LUNARAY
BLOCKCHAINSECURITY

CODE SECURITY AUDIT REPORT

For RealPerp

25 October 2023

lunaray.co

1. Table of Contents

1. Table of Contents 2
2. Overview 3
3. Background 4
   3.1 Project Description 4
   3.2 Audit Range 5
4. High Risk Vulnerabilities 6
   4.1 Unauthorized information access 6
   4.2 All Controller has CORS vulnerabilities 7
   4.3 arbitrary affiliation update 10
   4.4 Input unfiltered leading code injection 13
5. Medium Risk vulnerabilities 14
   5.1 Full controller has CORS vulnerability 14
6. Low Risk Vulnerabilities 16
   6.1 log injection 16
   6.2 Input Not Validated 17
7. Information Alerts 19
   7.1 Job usage issues 19
   7.2 excessive privileges 20
8. Audit details 25
   8.1 Findings Summary 25
   8.2 Risk distribution 26

2. Overview

On Oct 19, 2023, the security team of Lunaray PTE LTD received the security audit request of the **REALPERP project**. The team completed the audit of the **REALPERP project** on Oct 25, 2023. During the audit process, the security audit experts of Lunaray PTE LTD and the REALPERP project interface Personnel communicate and maintain symmetry of information, conduct security audits under controllable operational risks, and avoid risks to project generation and operations during the testing process.

Through communicat and feedback with REALPERP project party, it is confirmed that the loopholes and risks found in the audit process have been repaired or within the acceptable range. The result of this REALPERP code security audit: **Passed**

Audit Report Hash:
8C8C038B4E4D63CD60A698FA438CFA20F0BD37FDBDC04F472A2D342010820E31

3. Background

3.1 Project Description

Project name	RealPerp
Contract type	Decentralized exchange
Code language	nodejs
Public chain	Manta Pacific
Project website	http://realperp.com
Introduction	Realperp is a decentralized swap and perpetual DEX on Manta Pacific, offering diverse trading options and high liquidity for blue chip crypto assets. It's designed for traders prioritizing capital control and superior trading experiences. With low swap fees and minimal price impact trades, Realperp enhances its efficient trading environment through unique multi-asset pools that facilitate trading and generate earnings for liquidity providers from market making, swap fees, and leverage trading.
Code Repository	RealPerp/realperp-backend RealPerp/realperp-token-price

3.2 Audit Range

repository name and corresponding commit SHA：

Repository Name	commit SHA
realperp-backend	e1fc78e2603efa7a0c5259d809ab4694a6c99f80
realperp-token-price	2e60667299f92c240c67d6778bfb35edc1e83218

4. High Risk Vulnerabilities

4.1 Unauthorized information access

- **Risk description**

According to the code src\query\query.controller.ts getTotalUsers method, found that the call queryService.getTotalUsers

```
@Get('total_users')
async getTotalUsers(@Res() res) {
  const totalUsers = await this.queryService.getTotalUsers();
  return res.status(HttpStatus.OK).json({ users: totalUsers });
}
```

Continuing to follow the code, I found that getTotalUsers in src\query\query.service.ts does not require permission checking.

```
 /**

  * get total active users count from cache or SubGraph
  */
async getTotalUsers() {
  try {
    const key = 'total.users';
    const cachedTotalUsers = await this.cacheManager.get(key);
    if (cachedTotalUsers) {
      return cachedTotalUsers;
    } else {
      const response: UserStats = await this.client.request(gql`
        {
          userStats(where: { period: total }) {
            uniqueCountCumulative
          }
        }
      `);
      const totalUsers =
response.userStats[0].uniqueCountCumulative;
      await this.cacheManager.set(key, totalUsers, {
        ttl: GRAPH_DATA_CACHE_TIME_TO_LIVE,
      } as any);
      return totalUsers;
    }
```

```
  } catch (e) {
    Logger.error(e);
    return 0;
  }
}
```

This in turn leads directly to leaking all user's information.

- **Security Advice**

Add permission checks to this controller

- **Repair Status**

Resolved.

4.2 All Controller has CORS vulnerabilities

- **Risk description**

The vulnerability occurs in controllers such as src\query\query.controller.ts, src\health\health.controller.ts, src\graph\graph.controller.ts, and src\app.controller.ts

```
@Controller()

export class QueryController {
  private readonly logger = new Logger(QueryService.name);
  constructor(
    private readonly queryService: QueryService,
    private readonly priceService: PriceService,
  ) {}

  @Get('ui_version')
  getUIVersion(@Res() res): string {
    const version = this.queryService.getUIVersion();
    return res.status(HttpStatus.OK).send(version);
  }

  @Get('rap_supply')
  getRapSupply(@Res() res): string {
    const supply = this.queryService.getRapSupply();
```

```
    return res.status(HttpStatus.OK).send(supply);
  }

  @Get('total_volume')
  async getTotalVolume(@Res() res) {
    const totalVolume = await this.queryService.getTotalVolume();

    return res.status(HttpStatus.OK).json([{ data: { volume:
totalVolume } }]);
  }

  @Get('fees_summary')
  async getFeesSummary(@Res() res) {
    const ret = await this.queryService.getFeesSummary();

    return res.status(HttpStatus.OK).json(ret);
  }

  @Get('prices')
  async prices(@Res() res) {
    const result = await this.priceService.getPrice();

    if (result) {
      result[contractSetting['pacific-
testnet'].allTokens.BTC.address] =
        ethers.utils.parseUnits(result['BTC'], 30).toString();
      result[contractSetting['pacific-
testnet'].allTokens.ETH.address] =
        ethers.utils.parseUnits(result['ETH'], 30).toString();

      return res.status(HttpStatus.OK).json(result);
    } else {
      this.logger.log('cant get price');
      return res.status(HttpStatus.SERVICE_UNAVAILABLE).send();
    }
  }

  @Get('orders_indices')
  async getOrdersIndices(@Res() res, @Query() query: any) {
    if (!query.account) {
```

```
      return res.status(HttpStatus.BAD_REQUEST).send('Bad Request');
    }
    const ret = await this.queryService.getOrdersIndices(
      query.account.toLowerCase(),
    );
    return res.status(HttpStatus.OK).json(ret);
  }

  @Get('actions')
  async getActions(@Res() res, @Query() query: any) {
    let ret = [];
    if (!query.account) {
      // query all actions
      ret = await this.queryService.getAllActions();
    } else {
      // query specific account
      ret = await this.queryService.getActionsByAccount(
        query.account.toLowerCase(),
      );
    }
    return res.status(HttpStatus.OK).json(ret);
  }

  @Get('position_stats')
  async getPositionStats(@Res() res) {
    const ret = await this.queryService.getPositionStats();
    if (!ret) {
      return res.status(HttpStatus.OK).json({
        totalShortPositionCollaterals: '0',
        totalShortPositionSizes: '0',
        totalLongPositionSizes: '0',
        totalActivePositions: 0,
        totalLongPositionCollaterals: '0',
      });
    }
    ...
}
```

The code does not specify the

Origin	The source URI of the surface preflight request or the actual request, which is sent by default for browser requests.
Access-Control-Request-Method	Inform the server of the HTTP method used for the actual request
Access-Control-	Inform the service of the first field carried by the actual request

judgmental processing leading to possible CORS vulnerability hazards.

This can lead to a possible CORS vulnerability.

- **Security Advice**

To fix the CORS issue, add the response header Access-Control-Allow-Origin and set it to the legitimate source domain to ensure that only specific sources can access your resources for added security.

- **Repair Status**

Resolved.

4.3 arbitrary affiliation update

- **Risk description**

According to the code src\referral\referral.controller.ts inside the updateOneTier method you can see that the referralService.updateOneTier method is called and does not filter affiliates.

```
@Post('updateOneTier')

async updateOneTier(@Body() input: any) {
  return await this.referralService.updateOneTier(input.affiliate);
}
```

Continuing to follow up on the updateOneTier method of src\referral\referral.service.ts, we found that the processOneAffiliate of src\referral\referral.service.ts was called.

```
async updateOneTier(id: string) {

  const affiliates = await this.queryOneAffiliates(id);
  if (affiliates.length == 0) {
    return null;
  }
  const affiliate = affiliates[0];
  return await this.processOneAffiliate(id, affiliate.tierId);
}
```

Followed up processOneAffiliate in src\referral\referral.service.ts and found that it can be updated directly.

```
async processOneAffiliate(affiliate: string, beforeTier: number,
backdate: number = 7) {

  // First check the total number of trades vs. total referrals,
  // if the total doesn't meet the minimum rank, then the weekly
certainly doesn't either
  const statsData = await this.needProcess(affiliate);
  if (!statsData.fulfill) {
    return statsData;
  }
  const date = new Date();
  date.setDate(date.getDate() - backdate);
  const lastWeekTs = date.setHours(0,0,0,0)/1000;
  // It is possible that the total data meets the minimum level that
can be upgraded,
  // but the weekly data does not, and then the level needs to be
rolled back
  const weeklyData = await this.calcWeeklyAffiliateStats(affiliate,
lastWeekTs);
  // Calculate whether you can upgrade your level based on one
week's statistics
  const targetTier = this.targetTier(beforeTier,
weeklyData.usdVolume, weeklyData.uniqReferralsCount);
  // No need to set setReferrerTier if the level hasn't changed.
  if (beforeTier == targetTier) {
    return {
      affiliate,
      fulfill: true,
```

```
      beforeTier,
      targetTier,
      updateTier: beforeTier != targetTier,
      executeResult: false,
      data: weeklyData
    };
  }
  let executeMessage = "success";
  try {
    await this.sdkService.setReferrerTier(affiliate, targetTier);
  } catch (err) {
    this.logger.info(`set tier ${affiliate} to ${targetTier} err:
${err.toString()}`, err);
    executeMessage = err.toString();
  }
  return {
    affiliate,
    fulfill: true,
    beforeTier,
    targetTier,
    updateTier: beforeTier != targetTier,
    executeResult: executeMessage == "success",
    executeMessage,
    data: weeklyData
  };
}
```

- **Security Advice**

Add permission checks to this src\referral\referral.controller.ts

- **Repair Status**

Resolved.

4.4 Input unfiltered leading code injection

- **Risk description**

src\jobs\jobs.controller.ts not code filtering on input query.name

May lead to code injection risks

```
@Get('test/queryJob')
async queryJob(@Query() query: any) {
  const job = this.schedulerRegistry.getCronJob(query.name);
  return `job: ${query.name}, running: ${job.running}`;
}
```

- **Security Advice**

Keyword filtering on user-entered query.name, such as eval, exec, etc.

- **Repair Status**

Resolved.

5. Medium Risk vulnerabilities

5.1 Full controller has CORS vulnerability

- **Risk description**

The vulnerability exists in a request inside price\price.controller.ts where the source header is not validated resulting in a cors vulnerability

```
@Controller('price')
export class PriceController {
  constructor(private readonly priceService: PriceService) {}

  @Get('getMarkedPrice')
  async getMarkedPrice() {
    const result = wrapResponse(() =>
this.priceService.getPrice('cex_marked'));
    return result;
  }

  @Get('getOraclePrice')
  async getOraclePrice() {
    const result = wrapResponse(() =>
this.priceService.getPrice('oracle'));
    return result;
  }
}
```

The code does not specify the

Origin	The source URI of the surface preflight request or the actual request, which is sent by default for browser requests.
Access-Control-Request-Method	Inform the server of the HTTP method used for the actual request
Access-Control-	Inform the service of the first field carried by the actual request

judgmental processing leading to possible CORS vulnerability hazards.

- **Security Advice**

To fix the CORS issue, add the response header Access-Control-Allow-Origin and set it to the legitimate source domain to ensure that only specific sources can access your resources for added security.

- **Repair Status**

Resolved.

6. Low Risk Vulnerabilities

6.1 log injection

- **Risk description**

Found that the code src\sdk\sdk.helper.ts inside the getNextToAmount console.log did not handle the length of the string, or the content of the string, resulting in a direct console.log can be injected!

```
export function getNextToAmount(

  fromAmount: BigNumber,
  fromTokenAddress: string,
  toTokenAddress: string,
  tokenInfo: TokenInfoType,
  totalTokenWeights: BigNumber,
  STABLE_SWAP_FEE_BASIS_POINTS: BigNumber,
  SWAP_FEE_BASIS_POINTS: BigNumber,
  STABLE_TAX_BASIS_POINTS: BigNumber,
  TAX_BASIS_POINTS: BigNumber,
  usdgSupply: BigNumber, // total usdg supply
) {
  if (!fromAmount || !fromTokenAddress || !toTokenAddress
|| !tokenInfo) {
    console.log(
      `null occurs: fromAmount : ${fromAmount} ,
fromTokenAddress :${fromTokenAddress},
toTokenAddress:${toTokenAddress} tokenInfo: ${JSON.stringify(
        tokenInfo,
        null,
        4,
      )}`,
    );
    return null;
  }
  ...
}
```

- **Security Advice**

Process the contents of console.log for length judgment or remove the variable

- **Repair Status**

Resolved.

6.2 Input Not Validated

- **Risk description**

Code located to price\price.controller.ts testSetPrice method inside type not specified input type

```
@Get('test/setPrice')

async testSetPrice(@Query() query: any) {
  const type = query.type as PriceType;
  if (!type) {
    return{
      success: false,
      message:`Invalid query type'message`,
    }
    const price = {
      time: Date.now(),
      data: {},
    };

    Object.keys(query).forEach((key: string) => {
      const token = TOKEN_SYMBOL[key] || STABLE_TOKEN_SYMBOL[key];
      if (!token) {
        return;
      }
      price.data[token] = query[key] || '0';
    });
    const result = wrapResponse(() =>
      this.priceService.setTestPrice(type,price as PriceData)
    );
  }
}
```

Type inputs that should be specified as PriceType to avoid potential security vulnerabilities

- **Security Advice**

Some query parameter extraction and processing is performed in the code, but there is not enough validation of these parameters. It is recommended that the input parameters be validated to ensure that they are valid, acceptable values. It is possible to validate that the type parameter is a valid PriceType value.

- **Repair Status**

Resolved.

7. Information Alerts

7.1 Job usage issues

- **Risk description**

If you use a timed job in the background to perform a series of clearing tasks or order tasks, because the speed of the blocks on the chain is only an average speed, there will be fluctuations, which may lead to deviations in the time of the blocks, and ultimately lead to the possibility that some tasks will miss the processing of a certain block or so on.

```
@Cron('*/15 * * * * *', {
  name: 'marketPricePositionRequests',
})
async handleMarketPricePositionRequests() {
  if (!(await this.canRunning())) return;
  this.logger.log('Handle Market Price Position Requests...');
  await this.sdkService.executeMarketPricePositionRequests();
  this.logger.log('Market Price Position Requests Handled');
}
@Cron('*/15 * * * * *', {
  name: 'updateSwapOrders',
})
async handleUpdateSwapOrders() {
  if (!(await this.canRunning())) return;
  this.logger.log('Updating SwapOrders...');
  await this.sdkService.updateSwapOrders();
  this.logger.log('SwapOrders Updated');
}
@Cron('*/15 * * * * *', {
  name: 'updateIncreasePositionOrders',
})
async handleUpdateIncreasePositionOrders() {
  if (!(await this.canRunning())) return;
  this.logger.log('Updating IncreasePositionOrders...');
  await this.sdkService.updateIncreasePositionOrders();
  this.logger.log('IncreasePositionOrders Updated');
}
@Cron('*/15 * * * * *', {
  name: 'updateDecreasePositionOrders',
```

```
})
async handleUpdateDecreasePositionOrders() {
  if (!(await this.canRunning())) return;
  this.logger.log('Updating DecreasePositionOrders...');
  await this.sdkService.updateDecreasePositionOrders();
  this.logger.log('DecreasePositionOrders Updated');
}
```

- **Security Advice**

Change some jobs to rpc high-level subscriptions to complete real-time change monitoring and execution.

- **Repair Status**

Acknowledged

7.2 excessive privileges

- **Risk description**

The signature used in the automation operation in the back-end program is in the chain contract in the authority, the authority is larger, in addition to the main existing automation call function, there are a lot of other privileged functions locally not called, but their signatures have the authority to call, then there may be a private key leakage or the content of the person committing a crime leading to other privileged functions are maliciously invoked.

```
// Only this function has a call relationship in the automation
program
function batchWithdrawFees(address _vault, address[] memory _tokens)
external onlyKeeperAndAbove {
    for (uint256 i = 0; i < _tokens.length; i++) {
        IVault(_vault).withdrawFees(_tokens[i], admin);
    }
}
```

```
function setIsSwapEnabled(address _vault, bool _isSwapEnabled)
external onlyKeeperAndAbove {
    IVault(_vault).setIsSwapEnabled(_isSwapEnabled);
}
function setReferrerTier(address _referralStorage, address
_referrer, uint256 _tierId) external onlyKeeperAndAbove {
    IReferralStorage(_referralStorage).setReferrerTier(_referrer,
_tierId);
}
function govSetCodeOwner(address _referralStorage, bytes32 _code,
address _newAccount) external onlyKeeperAndAbove {
    IReferralStorage(_referralStorage).govSetCodeOwner(_code,
_newAccount);
}
function batchSetBonusRewards(address _vester, address[] memory
_accounts, uint256[] memory _amounts) external onlyKeeperAndAbove {
    require(_accounts.length == _amounts.length, "Timelock: invalid
lengths");
    IHandlerTarget(_vester).setHandler(address(this), true);
    for (uint256 i = 0; i < _accounts.length; i++) {
        address account = _accounts[i];
        uint256 amount = _amounts[i];
        IVester(_vester).setBonusRewards(account, amount);
    }
    IHandlerTarget(_vester).setHandler(address(this), false);
}
function setTokenConfig(
    address _vault,
    address _token,
    uint256 _tokenWeight,
    uint256 _minProfitBps,
    uint256 _maxUsdgAmount,
    uint256 _bufferAmount,
    uint256 _usdgAmount
) external onlyKeeperAndAbove {
    require(_minProfitBps <= 500, "Timelock: invalid
_minProfitBps");
    IVault vault = IVault(_vault);
    require(vault.whitelistedTokens(_token), "Timelock: token not
yet whitelisted");
```

```
    uint256 tokenDecimals = vault.tokenDecimals(_token);
    bool isStable = vault.stableTokens(_token);
    bool isShortable = vault.shortableTokens(_token);
    IVault(_vault).setTokenConfig(
        _token,
        tokenDecimals,
        _tokenWeight,
        _minProfitBps,
        _maxUsdgAmount,
        isStable,
        isShortable
    );
    IVault(_vault).setBufferAmount(_token, _bufferAmount);
    IVault(_vault).setUsdgAmount(_token, _usdgAmount);
}
function setUsdgAmounts(address _vault, address[] memory _tokens,
uint256[] memory _usdgAmounts) external onlyKeeperAndAbove {
    for (uint256 i = 0; i < _tokens.length; i++) {
        IVault(_vault).setUsdgAmount(_tokens[i], _usdgAmounts[i]);
    }
}
function updateUsdgSupply(uint256 usdgAmount) external
onlyKeeperAndAbove {
    address usdg = IGlpManager(glpManager).usdg();
    uint256 balance = IERC20(usdg).balanceOf(glpManager);
    IUSDG(usdg).addVault(address(this));
    if (usdgAmount > balance) {
        uint256 mintAmount = usdgAmount.sub(balance);
        IUSDG(usdg).mint(glpManager, mintAmount);
    } else {
        uint256 burnAmount = balance.sub(usdgAmount);
        IUSDG(usdg).burn(glpManager, burnAmount);
    }
    IUSDG(usdg).removeVault(address(this));
}
function setSwapFees(
    address _vault,
    uint256 _taxBasisPoints,
    uint256 _stableTaxBasisPoints,
    uint256 _mintBurnFeeBasisPoints,
```

```
    uint256 _swapFeeBasisPoints,
    uint256 _stableSwapFeeBasisPoints
) external onlyKeeperAndAbove {
    IVault vault = IVault(_vault);

    vault.setFees(
        _taxBasisPoints,
        _stableTaxBasisPoints,
        _mintBurnFeeBasisPoints,
        _swapFeeBasisPoints,
        _stableSwapFeeBasisPoints,
        maxMarginFeeBasisPoints,
        vault.liquidationFeeUsd(),
        vault.minProfitTime(),
        vault.hasDynamicFees()
    );
}

function setFees(
    address _vault,
    uint256 _taxBasisPoints,
    uint256 _stableTaxBasisPoints,
    uint256 _mintBurnFeeBasisPoints,
    uint256 _swapFeeBasisPoints,
    uint256 _stableSwapFeeBasisPoints,
    uint256 _marginFeeBasisPoints,
    uint256 _liquidationFeeUsd,
    uint256 _minProfitTime,
    bool _hasDynamicFees
) external onlyKeeperAndAbove {
    marginFeeBasisPoints = _marginFeeBasisPoints;
    IVault(_vault).setFees(
        _taxBasisPoints,
        _stableTaxBasisPoints,
        _mintBurnFeeBasisPoints,
        _swapFeeBasisPoints,
        _stableSwapFeeBasisPoints,
        maxMarginFeeBasisPoints,
        _liquidationFeeUsd,
        _minProfitTime,
```

```
        _hasDynamicFees
    );
}
```

- **Security Advice**

It is recommended to narrow down the permissions of their roles in the contract and open up the function call permissions on an as-needed basis

- **Repair Status**

Acknowledged

8. Audit details

8.1 Findings Summary

Severity	Found	Resolved	Acknowledged
High	4	**4**	0
Medium	1	1	0
Low	2	2	0
Info	2	0	2

8.2 Risk distribution

application system	Vulnerability	number of individuals	hierarchy	Repair status
RealPerp Code Audit	privilege checking	1	high	resolved
	Site-wide cors	1	high	resolved
	Anyupdate	1	high	resolved
	code injection	1	high	resolved
	cors	1	Medium	resolved
	log injection	1	low	resolved
	Input not validated	1	low	resolved
	Job usage issues	1	Info	acknowledged
	excessive privileges	1	Info	acknowledged

Disclaimer：

Lunaray Technology only issues a report and assumes corresponding responsibilities for the facts that occurred or existed before the issuance of this report, Since the facts that occurred after the issuance of the report cannot determine the security status of the smart contract, it is not responsible for this.

Lunaray Technology conducts security audits on the security audit items in the project agreement, and is not responsible for the project background and other circumstances, The subsequent deployment and operation methods of the project party are beyond the scope of this audit.

This report only conducts a security audit based on the information provided by the information provider to Lunaray at the time the report is issued, If the information of this project is concealed or the situation reflected is inconsistent with the actual situation, Lunaray Technology shall not be liable for any losses and adverse effects caused thereby.

There are risks in the market, and investment needs to be cautious. This report only conducts security audits and results announcements on smart contract codes, and does not make investment recommendations and basis.

https://lunaray.co

https://github.com/lunaraySec

https://twitter.com/lunaray_co

http://t.me/lunaraySec