# Competitive Security Assessment

## Mantle-LSD-Oracle-Service-Core

Nov 6th, 2023

Secure3

secure3.io

# Summary

This report is prepared for the project to identify vulnerabilities and issues in the smart contract source code. A group of NDA covered experienced security experts have participated in the Secure3's Audit Contest to find vulnerabilities and optimizations. Secure3 team has participated in the contest process as well to provide extra auditing coverage and scrutiny of the finding submissions.

The comprehensive examination and auditing scope includes:

- Cross checking contract implementation against functionalities described in the documents and white paper disclosed by the project owner.
- Contract Privilege Role Review to provide more clarity on smart contract roles and privilege.
- Using static analysis tools to analyze smart contracts against common known vulnerabilities patterns.
- Verify the code base is compliant with the most up-to-date industry standards and security best practices.
- Comprehensive line-by-line manual code review of the entire codebase by industry experts.

The security assessment resulted in findings that are categorized in four severity levels: Critical, Medium, Low, Informational. For each of the findings, the report has included recommendations of fix or mitigation for security and best practices.

# Overview

### Project Detail

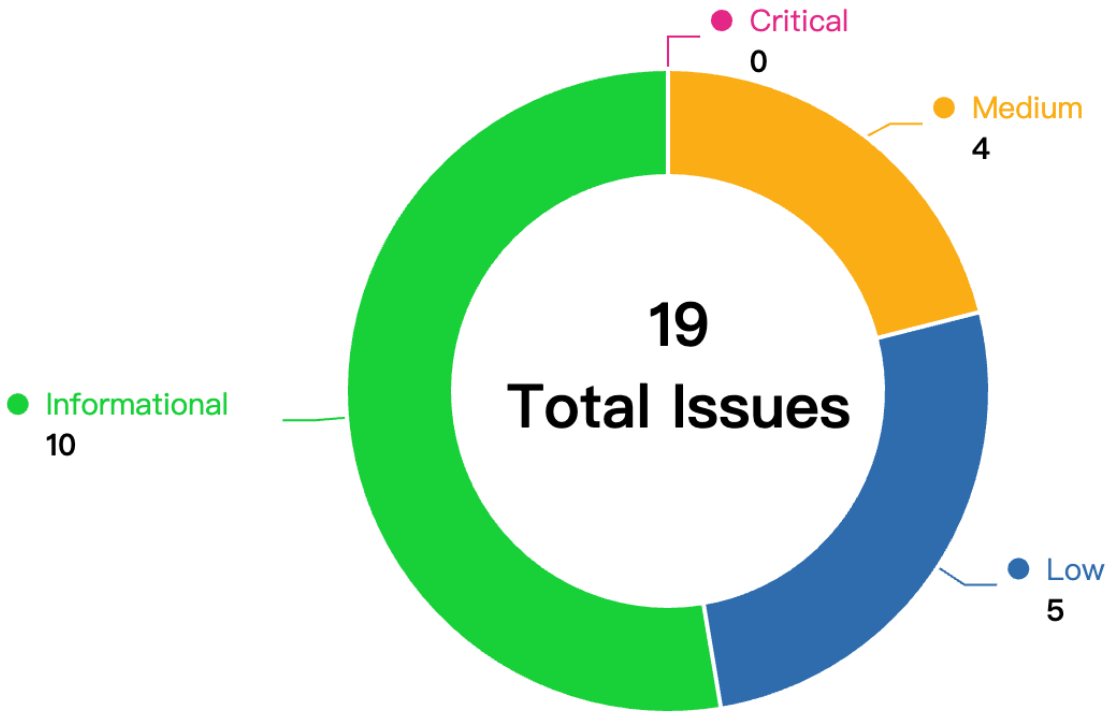
| | |
|---|---|
| **Project Name** | Mantle-LSD-Oracle-Service-Core |
| **Platform & Language** | Go |
| **Codebase** | • https://github.com/TwoFiftySixLabs/services<br>• audit commit - 44c07aec02eaa81fab10ab20c7e88787427bc0bd<br>• final commit - 0416e2ff42461ab131e391b15ea9c8a48e6eec65 |
| **Audit Methodology** | • Audit Contest<br>• Business Logic and Code Review<br>• Privileged Roles Review<br>• Static Analysis |

# Audit Scope

| File | SHA256 Hash |
|---|---|
| ./oracle/main.go | c00b8087d8f884f62363f4fd81b5c31daae3afa0818d92095d62db79d05dcf80 |
| ./oracle/scheduler/scheduler.go | bf8031339ef7b9488c47a03076afd5427832e93492d64f50a9b275e6546f2db2 |
| ./oracle/service/service.go | 9b06118c3985c7f97361a0499dc4bb5af45688f65c1d7bbb4cc8c20cf4bd30e1 |
| ./lib/withdrawals/withdrawals.go | aac9410867fd2450730beaf2696ab6ef882888189417246 72a69b33bdd2fde33 |
| ./oracle/reporter/reporter.go | f84c04b9ab83d0a13e8a15742aaa75db10132f3dcdf719f985816bacb3d27e9f |
| ./lib/consensus/finder/finder.go | ae92d1d3d2df3d5071e0e2c73a4cd998ad2eac907752ce29315d88330ab48f0b |
| ./lib/eventsearcher/events.go | a5f34854821f1ff854e543fabecda27bbef26b1394b802d838fe7733794ac6c4 |
| ./lib/consensus/blockstamp.go | d59b9bfb7b8ee054bc8a356bd088ea3f7fce5337776d8e22b90aa1d01c7f1c65 |
| ./lib/validator/loader.go | 216b3285a789b9e5983b49f580b6d63b0ed3b876e1b3d5964a724d26d81968cb |
| ./lib/validator/sourcer/sourcer.go | 599e32589fdf325d1d78dc15841bab3613cb7666433aacda7131491b3523bad3 |
| ./lib/transaction/gcpkmssigner.go | a34e377cfdb1994699fdbf4277f9477b793c295def694459a8a03f8ee7d38fe2 |
| ./oracle/service/config.go | 818af00dbcfebf649c663fe1aab7b0378c284fe3f92cfb9a207ccf1515a5a6e8 |
| ./lib/validator/validator.go | f3de4ef10678ea238fc582054ea463ce0271500c85347c4b5e6e6ba8d1121af4 |
| ./lib/consensus/pubkeyfilter.go | 21fd5a93deb2dec6f60fde65cf275a562f4fe529ef0403f1ef8c686a46654da1 |
| ./lib/runner/runner.go | 89235295ca231598a0ae682403f7b7b4874836cb02764fea1e32b42677fb3c0d |

| | |
|---|---|
| ./lib/validator/keyset.go | 1d72f98030044d186db700752357e4e56f675bfaeac1efef95f572c3751a5158 |
| ./lib/client/multiplex.go | b4f0f34729ca5fadf50d10d4e53a4f962d93e22c4d77715522b297c6637daa14 |
| ./lib/consensus/client.go | 474c6aefaf489e608a5bda02cb32e8bfaf6e8768dc723ac4527a35d4ff75b458 |
| ./lib/validator/filter.go | b29bacc5ac6ae088c4d47f09e28cd8ce7c595f0a6e91e7078d7a93c6aa9d46c5 |
| ./lib/consensus/constants.go | 7c55057506f1865b6c794020fd08072e6f4c92ad55e1e29d0ad59abde999e558 |
| ./lib/validator/predicates.go | 90ac9fbce9b02cc693a6bb5d4a819f4706c775ab024767fd8bf6af1ae9613b7a |
| ./oracle/service/clients.go | b7eb356ec7c83c629818ee70c001c62b0bbbeedf8eb797eb3a5575da343b91c3 |
| ./lib/validator/sourcer/set.go | 342572dff97bcb680a8b6d13459269b0b4f3ddf328f6747b37c3b14cc5782867 |
| ./lib/network/network.go | 8c85dd08132ee63a647bd97647fa40c9dc4a61bf2084ab857ad5d07c38fb221a |

# Code Assessment Findings




| ID | Name | Category | Severity | Client Response | Contributor |
|---|---|---|---|---|---|
| MTO-1 | Potential panic when executing a transaction | Logical | Medium | Fixed | biakia |
| MTO-2 | Potential Nil Dereference | Code Style | Medium | Fixed | lfzkoala |
| MTO-3 | Lack of Rate Limiting Control | DOS | Medium | Fixed | lfzkoala |
| MTO-4 | Change `ethclient.Dial(conf.RPCUrl)` to `ethclient.DialContext(ctx, conf.RPCUrl)` Thus timeout function also takes effect on ethclient | Logical | Medium | Fixed | seek.guo |

| | | | | | |
|---|---|---|---|---|---|
| MTO-5 | Lack of Authentication for Ethereum RPC | Logical | Medium | Declined | biakia, lfzkoala |
| MTO-6 | Insufficient Input Validation in `ComputeWithdrawals` Method | Logical | Medium | Declined | yekong, lfzkoala, Hacker007 |
| MTO-7 | Lack of Secure Headers in HTTP Responses | Code Style | Medium | Declined | lfzkoala |
| MTO-8 | Resources not released after querying events | Logical | Low | Fixed | biakia |
| MTO-9 | Not check the existence of the block returned from the Beacon. | Logical | Low | Fixed | biakia |
| MTO-10 | Incomplete nil check | Logical | Low | Fixed | Hacker007 |
| MTO-11 | Missing parameter validation in `withdrawals::NewAnalyzer()` function | Logical | Low | Fixed | Hupixiong3 |
| MTO-12 | start-block and end-block flags should define Uint64Flag, num-runs and start-index flag should use UintFlag, Because they cannot have negative numbers | Logical | Low | Fixed | seek.guo |
| MTO-13 | Uncontrolled Public Key Input in ValidatorsAt | Logical | Low | Declined | lfzkoala |
| MTO-14 | No further error handling in `scheduler::checkForUpdate()` function | Logical | Low | Declined | Hupixiong3 |
| MTO-15 | Optimization for the binary search | Logical | Informational | Fixed | biakia, alansh |
| MTO-16 | wrong error info in `Reporter::BuildOracleReport` function | Logical | Informational | Fixed | alansh |
| MTO-17 | real reason for error from `consensusClient` is unexpectedly ignored in `BinarySearch::BlockStamp()` function | Code Style | Informational | Fixed | alansh |
| MTO-18 | Incomplete Configuration Checks | Logical | Informational | Fixed | lfzkoala |
| MTO-19 | gc optimization for command `regenerate-reports` and `verify-reports` | Logical | Informational | Acknowledged | biakia |

| | | | | | |
|---|---|---|---|---|---|
| MTO-20 | Function Name Mismatch | Code Style | Informational | Fixed | yekong |
| MTO-21 | Inefficient loop statement in `withdrawals.go` | Gas Optimization | Informational | Acknowledged | Hacker007 |
| MTO-22 | Unused function parameter `ctx` | Language Specific | Informational | Fixed | Hacker007 |
| MTO-23 | Inefficiency in Memory Allocation in `keyset::PublicKeys()` function | Language Specific | Informational | Acknowledged | Hupixiong3 |
| MTO-24 | Unused field `Epoch` | Language Specific | Informational | Fixed | Hacker007 |

# MTO-1:Potential panic when executing a transaction

| Category | Severity | Client Response | Contributor |
| --- | --- | --- | --- |
| Logical | Medium | Fixed | biakia |

## Code Reference

- code/lib/transaction/gcpkmssigner.go#L41-L67

```
41:func (s *GCPKMSTransactor) Run(ctx context.Context, f TransactorFunc) (*types.Transaction, error)
{
42:     nonce, err := s.noncer.NonceAt(ctx, s.Address(), nil)
43:     if err != nil {
44:             return nil, errors.Wrap(err, "failed to get nonce for transaction")
45:     }
46:
47:     opts := &bind.TransactOpts{
48:             From: s.Address(),
49:             Signer: func(address common.Address, tx *types.Transaction) (*types.Transaction, err
or) {
50:                     return s.sign(ctx, tx)
51:             },
52:             Context: ctx,
53:             Nonce:   big.NewInt(int64(nonce)),
54:     }
55:     tx, err := f(opts)
56:     if tx != nil {
57:             logger.Debugw("Sent transaction", "hash", tx.Hash().Hex(), "nonce", nonce)
58:     }
59:     if err != nil {
60:             extractedError, ok := extractErrorMessage(err)
61:             if ok {
62:                     logger.Infow("Transaction failed", "hash", tx.Hash().Hex(), "nonce", nonce,
"error", extractedError)
63:                     return tx, errors.Wrapf(err, "transaction failed with error from the contrac
t %s", extractedError)
64:             }
65:     }
66:     return tx, err
67:}
```

## Description

**biakia :** The `gcpkmssigner.go` is a Ethereum client using Google Cloud KMS. The function `Run` is used to execute a transaction which is signed remotely with the managed key. If the execution fails, it will handle the returned error:

```
        tx, err := f(opts)
        if tx != nil {
                logger.Debugw("Sent transaction", "hash", tx.Hash().Hex(), "nonce", nonce)
        }
        if err != nil {
                extractedError, ok := extractErrorMessage(err)
                if ok {
                        logger.Infow("Transaction failed", "hash", tx.Hash().Hex(), "nonce", nonce,
"error", extractedError)
                        return tx, errors.Wrapf(err, "transaction failed with error from the contrac
t %s", extractedError)
                }
        }
```

It will call `extractErrorMessage` to get the `extractedError` and log it:

```
  logger.Infow("Transaction failed", "hash", tx.Hash().Hex(), "nonce", nonce, "error", extractedError)
```

The issue here is that it is possible that the `tx` is `nil`. If the `tx` is nil, the `tx.Hash()` will be panic and the program will be terminated.

## Recommendation

**biakia :** Consider checking the `tx` when logging the error.

## Client Response

Fixed

# MTO-2:Potential Nil Dereference

| Category | Severity | Client Response | Contributor |
| --- | --- | --- | --- |
| Code Style | Medium | Fixed | lfzkoala |

## Code Reference

- code/lib/validator/sourcer/set.go#L15

```
15:func (s ValidatorSet) NodeOperatorID(key string) nodeoperators.ID {
```

## Description

**lfzkoala :** When trying to retrieve a `nodeoperators.ID` using a key that doesn't exist in the map, the method `NodeOperatorID` will return the zero value for the type (`nodeoperators.ID`). This may not be an immediate issue, but depending on how the zero value is used or interpreted elsewhere, it might lead to logical errors or potential vulnerabilities.

If a caller assumes that a valid `nodeoperators.ID` is always returned and doesn't handle the zero value, it might lead to bugs or unexpected behavior.

## Recommendation

**lfzkoala :** Provide a mechanism to check if the value actually exists in the map. Return an error or a boolean indicating the presence of the value.

```
func (s ValidatorSet) NodeOperatorID(key string) (nodeoperators.ID, bool) {
        val, ok := s[key]
        return val, ok
}
```

## Client Response

Fixed, but severity is inflated as this cannot actually happen in the current code.

# MTO-3:Lack of Rate Limiting Control

| Category | Severity | Client Response | Contributor |
| --- | --- | --- | --- |
| DOS | Medium | Fixed | lfzkoala |

## Code Reference

- code/lib/withdrawals/withdrawals.go#L87

```
87:limiter := rate.NewLimiter(rate.Limit(a.maxRPS), a.maxRPS)
```

## Description

**lfzkoala :** The Analyzer uses a rate limiter ( `rate.NewLimiter` ) for controlling the frequency of its operations; however, it relies on the `maxRPS` parameter from the user without validating its value. If a user initializes the `Analyzer` with an extremely high `maxRPS` or with a negative value, it could lead to unexpected system behavior. **Proof of Concept of Attack with code example:** Creating an `Analyzer` with a negative `maxRPS` :

```
analyzer := NewAnalyzer(client, -10, address)
```

**Location of the security issue with code snippet:**

```
limiter := rate.NewLimiter(rate.Limit(a.maxRPS), a.maxRPS)
```

## Recommendation

**lfzkoala :** Add validation for `maxRPS` during the `Analyzer` initialization to ensure it's within a safe range.

```
if maxRPS <= 0 || maxRPS > someSafeUpperLimit {
    return nil, errors.New("Invalid maxRPS value")
}
```

## Client Response

Fixed,
All configurations are currently hardcoded as positive integers. However, we think that it is likely they will become configur able later and therefore it is preferable to have this check.

# MTO-4:Change `ethclient.Dial(conf.RPCUrl)` to `ethclient.DialContext(ctx, conf.RPCUrl)` Thus timeout function also takes effect on ethclient

| Category | Severity | Client Response | Contributor |
| --- | --- | --- | --- |
| Logical | Medium | Fixed | seek.guo |

## Code Reference

- code/oracle/service/clients.go#L25

```
25:ethClient, err := ethclient.Dial(conf.RPCUrl)
```

## Description

**seek.guo :** Change `ethclient.Dial(conf.RPCUrl)` to `ethclient.DialContext(ctx, conf.RPCUrl)` Thus timeout function also takes effect on ethclient

## Recommendation

**seek.guo :** change this

```
ethClient, err := ethclient.Dial(conf.RPCUrl)
if err != nil {
    return nil, errors.Wrap(err, "failed to create execution client")
}
```

to

```
ethClient, err := ethclient.DialContext(ctx, conf.RPCUrl)
if err != nil {
     return nil, errors.Wrap(err, "failed to create execution client")
}
```

## Client Response

Fixed

# MTO-5:Lack of Authentication for Ethereum RPC

| Category | Severity | Client Response | Contributor |
| --- | --- | --- | --- |
| Logical | Medium | Declined | biakia, lfzkoala |

## Code Reference

- code/oracle/service/clients.go#L25
- code/oracle/main.go#L262-L281

```
25:ethClient, err := ethclient.Dial(conf.RPCUrl)

262:go func(port int) {
263:            logger.Infow("Health check", "port", port)
264:            http.HandleFunc("/", func(w http.ResponseWriter, r *http.Request) {
265:                    w.WriteHeader(http.StatusOK)
266:                    _, err := w.Write([]byte("ok"))
267:                    if err != nil {
268:                            logger.Logger.Fatal("error health check", "error", err)
269:                    }
270:            })
271:
272:            addr := ":" + strconv.Itoa(port)
273:            server := &http.Server{
274:                    Addr:              addr,
275:                    ReadHeaderTimeout: 30 * time.Second,
276:            }
277:            err := server.ListenAndServe()
278:            if err != nil {
279:                    logger.Logger.Fatal("error health check", "error", err)
280:            }
281:    }(conf.HealthCheckPort)
```

## Description

**biakia :** In `main.go`, there is a health-check http service without authentication mechanism:

```
	go func(port int) {
		logger.Infow("Health check", "port", port)
		http.HandleFunc("/", func(w http.ResponseWriter, r *http.Request) {
			w.WriteHeader(http.StatusOK)
			_, err := w.Write([]byte("ok"))
			if err != nil {
				logger.Logger.Fatal("error health check", "error", err)
			}
		})

		addr := ":" + strconv.Itoa(port)
		server := &http.Server{
			Addr:              addr,
			ReadHeaderTimeout: 30 * time.Second,
		}
		err := server.ListenAndServe()
		if err != nil {
			logger.Logger.Fatal("error health check", "error", err)
		}
	}(conf.HealthCheckPort)
```

If the server port is exposed to the public, the service may be subject to a ddos attack, which can cause the entire program to crash.

**lfzkoala :** The code uses `ethclient.Dial` to connect to the Ethereum RPC endpoint without any checks for authentication. If the RPC endpoint is not publicly accessible (which it generally shouldn't be), this can expose it to unauthorized access.

Someone can scan for open RPC ports and if they find one, they can gain unrestricted access to the Ethereum node, potentially leading to various attack vectors, like reading private information, broadcasting malicious transactions, etc.

## Recommendation

**biakia :** Consider using JWT or making sure the server not be exposed to the public.

**lfzkoala :** 1. Always ensure that your Ethereum node's RPC endpoint is not exposed to the public. 2. If you must expose it, use an authentication mechanism. `ethclient.Dial` supports basic authentication via URL. The URL can be of the form `http://username:password@localhost:8545`. 3. Another more secure method would be to use a VPN or another networking solution to ensure that only authorized clients can connect to the RPC.

## Client Response

Declined, The service is not exposed to the public and has no endpoints other than health or metrics, neither of which do RPC calls. Regardless, the client is autheticated through the usual mechansm of including the token in the URL.

`Secure3 comment`: Taking this measure can add one more layer of security to your intranet.

# MTO-6:Insufficient Input Validation in `ComputeWithdrawals` Method

| Category | Severity | Client Response | Contributor |
| --- | --- | --- | --- |
| Logical | Medium | Declined | yekong, lfzkoala, Hacker007 |

## Code Reference

- code/lib/withdrawals/withdrawals.go#L57-L69
- code/lib/withdrawals/withdrawals.go#L71-L74

```
57:func (a *Analyzer) ComputeWithdrawals(ctx context.Context, begin, end *consensus.BlockStamp, endV
alidators []*validator.Validator) (principal *big.Int, reward *big.Int, err error) {
58:     withdrawals, err := a.findWithdrawalsInRange(ctx, begin, end)
59:     if err != nil {
60:             return nil, nil, errors.Wrap(err, "failed to find withdrawals")
61:     }
62:     // Find only withdrawals for our validators.
63:     ourWithdrawals := a.filterOurWithdrawals(withdrawals, valIndicesSet(endValidators))
64:
65:     // Find validators which are withdrawn (i.e. have become withdrawn in this range).
66:     withdrawnValidators := validator.Filter(endValidators, validator.HasState(v1.ValidatorStateW
ithdrawalDone))
67:     principal, reward = a.sumWithdrawals(ourWithdrawals, withdrawnValidators)
68:     return principal, reward, nil
69:}

71:func (a *Analyzer) findWithdrawalsInRange(ctx context.Context, begin, end *consensus.BlockStamp)
([]*Withdrawal, error) {
72:     logger.Debugw("Finding withdrawals in slots", "begin", begin.SlotNumber, "end", end.SlotNumb
er)
73:
74:     tasks := make([]runner.TaskFunc[[]*Withdrawal], end.SlotNumber-begin.SlotNumber+1)
```

## Description

**yekong :** The ComputeWithdrawals function in the withdrawals package accepts external parameters `begin` and `end` representing block stamps. However, the function doesn't appear to validate the legitimacy, range, or validity of these parameters before processing them. This lack of input validation may expose the system to unexpected behavior or potential vulnerabilities when given malicious or malformed inputs.

**lfzkoala :** The function `ComputeWithdrawals` accepts `begin` and `end` `BlockStamp` without validating if `begin` is truly before `end`.

An attacker could reverse `begin` and `end` timestamps to create an invalid range.

**Hacker007 :** The function `findWithdrawalsInRange()` is intended to find all the available `Withdrawal` between the `begin` and `end` Blockstamp.

```
func (a *Analyzer) findWithdrawalsInRange(ctx context.Context, begin, end *consensus.BlockStamp) ([]
*Withdrawal, error) {
        logger.Debugw("Finding withdrawals in slots", "begin", begin.SlotNumber, "end", end.SlotNumb
er)

        tasks := make([]runner.TaskFunc[]*Withdrawal], end.SlotNumber-begin.SlotNumber+1
//...
}
```

However, the function does not check if `begin.SlotNumber` is greater than `end.SlotNumber` plus 1, a panic raised caused by an underflow error.

Traversing the following call stack revealed no additional checks.

- func (a *Analyzer) ComputeWithdrawals()
- func (r *Reporter) BuildOracleReport()
- func (s *Service) GenerateReport()
- func (s *Service) processUpdate()
- func New(ctx context.Context, conf *Config, clients *Clients) (*Service, error)

Proof of concept

Add a test case in the withdrawals_test.go

```
                {
                        name: "Nil block1",
                        blocks: []testBlock{
                                {
                                        slotNumber: 103,
                                        data:       nil,
                                },
                                {
                                        slotNumber: 100,
                                        data:       nil,
                                },
                        },
                        expectedPrincipal: big.NewInt(0),
                        expectedReward:    big.NewInt(0),
                        expectedError:     nil,
                },
```

The test result:

```
--- FAIL: TestComputeWithdrawals (0.00s)
    --- FAIL: TestComputeWithdrawals/Nil_block1 (0.00s)
panic: runtime error: makeslice: len out of range [recovered]
        panic: runtime error: makeslice: len out of range
```

## Recommendation

**yekong :** Before processing the inputs, validate the range, format, and legitimacy of `begin` and `end`.
**lfzkoala :** Add a validation check at the beginning of the function:

```
if begin.SlotNumber > end.SlotNumber {
    return nil, nil, errors.New("Invalid range: begin should be before end")
}
```

**Hacker007 :** Check if `end.SlotNumber` is not smaller than `begin.SlotNumber`.

```
if end.SlotNumber < begin.SlotNumber {
    return []*Withdrawal{}, nil
}
```

## Client Response

Declined, These values are not given by users, but arise programatically in such a way that it is not possible for the start block to be after the end block, as the end block is computed as the start + target in the scheduler.
`Secure3 comment`: adding an extra check method is recommended for extra security.

# MTO-7:Lack of Secure Headers in HTTP Responses

| Category | Severity | Client Response | Contributor |
| --- | --- | --- | --- |
| Code Style | Medium | Declined | lfzkoala |

## Code Reference

- code/oracle/main.go#L265

```
265:w.WriteHeader(http.StatusOK)
```

## Description

**lfzkoala :** The health check server does not set any secure HTTP headers, which can protect against certain types of attacks, like Clickjacking.
**Proof of Concept**: No HTTP headers like `X-Content-Type-Options`, `X-Frame-Options`, and `Strict-Transport-Security` are set in the response.

## Recommendation

**lfzkoala :** Set secure HTTP headers in the response:

```
w.Header().Set("X-Content-Type-Options", "nosniff")
w.Header().Set("X-Frame-Options", "DENY")
w.Header().Set("Strict-Transport-Security", "max-age=63072000; includeSubDomains")
```

## Client Response

Declined, This is an internal health endpoint and does not require security headers.

# MTO-8:Resources not released after querying events

| Category | Severity | Client Response | Contributor |
| --- | --- | --- | --- |
| Logical | Low | Fixed | biakia |

## Code Reference

- code/lib/validator/sourcer/sourcer.go#L47-L71

```
47:func (c *stakingLoader) Load(ctx context.Context, start, end uint64) ([]*wrappedInitEvent, error)
{
48:     opts := &bind.FilterOpts{
49:             Context: ctx,
50:             Start:   start,
51:             End:     &end,
52:     }
53:     events, err := metrics.Instrumented("staking_event_loader_filter", nil, func() (*staking.Sta
kingValidatorInitiatedIterator, error) {
54:             return c.contract.FilterValidatorInitiated(opts, nil, nil)
55:     })
56:     if err != nil {
57:             return nil, err
58:     }
59:
60:     wrappedEvents := []*wrappedInitEvent{}
61:     for events.Next() {
62:             wrappedEvents = append(wrappedEvents, &wrappedInitEvent{events.Event})
63:     }
64:
65:     if events.Error() != nil {
66:             return nil, errors.Wrap(events.Error(), "failed to iterate over events")
67:     }
68:
69:     metrics.Count("staking_event_loaded_events", float64(len(wrappedEvents)))
70:     return wrappedEvents, nil
71:}
```

## Description

**biakia :** In `sourcer.go`, the function `Load` will query all `ValidatorInitiated` events between the given start and end block numbers. It will call the function `FilterValidatorInitiated` in `staking.go`:

```
events, err := metrics.Instrumented("staking_event_loader_filter", nil, func() (*staking.StakingVali
datorInitiatedIterator, error) {
                return c.contract.FilterValidatorInitiated(opts, nil, nil)
})
```

The `staking.go` is a generated binding contract. It provides three helper functions to query events:

```
// Next advances the iterator to the subsequent event, returning whether there
// are any more events found. In case of a retrieval or parsing error, false is
// returned and Error() can be queried for the exact failure.
func (it *StakingValidatorInitiatedIterator) Next() bool {
        // If the iterator failed, stop iterating
        if it.fail != nil {
                return false
        }
        // If the iterator completed, deliver directly whatever's available
        if it.done {
                select {
                case log := <-it.logs:
                        it.Event = new(StakingValidatorInitiated)
                        if err := it.contract.UnpackLog(it.Event, it.event, log); err != nil {
                                it.fail = err
                                return false
                        }
                        it.Event.Raw = log
                        return true

                default:
                        return false
                }
        }
        // Iterator still in progress, wait for either a data or an error event
        select {
        case log := <-it.logs:
                it.Event = new(StakingValidatorInitiated)
                if err := it.contract.UnpackLog(it.Event, it.event, log); err != nil {
                        it.fail = err
                        return false
                }
                it.Event.Raw = log
                return true

        case err := <-it.sub.Err():
                it.done = true
                it.fail = err
                return it.Next()
        }
}
```

```
// Error returns any retrieval or parsing error occurred during filtering.
func (it *StakingValidatorInitiatedIterator) Error() error {
        return it.fail
}

// Close terminates the iteration process, releasing any pending underlying
// resources.
func (it *StakingValidatorInitiatedIterator) Close() error {
        it.sub.Unsubscribe()
        return nil
}
```

The `sourcer` will call `Next()` to iterate the returned events and `Error()` to find any errors:

```
        wrappedEvents := []*wrappedInitEvent{}
        for events.Next() {
                wrappedEvents = append(wrappedEvents, &wrappedInitEvent{events.Event})
        }

        if events.Error() != nil {
                return nil, errors.Wrap(events.Error(), "failed to iterate over events")
        }
```

However, the function `Close()` is not called after all events have been handled. This means that there may be a risk of memory leakage.

## Recommendation

**biakia :** Consider closing the `StakingValidatorInitiatedIterator` after all events have been handled:

```
defer events.Close()
```

## Client Response

Fixed

# MTO-9:Not check the existence of the block returned from the Beacon.

| Category | Severity | Client Response | Contributor |
| --- | --- | --- | --- |
| Logical | Low | Fixed | biakia |

## Code Reference

- code/oracle/scheduler/scheduler.go#L355-L359

```
355:// Load the current block from the chain, and ensure that there's an execution layer block for i
t.
356:    currentBlock, err := s.consensusClient.SignedBeaconBlock(ctx, consensus.BlockIdentifierHead)
357:    if err != nil {
358:            return 0, 0, errors.Wrap(err, "failed to get current block")
359:    }
```

## Description

**biakia :** In `scheduler.go`, the function `alignBlockWindow` will call the function `SignedBeaconBlock` to get a Beacon block:

```
        // Load the current block from the chain, and ensure that there's an execution layer block f
or it.
        currentBlock, err := s.consensusClient.SignedBeaconBlock(ctx, consensus.BlockIdentifierHead)
        if err != nil {
                return 0, 0, errors.Wrap(err, "failed to get current block")
        }
```

As per the document(https://pkg.go.dev/github.com/attestantio/go-eth2-client@v0.18.3/http#Service.SignedBeaconBlock) of the function `SignedBeaconBlock`, it says `SignedBeaconBlock fetches a signed beacon block given a block ID. N.B if a signed beacon block for the block ID is not available this will return nil without an error.` However, in the above code, it only checks the returned error. When the block is not available, the `currentBlock` will be nil and the `err` will be nil too. Finally, the `if` condition will be ignored.

## Recommendation

**biakia :** Consider adding a check on the returned value:

```
// Load the current block from the chain, and ensure that there's an execution layer block for it.
        currentBlock, err := s.consensusClient.SignedBeaconBlock(ctx, consensus.BlockIdentifierHead)
        if err != nil || currentBlock == nil{
                return 0, 0, errors.Wrap(err, "failed to get current block")
        }
```

## Client Response

Fixed

# MTO-10:Incomplete nil check

| Category | Severity | Client Response | Contributor |
| --- | --- | --- | --- |
| Logical | Low | Fixed | Hacker007 |

## Code Reference

- code/lib/consensus/blockstamp.go#L91-L105
- code/lib/consensus/blockstamp.go#L115-L124

```
91:case spec.DataVersionBellatrix:
92:             if v.Bellatrix == nil || v.Bellatrix.Message.Body.ExecutionPayload == nil {
93:                     return 0, ErrInvalidBlock
94:             }
95:             return v.Bellatrix.Message.Body.ExecutionPayload.BlockNumber, nil
96:     case spec.DataVersionCapella:
97:             if v.Capella == nil || v.Capella.Message.Body.ExecutionPayload == nil {
98:                     return 0, ErrInvalidBlock
99:             }
100:            return v.Capella.Message.Body.ExecutionPayload.BlockNumber, nil
101:    case spec.DataVersionDeneb:
102:            if v.Deneb == nil || v.Deneb.Message.Body.ExecutionPayload == nil {
103:                    return 0, ErrInvalidBlock
104:            }
105:            return v.Deneb.Message.Body.ExecutionPayload.BlockNumber, nil

115:case spec.DataVersionCapella:
116:            if v.Capella == nil || v.Capella.Message.Body.ExecutionPayload == nil {
117:                    return nil, ErrInvalidBlock
118:            }
119:            return v.Capella.Message.Body.ExecutionPayload.Withdrawals, nil
120:    case spec.DataVersionDeneb:
121:            if v.Deneb == nil || v.Deneb.Message.Body.ExecutionPayload == nil {
122:                    return nil, ErrInvalidBlock
123:            }
124:            return v.Deneb.Message.Body.ExecutionPayload.Withdrawals, nil
```

## Description

**Hacker007 :** Per the Go file versionedsignedbeaconblock.go from go-eth2-client, the function Slot() will check if Phase0.Message is nil before returning the slot number

```
func (v *VersionedSignedBeaconBlock) Slot() (phase0.Slot, error) {
        switch v.Version {
        case DataVersionPhase0:
                if v.Phase0 == nil || v.Phase0.Message == nil {
                        return 0, errors.New("no phase0 block")
                }
                return v.Phase0.Message.Slot, nil
//...
    }
```

However, in the file blockstamp.go, the functions `BlockNumberFromBlock()` and `WithdrawalsFromBlock()` do not check if `Message` is nil before dereferencing it. An example.

```
func WithdrawalsFromBlock(v *spec.VersionedSignedBeaconBlock) ([]*capella.Withdrawal, error) {
        switch v.Version {
        case spec.DataVersionCapella:
                if v.Capella == nil || v.Capella.Message.Body.ExecutionPayload == nil {
                        return nil, ErrInvalidBlock
                }
```

## Recommendation

**Hacker007 :** Check if v.Capella.Message is nil before dereferencing Message. An example.

```
                if v.Capella == nil || v.Capella.Message == nil || v.Capella.Message.Body.ExecutionP
ayload == nil {
                        return nil, ErrInvalidBlock
                }
```

## Client Response

Fixed

# MTO-11:Missing parameter validation in `withdrawals::NewAnalyzer()` function

| Category | Severity | Client Response | Contributor |
| --- | --- | --- | --- |
| Logical | Low | Fixed | Hupixiong3 |

## Code Reference

- code/lib/withdrawals/withdrawals.go#L46-L51

```
46:func NewAnalyzer(client consensus.Client, maxRPS int, clReturnsReceiver common.Address) *Analyzer
{
47:     return &Analyzer{
48:             maxRPS:                  maxRPS,
49:             consensusClient:         client,
50:             clReturnsReceiverAddress: clReturnsReceiver,
51:     }
```

## Description

**Hupixiong3 :** In the NewAnalyzer function: maxRPS should be a valid positive integer: The maxRPS parameter represents the maximum requests per second and should be a positive integer. If it's not a positive integer or if it's zero or negative, it would indicate an invalid configuration.consensusClient should not be nil: The consensusClient parameter represents the consensus client used for interacting with the blockchain. If it's nil, the analyzer won't be able to perform its functions, leading to runtime errors.

## Recommendation

**Hupixiong3 :** Check if maxRPS is greater than zero and if consensusClient is not nil. If either condition is not met, an error is returned, preventing the creation of an Analyzer instance with invalid or missing parameters.

## Client Response

Fixed

# MTO-12:start-block and end-block flags should define Uint64Flag, num-runs and start-index flag should use UintFlag, Because they cannot have negative numbers

| Category | Severity | Client Response | Contributor |
| --- | --- | --- | --- |
| Logical | Low | Fixed | seek.guo |

## Code Reference

- code/oracle/main.go#L131
- code/oracle/main.go#L154
- code/oracle/main.go#L162
- code/oracle/main.go#L186
- code/oracle/main.go#L193
- code/oracle/main.go#L198
- code/oracle/main.go#L203
- code/oracle/main.go#L215
- code/oracle/scripts/verify.go#L21
- code/oracle/scripts/verify.go#L37
- code/oracle/scripts/events.go#L19
- code/oracle/scripts/events.go#L21
- code/oracle/scripts/events.go#L23
- code/oracle/service/config.go#L38
- code/oracle/service/config.go#L39
- code/oracle/service/config.go#L43

```
19:func LoadEvents(conf *service.Config, startBlock, endBlock uint64, numRuns int) error {

21:func VerifyReports(srv *service.Service, startIndex int, quietLogging bool) error {

21:wg.Add(numRuns)

23:for i := 0; i < numRuns; i++ {

37:for i := startIndex; i < int(numRecords.Int64()); i++ {

38:PrometheusPort:              c.Int("prometheus-port"),

39:HealthCheckPort:             c.Int("health-check-port"),

43:ChainID:                     c.Int("chain-id"),

131:&cli.IntFlag{

154:startBlock := uint64(c.Int("start-block"))

162:&cli.IntFlag{

186:return scripts.VerifyReports(srv, c.Int("start-index"), c.Bool("quiet-logging"))

193:&cli.IntFlag{

198:&cli.IntFlag{

203:&cli.IntFlag{

215:return scripts.LoadEvents(conf, uint64(c.Int("start-block")), uint64(c.Int("end-block")), c.Int
("num-runs"))
```

## Description

**seek.guo :** start-block and end-block flags should define Uint64Flag, num-runs， start-index， port and chainId flag should use UintFlag, Because they cannot have negative numbers

## Recommendation

**seek.guo :** code can change as follow

```
        Commands: []*cli.Command{
                        {
                                Name:  "regenerate-reports",
                                Usage: "Regenerate reports for a given range of records",
                                Flags: []cli.Flag{
                                        &cli.Uint64Flag{
                                                Name:     "start-block",
                                                Usage:    "The first block that reports should be ge
nerated from. This should correspond to the start block of the first report.",
                                                Required: true,
                                        },
                                        &cli.StringFlag{
                                                Name:  "output-file",
                                                Usage: "The file to write the output to. If not spec
ified, the output will be written to stdout.",
                                        },
                                },
                                Action: func(c *cli.Context) error {
                                        conf, err := service.ParseConfig(c)
                                        if err != nil {
                                                return err
                                        }
                                        clients, err := service.NewClients(context.Background(), con
f)
                                        if err != nil {
                                                return err
                                        }
                                        srv, err := service.New(context.Background(), conf, clients)
                                        if err != nil {
                                                return err
                                        }
                                        startBlock := c.Uint64("start-block")
                                        return scripts.RegenerateReports(srv, startBlock, c.String
("output-file"))
                                },
                        },
                        {
                                Name:  "verify-reports",
                                Usage: "Verify the integrity of the oracle reports",
                                Flags: []cli.Flag{
                                        &cli.UintFlag{
                                                Name:     "start-index",
```

```
Usage:    "The first report to start checking from.",
                                        Required: true,
                                },
                                &cli.BoolFlag{
                                        Name:  "quiet-logging",
                                        Usage: "Only log results rather than the full logs o
f the process.",
                                        Value: true,
                                },
                        },
                        Action: func(c *cli.Context) error {
                                conf, err := service.ParseConfig(c)
                                if err != nil {
                                        return err
                                }
                                clients, err := service.NewClients(context.Background(), con
f)
                                if err != nil {
                                        return err
                                }
                                srv, err := service.New(context.Background(), conf, clients)
                                if err != nil {
                                        return err
                                }
                                return scripts.VerifyReports(srv, c.Uint("start-index"), c.B
ool("quiet-logging"))
                        },
                },
                {
                        Name:  "load-events",
                        Usage: "Load validator initiation events for a block range",
                        Flags: []cli.Flag{
                                &cli.Uint64Flag{
                                        Name:     "start-block",
                                        Usage:    "The first block to load events from (incl
usive)",
                                        Required: true,
                                },
                                &cli.Uint64Flag{
                                        Name:     "end-block",
                                        Usage:    "The last block to load events
```

```
from (inclusive)",
                                                Required: true,
                                        },
                                        &cli.UintFlag{
                                                Name:     "num-runs",
                                                Required: false,
                                                Value:    1,
                                        },
                                },
                                Action: func(c *cli.Context) error {
                                        conf, err := service.ParseConfig(c)
                                        if err != nil {
                                                return err
                                        }

                                        return scripts.LoadEvents(conf, uint64(c.Int("start-bloc
k")), uint64(c.Int("end-block")), c.Uint("num-runs"))
                                },
                        },
```

```
func LoadEvents(conf *service.Config, startBlock, endBlock uint64, numRuns uint) error {
        wg.Add(int(numRuns))
        for i := 0; i < int(numRuns); i++ {
                ....
        }
        ....
}
```

```
func VerifyReports(srv *service.Service, startIndex uint, quietLogging bool) error {
...
    for i := int64(startIndex); i < numRecords.Int64(); i++ {
        ....
     }
}
```

## Client Response

Fixed

# MTO-13:Uncontrolled Public Key Input in ValidatorsAt

| Category | Severity | Client Response | Contributor |
| --- | --- | --- | --- |
| Logical | Low | Declined | lfzkoala |

## Code Reference

- code/lib/validator/loader.go#L60-L62

```
60:valSet, err := metrics.Instrumented("load_validator_set", nil, func() (sourcer.ValidatorSet, erro
r) {
61:             return l.sourcer.ValidatorSetAt(ctx, bs.BlockNumber)
62:     }, metrics.WithBuckets(metrics.TimingBucketsLong))
```

## Description

**lfzkoala :** The function `ValidatorsAt` processes a list of public keys sourced from a third party (`l.sourcer.ValidatorSetAt`). If the source is compromised or the public keys are not verified elsewhere, this could lead to data contamination or unexpected behavior. The code only checks the length of the public key.

If an attacker can influence the list of public keys returned by the `l.sourcer.ValidatorSetAt`, they might cause the system to process invalid or malicious public keys.

## Recommendation

**lfzkoala :** Ensure that public keys are validated before use. Implementing cryptographic checks or validation against a whitelist of trusted public keys can be a safeguard.

## Client Response

Declined,The data is sourced from the blockchain and cannot be manipulated.

`Secure3 comment`:The chance of this happening is low meanwhile ensuring the credibility of data sources remains important.

# MTO-14:No further error handling in `scheduler::checkForUpdate()` function

| Category | Severity | Client Response | Contributor |
| --- | --- | --- | --- |
| Logical | Low | Declined | Hupixiong3 |

## Code Reference

- code/oracle/scheduler/scheduler.go#L275-L281

```
275:lastRecord, err := s.oracleContract.LatestRecord(&bind.CallOpts{
276:            Context: ctx,
277:    })
278:    if err != nil {
279:            logger.Errorw("Error loading latest record", "error", err)
280:            return
281:    }
```

## Description

**Hupixiong3 :**

```
        lastRecord, err := s.oracleContract.LatestRecord(&bind.CallOpts{
                Context: ctx,
        })
        if err != nil {
                logger.Errorw("Error loading latest record", "error", err)
                return
        }
```

when an error occurs during the invocation of the s.oracleContract.LatestRecord contract method, the error message is logged, but there is no further error handling.

## Recommendation

**Hupixiong3 :** Retry Operation: Some contract call errors may be due to transient issues. You can choose to retry the operation within a certain time frame to succeed on the next attempt. In this case, be cautious in handling retry logic to avoid infinite loops. Return Default or Backup Data: If it's not possible to retrieve the latest record, you can return default values or backup data to ensure the application continues running. This is applicable in situations where a failure in the contract call won't disrupt the core functionality. Report Error and Exit: In some cases, if it's not possible to retrieve the latest record, and there's no reasonable alternative, you may choose to report the error and exit the application.

## Client Response

Declined,The proposed fix is already implemented - the service will retry the operation on the next scheduled tick.
`Secure3 comment`:The issue is valid and fixed. So we think the question is valid.

# MTO-15:Optimization for the binary search

| Category | Severity | Client Response | Contributor |
| --- | --- | --- | --- |
| Logical | Informational | Fixed | biakia, alansh |

## Code Reference

- code/lib/consensus/finder/finder.go#L78

```
78:func (f *BinarySearch) BlockStamp(ctx context.Context, blockNumber uint64) (*consensus.BlockStamp, error) {
```

## Description

**biakia :** The `finder.go` provides a function `BlockStamp` to obtain consensus layer information. It will use a binary search. The `slotLowerBound` is the lower bound of the slot range and the `consensus.BlockIdentifierHead` is the upper bound of the slot range. This function will check if the given block is out of the upper bound:

```
block, err := f.consensusClient.SignedBeaconBlock(ctx, consensus.BlockIdentifierHead)
        if err != nil || block == nil {
                return nil, errors.Wrap(err, "failed to obtain the head block")
        }

        bn, err := consensus.BlockNumberFromBlock(block)
        if err != nil {
                return nil, errors.Wrap(err, "failed to obtain block number from head block")
        }
        if blockNumber == bn {
                return consensus.NewFromSignedBeaconBlock(block)
        }
        if bn < blockNumber {
                return nil, errors.Errorf("latest canonical head block's number is %d but trying to retrieve %d", bn, blockNumber)
        }
```

However, it will not check if the given block is out of the lower bound. It means if the `blockNumber` is less than the lower bound, the binary search will still be applied.

**alansh :** Currently this function tries to find the block by binary search.

But it can do better, since empty slots are quite few compared with non-empty slots.

For example, suppose we're searching for block number 100, and the high block number of slot `S` is 300, we can search slot `S-200` directly, this will converge much faster than binary search, `O(1)` vs `O(lg n)`.

## Recommendation

**biakia :** Consider checking the lower bound in function `BlockStamp`:

```
        l := f.slotLowerBound
        lowerBlock,err:=f.consensusClient.SignedBeaconBlock(ctx, f.slotLowerBound)
        if err != nil || lowerBlock == nil {
                return nil, errors.Wrap(err, "failed to obtain the lower block")
        }
        lbn, err := consensus.BlockNumberFromBlock(lowerBlock)
        if err != nil {
                return nil, errors.Wrap(err, "failed to obtain block number from lower block")
        }
        if blockNumber == lbn {
                return consensus.NewFromSignedBeaconBlock(lowerBlock)
        }
        if lbn > blockNumber {
                return nil, errors.Errorf("lower block's number is %d but trying to retrieve %d", lb
n, blockNumber)
        }
```

**alansh :** Apply the above idea for `O(1)` convergence.

## Client Response

Fixed,Cannot happen in normal operation (lower bound is given by the init block number from the contract), but done for completeness

# MTO-16:wrong error info in `Reporter::BuildOracleReport` function

| Category | Severity | Client Response | Contributor |
| --- | --- | --- | --- |
| Logical | Informational | Fixed | alansh |

## Code Reference

- code/oracle/reporter/reporter.go#L133

```
133:return nil, errors.Wrapf(err, "failed to load validators at start block %d", startBlock)
```

## Description

**alansh :**

```
        previousEndBlockVals, err := r.validatorLoader.ValidatorsAt(ctx, previousEndBlockStamp)
        if err != nil {
                return nil, errors.Wrapf(err, "failed to load validators at start block %d", startBl
ock)
        }
```

The block corresponding to `previousEndBlockStamp` is not `startBlock` but `lastRecord.UpdateEndBlock`

## Recommendation

**alansh :**

```
        previousEndBlockVals, err := r.validatorLoader.ValidatorsAt(ctx, previousEndBlockStamp)
        if err != nil {
                return nil, errors.Wrapf(err, "failed to load validators at previous end block %d",
lastRecord.UpdateEndBlock)
        }
```

## Client Response

Fixed

# MTO-17:real reason for error from `consensusClient` is unexpectedly ignored in `BinarySearch::BlockStamp()` function

| Category | Severity | Client Response | Contributor |
| --- | --- | --- | --- |
| Code Style | Informational | Fixed | alansh |

## Code Reference

- code/lib/consensus/finder/finder.go#L116

```
116:return nil, &ErrorBlockNotFound{BlockNumber: blockNumber}
```

## Description

**alansh :**

```
                        midBlock, err = f.consensusClient.SignedBeaconBlock(ctx, fmt.Sprintf("%d",
m))
                        if err != nil {
                                return nil, &ErrorBlockNotFound{BlockNumber: blockNumber}
                        }
```

Here the real reason for error from `consensusClient` is replaced with `ErrorBlockNotFound`, it may not be good for diagnosing the system from log.

## Recommendation

**alansh :** Should not hide the transient error if possible.

## Client Response

Fixed

# MTO-18:Incomplete Configuration Checks

| Category | Severity | Client Response | Contributor |
| --- | --- | --- | --- |
| Logical | Informational | Fixed | lfzkoala |

## Code Reference

- code/oracle/service/config.go#L52-L65

```
52:if c.String("service-private-key") != "" {
53:             conf.SigningMode = transaction.SigningModeLocal
54:             pk, err := crypto.HexToECDSA(c.String("service-private-key"))
55:             if err != nil {
56:                     return nil, errors.Wrap(err, "failed to parse private key")
57:             }
58:             conf.ReporterPrivateKey = pk
59:     }
60:
61:     if c.String("kms-signing-key") != "" && c.String("kms-account-json") != "" {
62:             conf.SigningMode = transaction.SigningModeRemote
63:             conf.KMSSigningKey = c.String("kms-signing-key")
64:             conf.KMSAccountJSON = c.String("kms-account-json")
65:     }
```

## Description

**lfzkoala :** The code checks for the existence of the `service-private-key` and `kms-signing-key` & `kms-account-json`. However, it doesn't explicitly handle cases where neither of them is provided. This might lead to a misconfigured service running without the capability to sign transactions or interact securely.

Run the application without providing either of the key configurations and observe potential issues.

## Recommendation

**lfzkoala :** Add an explicit check after these conditions to verify that at least one signing mechanism is provided and properly configured. If neither is available, the application should raise an error and prevent further execution.

## Client Response

Fixed

# MTO-19:gc optimization for command `regenerate-reports` and `verify-reports`

| Category | Severity | Client Response | Contributor |
| --- | --- | --- | --- |
| Logical | Informational | Acknowledged | biakia |

## Code Reference

- code/oracle/main.go#L127-L188

```
127:{
128:                            Name:  "regenerate-reports",
129:                            Usage: "Regenerate reports for a given range of records",
130:                            Flags: []cli.Flag{
131:                                    &cli.IntFlag{
132:                                            Name:     "start-block",
133:                                            Usage:    "The first block that reports should be ge
nerated from. This should correspond to the start block of the first report.",
134:                                            Required: true,
135:                                    },
136:                                    &cli.StringFlag{
137:                                            Name:  "output-file",
138:                                            Usage: "The file to write the output to. If not spec
ified, the output will be written to stdout.",
139:                                    },
140:                            },
141:                            Action: func(c *cli.Context) error {
142:                                    conf, err := service.ParseConfig(c)
143:                                    if err != nil {
144:                                            return err
145:                                    }
146:                                    clients, err := service.NewClients(context.Background(), con
f)
147:                                    if err != nil {
148:                                            return err
149:                                    }
150:                                    srv, err := service.New(context.Background(), conf, clients)
151:                                    if err != nil {
152:                                            return err
153:                                    }
154:                                    startBlock := uint64(c.Int("start-block"))
155:                                    return scripts.RegenerateReports(srv, startBlock, c.String
("output-file"))
156:                            },
157:                    },
158:                    {
159:                            Name:  "verify-reports",
160:                            Usage: "Verify the integrity of the oracle reports",
161:                            Flags: []cli.Flag{
162:                                    &cli.IntFlag{
163:                                            Name:     "start-index",
```

```
164:                                            Usage:    "The first report to start checking fro
m.",
165:                                            Required: true,
166:                                    },
167:                                    &cli.BoolFlag{
168:                                            Name:  "quiet-logging",
169:                                            Usage: "Only log results rather than the full logs o
f the process.",
170:                                            Value: true,
171:                                    },
172:                            },
173:                            Action: func(c *cli.Context) error {
174:                                    conf, err := service.ParseConfig(c)
175:                                    if err != nil {
176:                                            return err
177:                                    }
178:                                    clients, err := service.NewClients(context.Background(), con
f)
179:                                    if err != nil {
180:                                            return err
181:                                    }
182:                                    srv, err := service.New(context.Background(), conf, clients)
183:                                    if err != nil {
184:                                            return err
185:                                    }
186:                                    return scripts.VerifyReports(srv, c.Int("start-index"), c.Bo
ol("quiet-logging"))
187:                            },
188:                    },
```

## Description

**biakia :** In `main.go`, there are two commands which will create a local service:

```
{
                                Name:  "regenerate-reports",
                                Usage: "Regenerate reports for a given range of records",
                                Flags: []cli.Flag{
                                        &cli.IntFlag{
                                                Name:     "start-block",
                                                Usage:    "The first block that reports should be ge
nerated from. This should correspond to the start block of the first report.",
                                                Required: true,
                                        },
                                        &cli.StringFlag{
                                                Name:  "output-file",
                                                Usage: "The file to write the output to. If not spec
ified, the output will be written to stdout.",
                                        },
                                },
                                Action: func(c *cli.Context) error {
                                        conf, err := service.ParseConfig(c)
                                        if err != nil {
                                                return err
                                        }
                                        clients, err := service.NewClients(context.Background(), con
f)
                                        if err != nil {
                                                return err
                                        }
                                        srv, err := service.New(context.Background(), conf, clients)
                                        if err != nil {
                                                return err
                                        }
                                        startBlock := uint64(c.Int("start-block"))
                                        return scripts.RegenerateReports(srv, startBlock, c.String
("output-file"))
                                },
                        },
                        {
                                Name:  "verify-reports",
                                Usage: "Verify the integrity of the oracle reports",
                                Flags: []cli.Flag{
                                        &cli.IntFlag{
                                                Name:     "start-index",
                                                Usage:    "The first report to start chec-
```

```
king from.",
                                        Required: true,
                                },
                                &cli.BoolFlag{
                                        Name:  "quiet-logging",
                                        Usage: "Only log results rather than the full logs o
f the process.",
                                        Value: true,
                                },
                        },
                        Action: func(c *cli.Context) error {
                                conf, err := service.ParseConfig(c)
                                if err != nil {
                                        return err
                                }
                                clients, err := service.NewClients(context.Background(), con
f)
                                if err != nil {
                                        return err
                                }
                                srv, err := service.New(context.Background(), conf, clients)
                                if err != nil {
                                        return err
                                }
                                return scripts.VerifyReports(srv, c.Int("start-index"), c.Bo
ol("quiet-logging"))
                        },
                },
```

Although golang has a gc mechanism to ensure that the `srv` is destoried, we still recommend calling the `Stop()` function to shut down the service after using it.

## Recommendation

**biakia :** Consider calling the `Stop()` function in command `regenerate-reports` and `verify-reports`.

## Client Response

Acknowledged,irrelevant as scripts are not long running services

# MTO-20:Function Name Mismatch

| Category | Severity | Client Response | Contributor |
| --- | --- | --- | --- |
| Code Style | Informational | Fixed | yekong |

## Code Reference

- code/lib/validator/keyset.go#L15-L22

```
15:// NewPublicKeySetFromValidators creates a new public key set from a list of validators.
16:func NewPublicKeySet[T Keyer](keys []T) PublicKeySet {
17:     set := PublicKeySet{}
18:     for _, v := range keys {
19:             set.Add(v.Key())
20:     }
21:     return set
22:}
```

## Description

**yekong :** In other functions in the file, comments and function names remain consistent. The function is declared as NewPublicKeySetFromValidators, but it's implemented as NewPublicKeySet.

## Recommendation

**yekong :** Keep comments and function names consistent

## Client Response

Fixed

# MTO-21:Inefficient loop statement in `withdrawals.go`

| Category | Severity | Client Response | Contributor |
| --- | --- | --- | --- |
| Gas Optimization | Informational | Acknowledged | Hacker007 |

## Code Reference

- code/lib/withdrawals/withdrawals.go#L169-L180

```
169:slot, err := block.Slot()
170:            if err != nil {
171:                    return nil, errors.Wrapf(err, "failed to get slot from block %s", block.Stri
ng())
172:            }
173:            withdrawals = append(withdrawals, &Withdrawal{
174:                    ValidatorIndex: uint64(w.ValidatorIndex),
175:                    AmountWei:      consensus.ConvertGweiToWei(uint64(w.Amount)),
176:                    Address:        common.BytesToAddress(w.Address[:]),
177:                    Slot:           uint64(slot),
178:            })
179:    }
180:    return withdrawals, nil
```

## Description

**Hacker007 :** Since the ws are requested from the same block, the slot number will be the same value, thus there is no need to request a slot in every iteration.

```
        for _, w := range ws {
                slot, err := block.Slot()
                if err != nil {
                        return nil, errors.Wrapf(err, "failed to get slot from block %s", block.Stri
ng())
                }
                withdrawals = append(withdrawals, &Withdrawal{
                        ValidatorIndex: uint64(w.ValidatorIndex),
                        AmountWei:      consensus.ConvertGweiToWei(uint64(w.Amount)),
                        Address:        common.BytesToAddress(w.Address[:]),
                        Slot:           uint64(slot),
                })
        }
```

## Recommendation

**Hacker007 :** Place the getting slot code before the loop statement.

```
        slot, err := block.Slot()
        if err != nil {
            return nil, errors.Wrapf(err, "failed to get slot from block %s", block.String())
       }
        for _, w := range ws {
                withdrawals = append(withdrawals, &Withdrawal{
                        ValidatorIndex: uint64(w.ValidatorIndex),
                        AmountWei:      consensus.ConvertGweiToWei(uint64(w.Amount)),
                        Address:        common.BytesToAddress(w.Address[:]),
                        Slot:           uint64(slot),
                })
        }
```

## Client Response

Acknowledged,We are not in the business of shaving nanoseconds off a multiple-minute process

# MTO-22:Unused function parameter `ctx`

| Category | Severity | Client Response | Contributor |
| --- | --- | --- | --- |
| Language Specific | Informational | Fixed | Hacker007 |

## Code Reference

- code/oracle/reporter/reporter.go#L208
- code/oracle/reporter/reporter.go#L216
- code/oracle/reporter/reporter.go#L226

```
208:func (r *Reporter) computeNumValidatorsFullyWithdrawn(ctx context.Context, endVals []*validator.
Validator) uint64 {

216:func (r *Reporter) computeCurrentTotalValidatorBalance(ctx context.Context, validatorsWithBalanc
e []*validator.Validator) (*big.Int, error) {

226:func (r *Reporter) computeProcessedDepositAmount(ctx context.Context, beginVals, endVals []*vali
dator.Validator) (*big.Int, error) {
```

## Description

**Hacker007 :** The function parameter `ctx` is declared but never used in the following functions.

- computeProcessedDepositAmount()
- computeCurrentTotalValidatorBalance()
- computeNumValidatorsFullyWithdrawn()

## Recommendation

**Hacker007 :** Remove the unused parameter in the aforementioned functions.

## Client Response

Fixed

# MTO-23:Inefficiency in Memory Allocation in `keyset::PublicKeys()` function

| Category | Severity | Client Response | Contributor |
| --- | --- | --- | --- |
| Language Specific | Informational | Acknowledged | Hupixiong3 |

## Code Reference

- code/lib/validator/keyset.go#L25-L31

```
25:func (s PublicKeySet) PublicKeys() []phase0.BLSPubKey {
26:     var keys []phase0.BLSPubKey
27:     for k := range s {
28:             keys = append(keys, k)
29:     }
30:     return keys
31:}
```

## Description

**Hupixiong3 :** The function PublicKeys dynamically appends keys to the slice without preallocating space. If the set is large, this can cause multiple memory reallocations.

## Recommendation

**Hupixiong3 :** Preallocate space for the slice: `keys := make([]phase0.BLSPubKey, 0, len(s))`

## Client Response

Acknowledged,We prefer to be consistent with our general style rather than save a few small allocations

# MTO-24:Unused field `Epoch`

| Category | Severity | Client Response | Contributor |
| --- | --- | --- | --- |
| Language Specific | Informational | Fixed | Hacker007 |

## Code Reference

- code/lib/withdrawals/withdrawals.go#L22-L28

```
22:type Withdrawal struct {
23:     Epoch          uint64
24:     Slot           uint64
25:     ValidatorIndex uint64
26:     AmountWei      *big.Int
27:     Address        common.Address
28:}
```

## Description

**Hacker007 :** The field `Epoch` is declared in the struct `Withdrawal` but never used in the codebase.

```
type Withdrawal struct {
        Epoch          uint64
        Slot           uint64
        ValidatorIndex uint64
        AmountWei      *big.Int
        Address        common.Address
}
```

## Recommendation

**Hacker007 :** Remove the redundant field `Epoch`.

## Client Response

Fixed

# Disclaimer

This report is subject to the terms and conditions (including without limitation, description of services, confidentiality, disclaimer and limitation of liability) set forth in the Invoices, or the scope of services, and terms and conditions provided to you (“Customer” or the “Company”) in connection with the Invoice. This report provided in connection with the services set forth in the Invoices shall be used by the Company only to the extent permitted under the terms and conditions set forth in the Invoice. This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes, nor may copies be delivered to any other person other than the Company, without Secure3’s prior written consent in each instance.

This report is not an “endorsement” or “disapproval” of any particular project or team. This report is not an indication of the economics or value of any “product” or “asset” created by any team or project that contracts Secure3 to perform a security assessment. This report does not provide any warranty or guarantee of free of bug of codes analyzed, nor do they provide any indication of the technologies, business model or legal compliancy.

This report should not be used in any way to make decisions around investment or involvement with any particular project. Instead, it represents an extensive assessing process intending to help our customers increase the quality of their code and high-level consistency of implementation and business model, while reducing the risk presented by cryptographic tokens and blockchain technology.

Secure3’s position on the final decisions over blockchain technologies and corresponding associated transactions is that each company and individual are responsible for their own due diligence and continuous security.

The assessment services provided by Secure3 is subject to dependencies and under continuing development. The assessment reports could include false positives, false negatives, and other unpredictable results. The services may access, and depend upon, multiple layers of third-parties.