner : Wildcard Directive

Url: http://www.timeally.io/sitemap.xml
Risk: Medium
Confidence: Medium
Parameter: Content-Security-Policy
Attack:
Evidence: default-src 'none'
CWE ID: 16
WASC-ID: 15

Description:

The following directives either allow wildcard sources (or ancestors), are not defined, or are overly broadly defined:
frame-ancestor

Solution:

The above mentioned Vulnerability found while performing VAPT is solved by Eraswap team by configuring the web server, application server, and load balancer and the website is safe and vulnerable free.

X-Frame-Options Header Not Set

Url: http://www.timeally.io/
Risk: Medium
Confidence: Medium
Parameter: X-Frame-Options
Attack:
Evidence:
CWE ID: 16
WASC-ID: 15

Description:

X-Frame-Options header is not included in the HTTP response to protect against 'ClickJacking' attacks.

Solution:

The above mentioned Vulnerability found while performing VAPT is solved by Eraswap team by setting X-Frame-Options HTTP header by SAMEORIGIN.

Web Browser XSS Protection Not Enabled

Url: http://www.timeally.io/

Risk: Low

Confidence: Medium

Parameter: X-XSS-Protection

Attack:

Evidence:

CWE ID: 933

WASC-ID: 14

Description:

Web Browser XSS Protection is not enabled, or is disabled by the configuration of the 'X-XSS-Protection' HTTP response header on the web server.

Solution:

The above mentioned Vulnerability found while performing VAPT is solved by Eraswap team by enabling web browser's XSS filter and configuring the X-XSS-Protection HTTP response header to '1'.

Web Browser XSS Protection Not Enabled

Url: http://www.timeally.io/robots.txt
Risk: Low
Confidence: Medium
Parameter: X-XSS-Protection
Attack:
Evidence:
CWE ID: 933
WASC-ID: 14

Description:

Web Browser XSS Protection is not enabled, or is disabled by the configuration of the 'X-XSS-Protection' HTTP response header on the web server.

Solution:

The above mentioned Vulnerability found while performing VAPT is solved by Eraswap team by enabling web browser's XSS filter and configuring the X-XSS-Protection HTTP response header to '1'.

Web Browser XSS Protection Not Enabled

Url: http://www.timeally.io/sitemap.xml
Risk: Low
Confidence: Medium
Parameter: X-XSS-Protection
Attack:
Evidence:
CWE ID: 933
WASC-ID: 14

Description:

Web Browser XSS Protection is not enabled, or is disabled by the configuration of the 'X-XSS-Protection' HTTP response header on the web server.

Solution:

The above mentioned Vulnerability found while performing VAPT is solved by Eraswap team by enabling web browser's XSS filter and configuring the X-XSS-Protection HTTP response header to '1'.

X-Content-Type-Options Header Missing

Url: http://www.timeally.io/
Risk: Low
Confidence: Medium
Parameter: X-Content-Type-Options
Attack:
Evidence:
CWE ID: 16
WASC-ID: 15

Description:

The Anti-MIME-Sniffing header X-Content-Type-Options was not set to 'nosniff'. This allows older versions of Internet Explorer and Chrome to perform MIME-sniffing on the response body, potentially causing the response body to be interpreted and displayed as a content type other than the declared content type.

Solution:

The above mentioned Vulnerability found while performing VAPT is solved by Eraswap team by configuring X-Content-Type-Options header to 'nosniff

X-Content-Type-Options Header Missing

Url: http://www.timeally.io/favicon.ico
Risk: Low
Confidence: Medium
Parameter: X-Content-Type-Options
Attack:
Evidence:
CWE ID: 16
WASC-ID: 15

Description:

The Anti-MIME-Sniffing header X-Content-Type-Options was not set to 'nosniff'. This allows older versions of Internet Explorer and Chrome to perform MIME-sniffing on the response body, potentially causing the response body to be interpreted and displayed as a content type other than the declared content type.

Solution:

The above mentioned Vulnerability found while performing VAPT is solved by Eraswap team by configuring X-Content-Type-Options header to 'nosniff''

X-Content-Type-Options Header Missing

Url: http://www.timeally.io/manifest.json
Risk: Low
Confidence: Medium
Parameter: X-Content-Type-Options
Attack:
Evidence:
CWE ID: 16
WASC-ID: 15

Description:

The Anti-MIME-Sniffing header X-Content-Type-Options was not set to 'nosniff'. This allows older versions of Internet Explorer and Chrome to perform MIME-sniffing on the response body, potentially causing the response body to be interpreted and displayed as a content type other than the declared content type..

Solution:

The above mentioned Vulnerability found while performing VAPT is solved by Eraswap team by configuring X-Content-Type-Options header to 'nosniff'

X-Content-Type-Options Header Missing

Url: http://www.timeally.io/static/js/0.chunk.js

Risk: Low

Confidence: Medium

Parameter: X-Content-Type-Options

Attack:

Evidence:

CWE ID: 16

WASC-ID: 15

Description:

The Anti-MIME-Sniffing header X-Content-Type-Options was not set to 'nosniff'. This allows older versions of Internet Explorer and Chrome to perform MIME-sniffing on the response body, potentially causing the response body to be interpreted and displayed as a content type other than the declared content type.

Solution:

The above mentioned Vulnerability found while performing VAPT is solved by Eraswap team by configuring X-Content-Type-Options header to 'nosniff'

X-Content-Type-Options Header Missing

Url: http://www.timeally.io/static/js/bundle.js
Risk: Low
Confidence: Medium
Parameter: X-Content-Type-Options
Attack:
Evidence:
CWE ID: 16
WASC-ID: 15

Description:

The Anti-MIME-Sniffing header X-Content-Type-Options was not set to 'nosniff'. This allows older versions of Internet Explorer and Chrome to perform MIME-sniffing on the response body, potentially causing the response body to be interpreted and displayed as a content type other than the declared content type.

Solution:

The above mentioned Vulnerability found while performing VAPT is solved by Eraswap team by configuring X-Content-Type-Options header to 'nosniff'

X-Content-Type-Options Header Missing

Url: http://www.timeally.io/static/js/main.chunk.js
Risk: Low
Confidence: Medium
Parameter: X-Content-Type-Options
Attack:
Evidence:
CWE ID: 16
WASC-ID: 15

Description:

The Anti-MIME-Sniffing header X-Content-Type-Options was not set to 'nosniff'. This allows older versions of Internet Explorer and Chrome to perform MIME-sniffing on the response body, potentially causing the response body to be interpreted and displayed as a content type other than the declared content type.

Solution:

The above mentioned Vulnerability found while performing VAPT is solved by Eraswap team by configuring X-Content-Type-Options header to 'nosniff'.

VAPT testing Conclusion:

Penetration tests offer unparalleled insight into an organization’s security effectiveness as well as a road map for enhancing security. to simulate a cyber vulnerabilities that can be identified and corrected before they are exploited. The website is vulnerable free and **Good to go**.

TIMEALLY SOFTWARE TESTING REPORT

Software Testing Report On - https://www.timeally.io

Software Testing Overview:

1. Purpose
2. Overview
3. Testing Scope
4. Metrics
5. Types of testing performed
6. Best Practices
7. Practices.
8. Exit Criteria
9. Conclusion/Sign Off

1.Purpose

This Testing Report explains the various activities performed as part of Testing on 'TimeAlly' Website on Testnet.

2.Overview

TimeAlly is a Smart Contract controls volatility, Reward Distribution from ES NRT and Simply increase TA Holder's Era Swap (ES) counts based on their vesting period without getting into the hassle of day trading.

3. Testing Scope

Functional Testing for the following modules are in Scope of Testing

- Create Wallet
- Load wallet with Keystore file, Private Key, Mnemonic and Connect to Metamask
- Staking
- Apply for Loan
- Vesting Rewards
- Transaction
- Create Nominee under Staking

4. Metrics

- No. of test cases planned vs executed :24
  Severity of test cases planned vs executed :
  1. High : 00
  2. Medium : 16
  3. Low : 08

- No. of test cases passed : 22
  Severity of test cases passed :
  1. High : 00
  2. Medium : 16
  3. Low : 06
- No. of test cases failed: 02
  Severity of test cases failed :
  4. High : 00
  5. Medium : 00
  6. Low : 02

5. Types of testing performed

a) Smoke Testing:

This testing was done whenever a Build is received (deployed into Test environment) for Testing to make sure the major functionalities are working fine, Build can be accepted and Testing can start.

b) System Integration Testing:

This is the Testing performed on the Application under test, to verify the entire website works as per the requirements. Critical Business scenarios were tested to make sure important functionalities in the website works as intended without any errors.

c) Regression Testing:

Regression testing was performed each time a new build is deployed for testing which contains defect fixes and new enhancements, if any. Regression Testing is being done on the entire application and not just the new functionalities and Defect fixes. This testing ensures that existing functionalities works fine after defect fix and new enhancements are added to the existing application. Test cases for new functionalities are added to the existing test cases and executed.

d)Positive testing:

Positive testing is the type of testing that can be performed on the system by providing the valid data as input. It checks whether an application behaves as expected with positive inputs. This test is done to check the application that does what it is supposed to do.

e)Negative Testing:

Negative Testing is a variant of testing that can be performed on the system by providing invalid data as input. It checks whether an application behaves as expected with the negative inputs. This is to test the application does not do anything that it is not supposed to do so.

6. Best Practices

There will be a lot of activities done by the Testing team during the project. Some of them could have saved time, some proved to be a good & efficient way to work, etc.

Example: A repetitive task done manually every time was time consuming.This task was automated by creating scripts and run each time, which saved time and resources.Smoke test cases were automated and the scripts were run, which ran fast and saved time. Automation scripts were prepared where lot of records need to be created for Testing.

7. Exit Criteria

- All test cases should be executed – Yes
- All defects in High,Low,Medium severity should be verified and closed – Yes.
- Any open defects in trivial severity – Action plan prepared with expected dates of closure.

8. Conclusion/Sign Off

As the Exit criteria was met and satisfied this website is suggested to **'Go Live'** by the Testing team.