Go-Helios

BLOCKCHAIN AUDIT REPORT

January 2025

ExVul

www.exvul.com

Table of Contents

1. EXECUTIVE SUMMARY

Exvul Web3 Security was engaged by go-helios to review Blockchain implementation. The assessment was conducted in accordance with our systematic approach to evaluate potential security issues based upon customer requirement. The report provides detailed recommendations to resolve the issue and provide additional suggestions or recommendations for improvement.

The outcome of the assessment outlined in chapter 3 provides the system's owners a full description of the vulnerabilities identified, the associated risk rating for each vulnerability, and detailed recommendations that will resolve the underlying technical issue.

1.1 Methodology

To standardize the evaluation, we define the following terminology based on OWASP Risk Rating Methodology [10] which is the gold standard in risk assessment using the following risk models:

- Likelihood: represents how likely a particular vulnerability is to be uncovered and exploited in the wild.
- Impact: measures the technical loss and business damage of a successful attack.
- Severity: determine the overall criticality of the risk.

Likelihood can be: High, Medium and Low and impact are categorized into for: High, Medium, Low, Informational. Severity is determined by likelihood and impact and can be classified into five categories accordingly, Critical, High, Medium, Low, Informational shown in table 1.1.

Likelihood				
High	*Informational*	Medium	High	Critical
Medium	*Informational*	Low	Medium	High
Low	*Informational*	Low	Low	Medium
	Informational	*Low*	*Medium*	*High*
		IMPACT		

Table 1.1 Overall Risk Severity

To evaluate the risk, we will be going through a list of items, and each would be labelled with a severity category. The audit was performed with a systematic approach guided by a comprehensive assessment list carefully designed to identify known and impactful security issues. If our tool or analysis does not identify any issue, the contract can be considered safe regarding the assessed item. For any discovered issue, we might further deploy contracts on our private test environment and run tests to confirm the findings. If necessary, we would additionally build a PoC to demonstrate the possibility of exploitation. The concrete list of check items is shown in Table 1.2.

- Basic Coding Bugs: We first statically analyze given Blockchain with our proprietary static code analyzer for known coding bugs, and then manually verify (reject or confirm) all the issues found by our tool.
- Code and business security testing: We further review business logics, examine system operations, and place DeFi-related aspects under scrutiny to uncover possible pitfalls and/or bugs.
- Additional Recommendations: We also provide additional suggestions regarding the coding and development of Blockchains from the perspective of proven programming practices.

Category	Assessment Item
P2P Communication Security	Connection Number Occupation Audit
	Eclipse Attack
	Packet Size Limit
	Node Communication Protocol Security
RPC Interface Security	RPC Sensitive Interface Permissions
	Traditional Web Security
	RPC Interface Security
Consensus Mechanism Security	Design Of Consensus Mechanism
	Implementation Of Consensus Verification
	Incentive Mechanism Audit
Transaction processing Security	Transaction Signature Logic
	Transaction Verification Logic
	Transaction Processing Logic
	Transaction Fee Setting
	Transaction Replay
Cryptography Security	Random Number Range And Probability Distribution
	Cryptographic Algorithm Lmplementation/Use
Wallet Module & Account Security Audit	Private Key / Mnemonic Word Storage Security
	Private Key / Mnemonic Word Usage Security
	Private key/mnemonic generation algorithm
Others Security Audit	Database Security
	Thread Security
	File Permission Security
	Historical Vulnerability Security

Table 1.2: The Full List of Assessment Items

To better describe each issue we identified, we categorize the findings with Common Weakness Enumeration (CWE-699) [14], which is a community-developed list of software weakness types to better delineate and organize weaknesses around concepts frequently encountered in software development.

2. FINDINGS OVERVIEW

2.1 Project Info

Project Name: go-helios

Audit Time: December 12, 2024 – January 24, 2025

Language: go

File Name	HASH
go-helios	https://github.com/unicornultrafoundation/go-helios/commit/672f91b5a0859de7146cdc8c226d14a9b8cf1cfe

2.2 Summary

Severity	Found	
Critical	4	🟥🟥🟥🟥
High	1	🟥
Medium	3	🟨🟨🟨
Low	3	🟦🟦🟦
Informational	1	🟩

2.3 Key Findings

ID	Severity	Findings Title	Status	Confirm
NVE-001	Critical	Missing Negative Check on memSize in readerLoop	Fixed	Confirmed
NVE-002	Critical	Unchecked Negative Input for SetWriteCount	Fixed	Confirmed
NVE-003	Critical	Overflow may occur during CountByIdx	Fixed	Confirmed
NVE-004	Critical	Negative maxSize in Resize Function Leads to DoS	Fixed	Confirmed
NVE-005	High	Should Add Thread Lock	Fixed	Confirmed
NVE-006	Medium	Missing peerID Validation in Loop Execution	Fixed	Confirmed
NVE-007	Medium	Potential Integer Overflow in Function tryToSync	Fixed	Confirmed
NVE-008	Medium	Path Traversal in OpenDB Function	Fixed	Confirmed
NVE-009	Low	Use of Deprecated ioutil.ReadDir Function	Fixed	Confirmed
NVE-010	Low	Unverified Return Value in MigrateTables Function	Fixed	Confirmed
NVE-011	Low	Inefficient UniqKey Check Method	Acknowledged	Confirmed
NVE-012	Informational	govulncheck result	Fixed	Confirmed

Table 2.3: Key Audit Findings

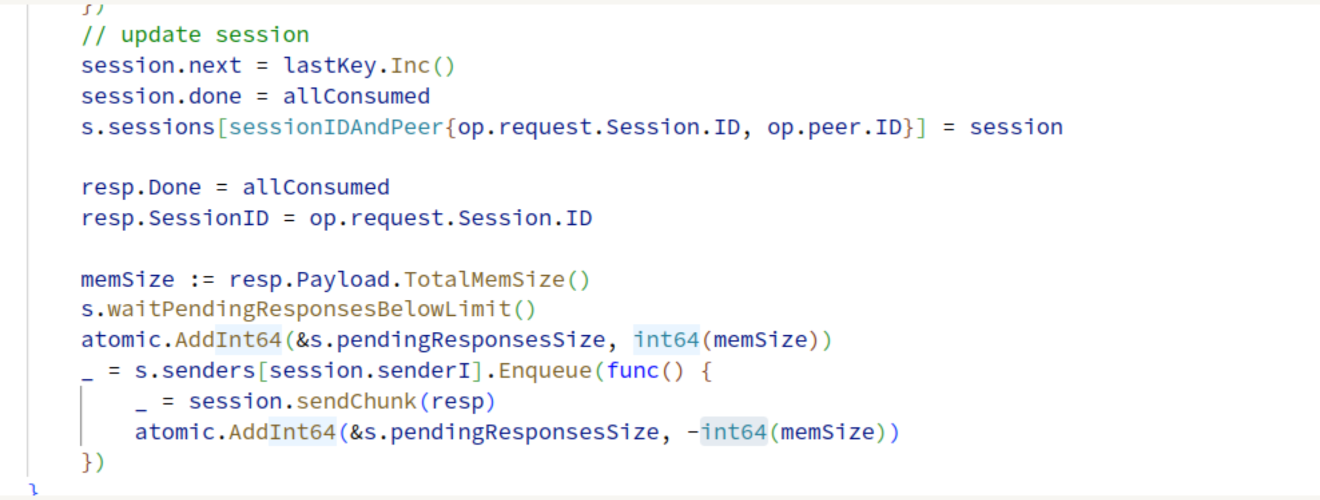
3. DETAILED DESCRIPTION OF FINDINGS

3.1 Missing Negative Check on memSize in readerLoop

ID:	NVE-001	**Location:**	gossip/basestream/basestreamseeder/seeder.go
Severity:	Critical	**Category:**	P2P Communication Security
Likelihood:	Medium	**Impact:**	High

Description:

In the readerLoop function, the final memSize is not checked to see if it's less than 0..



```
    })
    // update session
    session.next = lastKey.Inc()
    session.done = allConsumed
    s.sessions[sessionIDAndPeer{op.request.Session.ID, op.peer.ID}] = session

    resp.Done = allConsumed
    resp.SessionID = op.request.Session.ID

    memSize := resp.Payload.TotalMemSize()
    s.waitPendingResponsesBelowLimit()
    atomic.AddInt64(&s.pendingResponsesSize, int64(memSize))
    _ = s.senders[session.senderI].Enqueue(func() {
        _ = session.sendChunk(resp)
        atomic.AddInt64(&s.pendingResponsesSize, -int64(memSize))
    })
}
```

Result: Confirmed

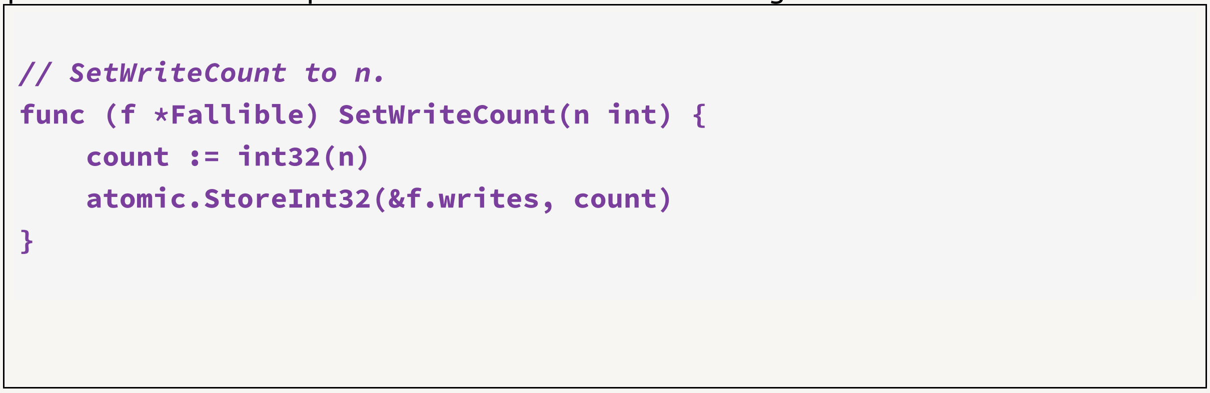
Fix Result: Fixed

3.2 Unchecked Negative Input for SetWriteCount

ID:	NVE-002	**Location:**	u2udb/fallible/fallible.go
Severity:	Critical	**Category:**	Transaction processing Security
Likelihood:	Medium	**Impact:**	High

Description:

In the SetWriteCount function, there is no check to verify if the parameter 'n' is greater than 0. Passing a negative number leads to the underflow of the 'writes' counter, causing subsequent operations like Put to panic due to erroneous state management.



```
// SetWriteCount to n.
func (f *Fallible) SetWriteCount(n int) {
    count := int32(n)
    atomic.StoreInt32(&f.writes, count)
}
```

Result: Confirmed

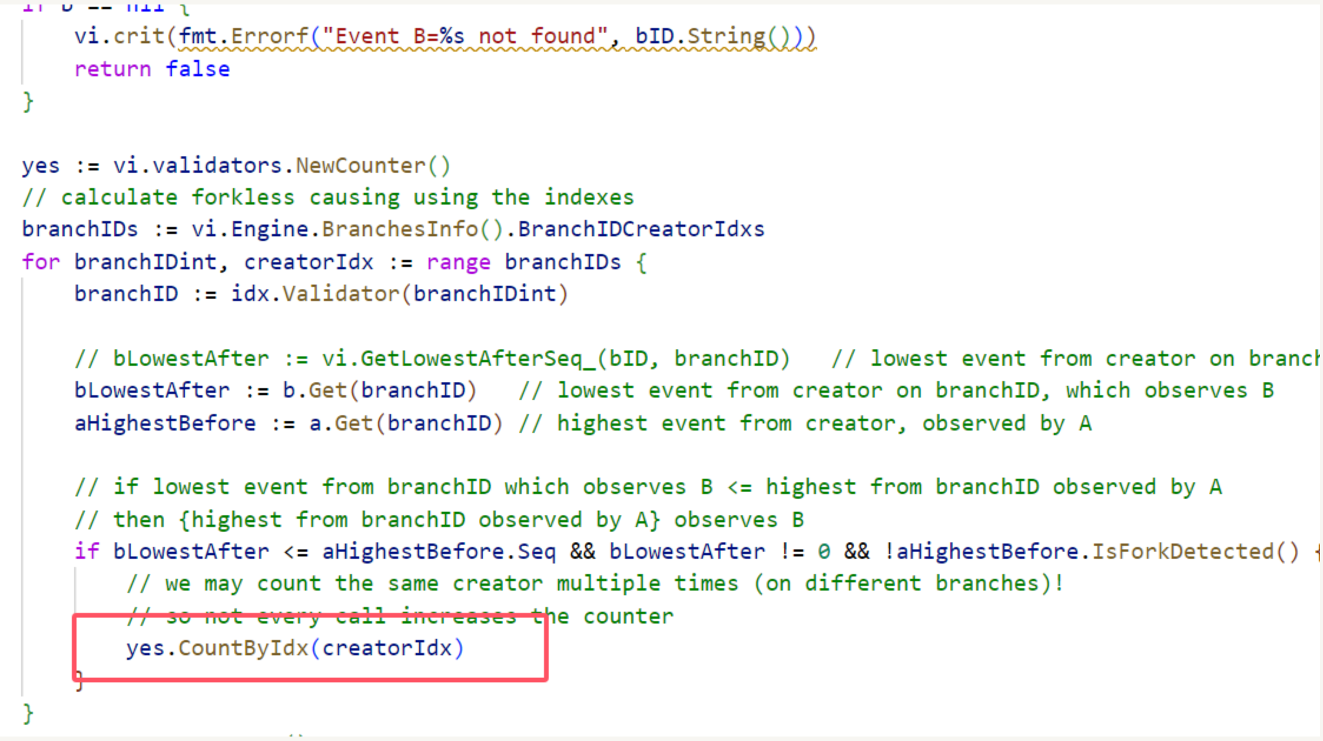
Fix Result: Fixed

3.3 Overflow may occur during CountByIdx

ID:	NVE-003	**Location:**	native/pos/stake.go
Severity:	Critical	**Category:**	Transaction processing Security
Likelihood:	Medium	**Impact:**	Medium

Description:

In the file vecfc/forkless_cause.go, the ForklessCause function uses CountByIdx which can lead to overflow.



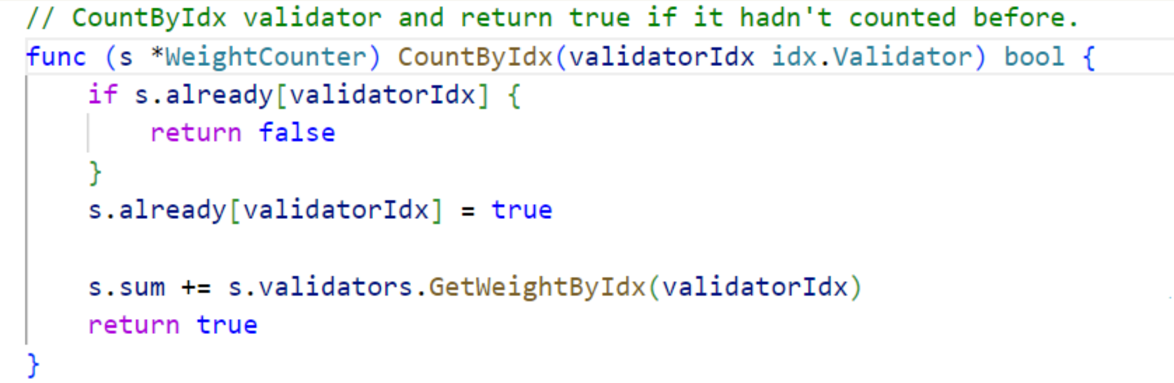
```
if b == nil {
    vi.crit(fmt.Errorf("Event B=%s not found", bID.String()))
    return false
}

yes := vi.validators.NewCounter()
// calculate forkless causing using the indexes
branchIDs := vi.Engine.BranchesInfo().BranchIDCreatorIdxs
for branchIDint, creatorIdx := range branchIDs {
    branchID := idx.Validator(branchIDint)

    // bLowestAfter := vi.GetLowestAfterSeq_(bID, branchID)   // lowest event from creator on branch
    bLowestAfter := b.Get(branchID)   // lowest event from creator on branchID, which observes B
    aHighestBefore := a.Get(branchID) // highest event from creator, observed by A

    // if lowest event from branchID which observes B <= highest from branchID observed by A
    // then {highest from branchID observed by A} observes B
    if bLowestAfter <= aHighestBefore.Seq && bLowestAfter != 0 && !aHighestBefore.IsForkDetected() {
        // we may count the same creator multiple times (on different branches)!
        // so not every call increases the counter
        yes.CountByIdx(creatorIdx)
    }
}
```

And there is no check in the function `CountByIdx`.



```
// CountByIdx validator and return true if it hadn't counted before.
func (s *WeightCounter) CountByIdx(validatorIdx idx.Validator) bool {
    if s.already[validatorIdx] {
        return false
    }
    s.already[validatorIdx] = true

    s.sum += s.validators.GetWeightByIdx(validatorIdx)
    return true
}
```

Result: Confirmed

Fix Result: Fixed

3.4 Negative maxSize in Resize Function Leads to DoS

ID:	NVE-004	**Location:**	utils/simplewlru/simplewlru.go
Severity:	Critical	**Category:**	Transaction processing Security
Likelihood:	Medium	**Impact:**	High

Description:

The Resize function is located in utils/wlru.go. If maxSize is negative, it triggers an infinite loop within the function, potentially causing a Denial of Service (DoS) as it consumes system resources indefinitely.

```
// Resize changes the cache size.
func (c *Cache) Resize(maxWeight uint, maxSize int) (evicted int) {
    c.lock.Lock()
    evicted = c.lru.Resize(maxWeight, maxSize)
    c.lock.Unlock()
    return evicted
}

// RemoveOldest removes the oldest item from the cache.
func (c *Cache) RemoveOldest() (key interface{}, value interface{}, ok
bool) {
    c.lock.Lock()
    key, value, ok = c.lru.RemoveOldest()
    c.lock.Unlock()
    return
}
```

Result: Confirmed

Fix Result: Fixed

3.5 Should Add Thread Lock

ID:	NVE-005	**Location:**	u2udb/flushable/flushable.go
Severity:	High	**Category:**	Transaction processing Security
Likelihood:	Medium	**Impact:**	Medium

Description:

The cacheBatch structure allows modification of its internal states, such as writes and size through the Put and Delete methods , without applying thread locks to the database (db). This could potentially lead to data corruption or loss if multiple threads simultaneously access these methods.

```
// cacheBatch is a batch structure.
type cacheBatch struct {
    db     *Flushable
    writes []kv
    size   int
}

// Put adds "add key-value pair" operation into batch.
func (b *cacheBatch) Put(key, value []byte) error {

b.writes=append(b.writes,kv{common.CopyBytes(key),common.CopyBytes(value)})
    b.size += len(value) + len(key)
    return nil
}

// Delete adds "remove key" operation into batch.
func (b *cacheBatch) Delete(key []byte) error {
    b.writes = append(b.writes, kv{common.CopyBytes(key), nil})
    b.size += len(key)
    return nil
}
```

Result: Confirmed

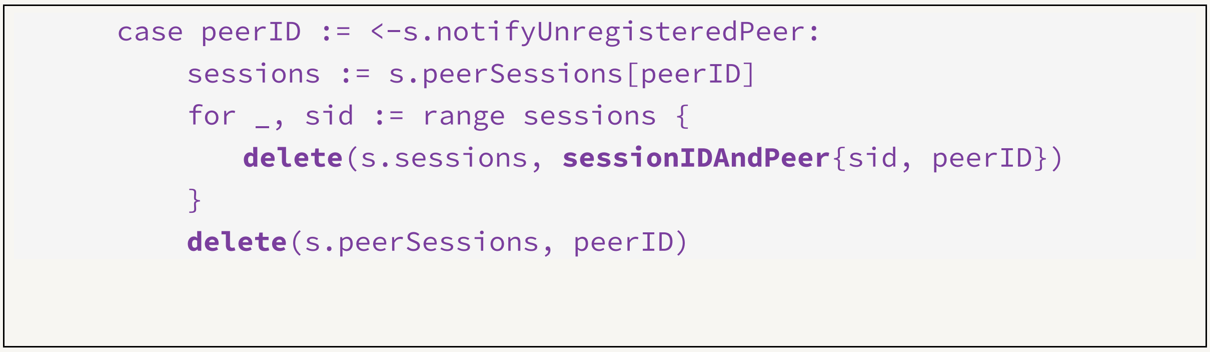
Fix Result: Fixed

3.6 Missing peerID Validation in Loop Execution

ID:	NVE-006	**Location:**	gossip/basestream/basestreamseeder/seeder.go
Severity:	Medium	**Category:**	P2P Communication Security
Likelihood:	Low	**Impact:**	Medium

Description:

The code section handling peerIDs within a session does not include a preliminary check to determine whether the peerID exists. If the peerID is nonexistent, the associated for loop will not execute, likely leading to unhandled scenarios or failures in session management.



```
        case peerID := <-s.notifyUnregisteredPeer:
            sessions := s.peerSessions[peerID]
            for _, sid := range sessions {
                delete(s.sessions, sessionIDAndPeer{sid, peerID})
            }
            delete(s.peerSessions, peerID)
```

Result: Confirmed

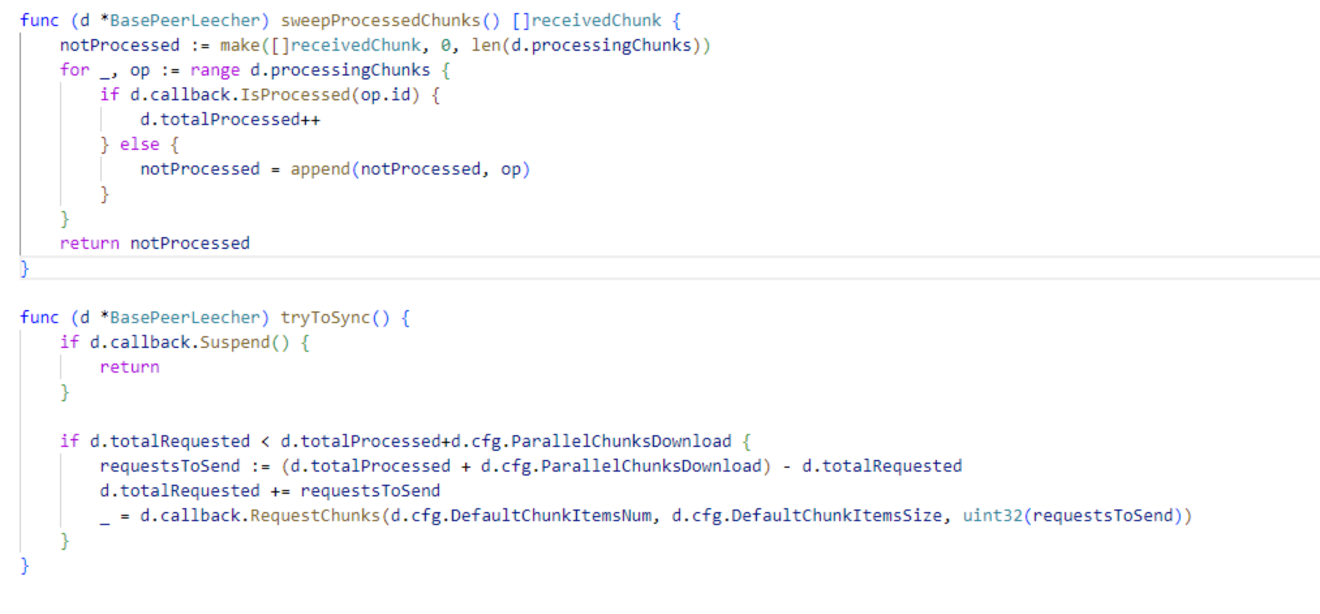
Fix Result: Fixed

3.7 Potential Integer Overflow in Function tryToSync

ID:	NVE-007	**Location:**	gossip/basestream/basepeerleecher/session.go
Severity:	Medium	**Category:**	P2P Communication Security
Likelihood:	Medium	**Impact:**	Medium

Description:

In the tryToSync function, there is a calculation involving d.totalProcessed and d.cfg.ParallelChunksDownload which may result in an integer overflow. This problematic situation arises when d.totalProcessed is excessively incremented by the sweepProcessedChunks function as the attacker manipulates to forcefully increase this value.



```go
func (d *BasePeerLeecher) sweepProcessedChunks() []receivedChunk {
    notProcessed := make([]receivedChunk, 0, len(d.processingChunks))
    for _, op := range d.processingChunks {
        if d.callback.IsProcessed(op.id) {
            d.totalProcessed++
        } else {
            notProcessed = append(notProcessed, op)
        }
    }
    return notProcessed
}

func (d *BasePeerLeecher) tryToSync() {
    if d.callback.Suspend() {
        return
    }

    if d.totalRequested < d.totalProcessed+d.cfg.ParallelChunksDownload {
        requestsToSend := (d.totalProcessed + d.cfg.ParallelChunksDownload) - d.totalRequested
        d.totalRequested += requestsToSend
        _ = d.callback.RequestChunks(d.cfg.DefaultChunkItemsNum, d.cfg.DefaultChunkItemsSize, uint32(requestsToSend))
    }
}
```

Result: Acknowledged

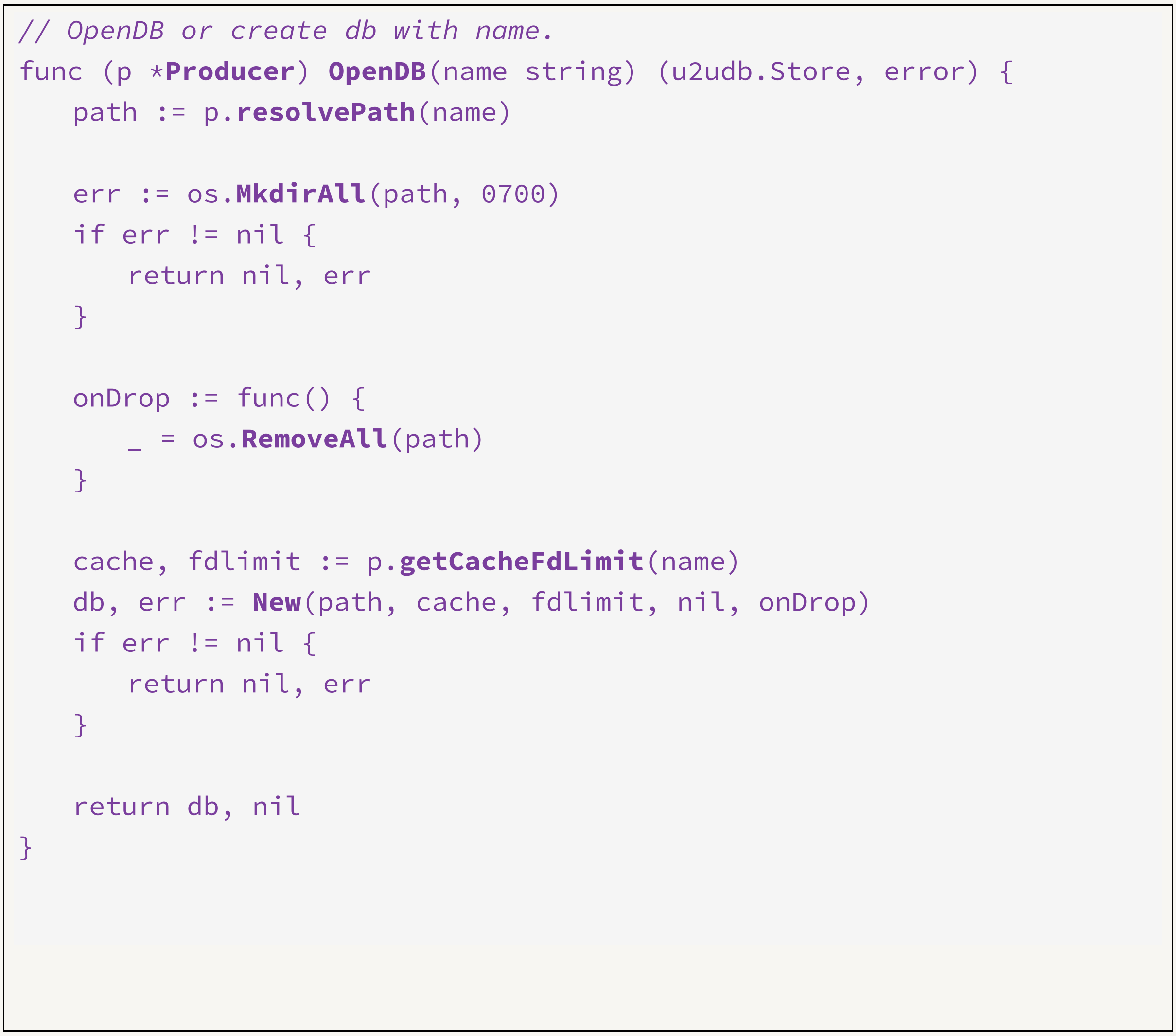
Fix Result: Acknowledged

3.8 Path Traversal in OpenDB Function

ID:	NVE-008	**Location:**	u2udb/leveldb/producer.go
Severity:	Medium	**Category:**	Transaction processing Security
Likelihood:	Medium	**Impact:**	Low

Description:

The OpenDB function does not sanitize the name parameter adequately, allowing directory traversal attacks through the use of "../" to access parent directories.



```
// OpenDB or create db with name.
func (p *Producer) OpenDB(name string) (u2udb.Store, error) {
    path := p.resolvePath(name)

    err := os.MkdirAll(path, 0700)
    if err != nil {
        return nil, err
    }

    onDrop := func() {
        _ = os.RemoveAll(path)
    }

    cache, fdlimit := p.getCacheFdLimit(name)
    db, err := New(path, cache, fdlimit, nil, onDrop)
    if err != nil {
        return nil, err
    }

    return db, nil
}
```

Result: Confirmed

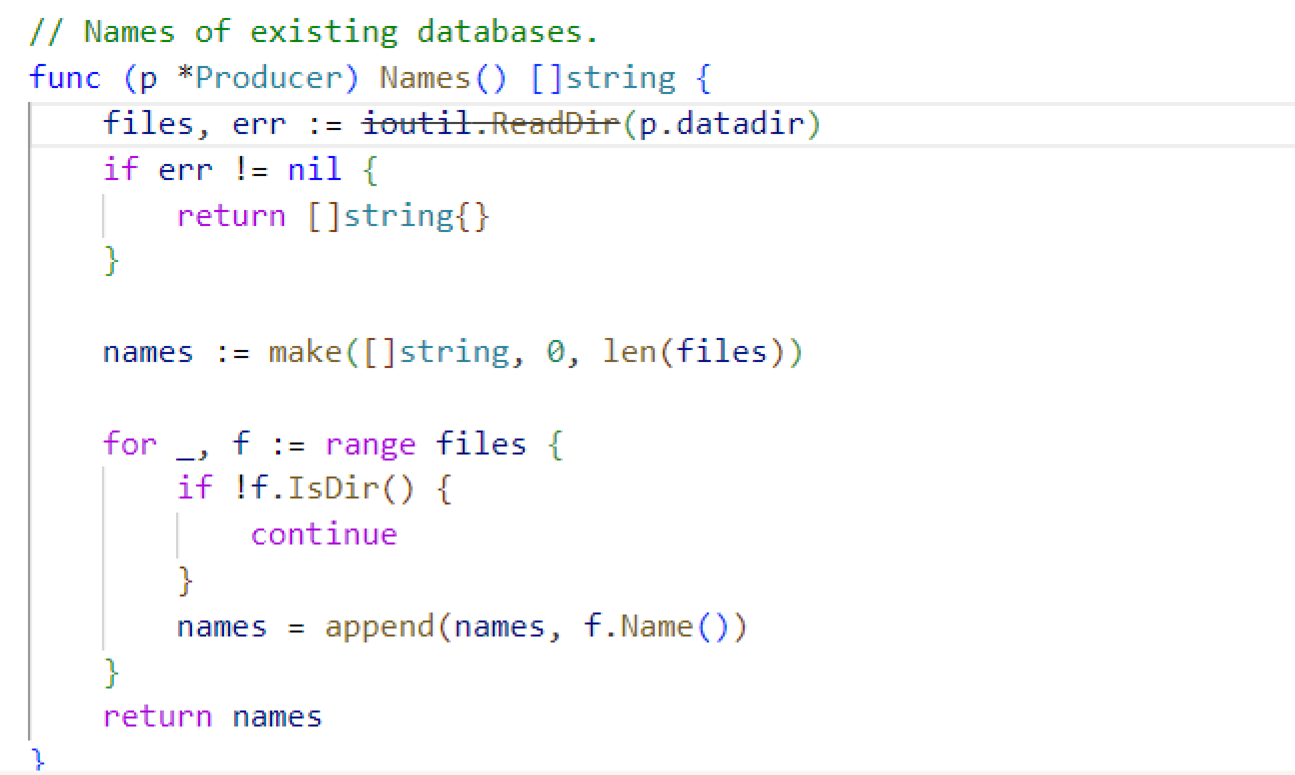
Fix Result: Fixed

3.9 Use of Deprecated ioutil.ReadDir Function

ID:	NVE-009	**Location:**	u2udb/leveldb/producer.go
Severity:	Low	**Category:**	Transaction processing Security
Likelihood:	Low	**Impact:**	Low

Description:

ioutil.ReadDir is deprecated: As of Go 1.16, [os.ReadDir] is a more efficient and correct choice: it returns a list of [fs.DirEntry] instead of [fs.FileInfo], and it returns partial results in the case of an error midway through reading a directory.



```
// Names of existing databases.
func (p *Producer) Names() []string {
    files, err := ~~ioutil.ReadDir~~(p.datadir)
    if err != nil {
        return []string{}
    }

    names := make([]string, 0, len(files))

    for _, f := range files {
        if !f.IsDir() {
            continue
        }
        names = append(names, f.Name())
    }
    return names
}
```

Result: **Confirmed**

Fix Result: Fixed

3.10 Unverified Return Value in MigrateTables Function

ID:	NVE-010	**Location:**	u2udb/table/reflect.go
Severity:	Low	**Category:**	Transaction processing Security
Likelihood:	Low	**Impact:**	Low

Description:

In the MigrateTables function, there is an unchecked defer statement defer keys.Check() used to verify data integrity or configuration after processing. The lack of error checking on this statement means that if an error occurs during the Check() function execution, it will not be handled or logged, potentially leading to unnoticed failures or inconsistent state within the application.

```
// MigrateTables sets target fields to database tables.
func MigrateTables(s interface{}, db u2udb.Store) {
    value := reflect.ValueOf(s).Elem()
    var keys uniqKeys
    defer keys.Check() // nolint:errcheck
    for i := 0; i < value.NumField(); i++ {
        if prefix := value.Type().Field(i).Tag.Get("table"); prefix !=
"" && prefix != "-" {

            field := value.Field(i)
            var val reflect.Value
            if db != nil {
                keys.Add(prefix)
                table := New(db, []byte(prefix))
                val = reflect.ValueOf(table)
            } else {
                val = reflect.Zero(field.Type())
            }
            field.Set(val)
        }
    }
}
```

Result: Confirmed

Fix Result: Fixed

3.11 Inefficient UniqKey Check Method

ID:	NVE-011	**Location:**	u2udb/table/reflect.go
Severity:	Low	**Category:**	Transaction processing Security
Likelihood:	Low	**Impact:**	Low

Description:

The Check method of uniqKeys uses nested loops to compare keys, which can be inefficient for large numbers of keys. Consider using a more efficient data structure like a map to check for duplicates.

Result: Confirmed

Fix Result: Acknowledged

3.12 Result Of govulncheck

ID:	NVE-012	**Location:**	go.mod
Severity:	Informational	**Category:**	Transaction processing Security
Likelihood:	Low	**Impact:**	Low

Description:

```
Fetching vulnerabilities from the database...

Checking the code against the vulnerabilities...

The package pattern matched the following 56 root packages:
  github.com/unicornultrafoundation/go-helios/common
  github.com/unicornultrafoundation/go-helios/common/bigendian
  github.com/unicornultrafoundation/go-helios/common/littleendian
  github.com/unicornultrafoundation/go-helios/common/prque
  github.com/unicornultrafoundation/go-helios/native/idx
  github.com/unicornultrafoundation/go-helios/hash
  github.com/unicornultrafoundation/go-helios/consensus/dagidx
  github.com/unicornultrafoundation/go-helios/native/pos
  github.com/unicornultrafoundation/go-helios/consensus/election
  github.com/unicornultrafoundation/go-helios/native/dag
  github.com/unicornultrafoundation/go-helios/types
  github.com/unicornultrafoundation/go-helios/u2udb
  github.com/unicornultrafoundation/go-helios/u2udb/devnulldb
  github.com/unicornultrafoundation/go-helios/u2udb/readonlystore
  github.com/unicornultrafoundation/go-helios/u2udb/synced
  github.com/unicornultrafoundation/go-helios/u2udb/flushable
  github.com/unicornultrafoundation/go-helios/u2udb/memorydb
  github.com/unicornultrafoundation/go-helios/u2udb/table
  github.com/unicornultrafoundation/go-helios/utils/cachescale
  github.com/unicornultrafoundation/go-helios/utils/simplewlru
  github.com/unicornultrafoundation/go-helios/consensus
  github.com/unicornultrafoundation/go-helios/utils/wlru
  github.com/unicornultrafoundation/go-helios/utils/wmedian
  github.com/unicornultrafoundation/go-helios/emitter/ancestor
  github.com/unicornultrafoundation/go-helios/emitter/doublesign
  github.com/unicornultrafoundation/go-helios/eventcheck/basiccheck
  github.com/unicornultrafoundation/go-helios/eventcheck/epochcheck
  github.com/unicornultrafoundation/go-helios/eventcheck/parentscheck
  github.com/unicornultrafoundation/go-helios/eventcheck
  github.com/unicornultrafoundation/go-helios/gossip/basestream
  github.com/unicornultrafoundation/go-
helios/gossip/basestream/basestreamleecher
  github.com/unicornultrafoundation/go-
helios/gossip/basestream/basestreamleecher/basepeerleecher
  github.com/unicornultrafoundation/go-helios/utils/workers
  github.com/unicornultrafoundation/go-
helios/gossip/basestream/basestreamseeder
  github.com/unicornultrafoundation/go-helios/gossip/dagordering
  github.com/unicornultrafoundation/go-helios/utils/datasemaphore
  github.com/unicornultrafoundation/go-helios/gossip/dagprocessor
  github.com/unicornultrafoundation/go-helios/gossip/itemsfetcher
  github.com/unicornultrafoundation/go-helios/native/dag/tdag
  github.com/unicornultrafoundation/go-helios/u2udb/batched
  github.com/unicornultrafoundation/go-helios/u2udb/cachedproducer
  github.com/unicornultrafoundation/go-helios/u2udb/fallible
  github.com/unicornultrafoundation/go-helios/u2udb/flaggedproducer
  github.com/unicornultrafoundation/go-helios/utils/piecefunc
  github.com/unicornultrafoundation/go-helios/u2udb/leveldb
  github.com/unicornultrafoundation/go-helios/utils/fmtfilter
  github.com/unicornultrafoundation/go-helios/u2udb/multidb
  github.com/unicornultrafoundation/go-helios/u2udb/nokeyiserr
  github.com/unicornultrafoundation/go-helios/u2udb/pebble
  github.com/unicornultrafoundation/go-helios/u2udb/skiperrors
  github.com/unicornultrafoundation/go-helios/u2udb/skipkeys
  github.com/unicornultrafoundation/go-helios/utils
  github.com/unicornultrafoundation/go-helios/vecengine
  github.com/unicornultrafoundation/go-helios/vecfc
  github.com/unicornultrafoundation/go-helios/utils/adapters
  github.com/unicornultrafoundation/go-helios/vecengine/vecflushable
Govulncheck scanned the following 30 modules and the go1.23.4 standard
library:
  github.com/unicornultrafoundation/go-helios
  github.com/DataDog/zstd@v1.5.2
  github.com/beorn7/perks@v1.0.1
  github.com/cespare/xxhash/v2@v2.2.0
  github.com/cockroachdb/errors@v1.9.1
  github.com/cockroachdb/logtags@v0.0.0-20230118201751-21c54148d20b
  github.com/cockroachdb/pebble@v0.0.0-20230209160836-829675f94811
  github.com/cockroachdb/redact@v1.1.3
  github.com/emirpasic/gods@v1.18.1
  github.com/getsentry/sentry-go@v0.18.0
  github.com/gogo/protobuf@v1.3.2
  github.com/golang/protobuf@v1.5.2
  github.com/golang/snappy@v0.0.5-0.20220116011046-fa5810519dcb
  github.com/kr/pretty@v0.3.1
  github.com/kr/text@v0.2.0
  github.com/matttproud/golang_protobuf_extensions@v1.0.4
  github.com/pkg/errors@v0.9.1
  github.com/prometheus/client_golang@v1.14.0
  github.com/prometheus/client_model@v0.3.0
  github.com/prometheus/common@v0.39.0
  github.com/prometheus/procfs@v0.9.0
  github.com/rogpeppe/go-internal@v1.9.0
  github.com/status-im/keycard-go@v0.2.0
  github.com/syndtr/goleveldb@v1.0.1-0.20220614013038-64ee5596c38a
  github.com/unicornultrafoundation/go-u2u@v1.0.0-
rc1.0.20231015194805-e285ed001123
  golang.org/x/crypto@v0.6.0
  golang.org/x/exp@v0.0.0-20230206171751-46f607a40771
  golang.org/x/sys@v0.7.0
  golang.org/x/text@v0.8.0
  google.golang.org/protobuf@v1.28.1

=== Symbol Results ===

No vulnerabilities found.

=== Package Results ===

No other vulnerabilities found.
```

```
=== Module Results ===

Vulnerability #1: GO-2024-3321
    Misuse of ServerConfig.PublicKeyCallback may cause authorization
bypass in
    golang.org/x/crypto
  More info: https://pkg.go.dev/vuln/GO-2024-3321
  Module: golang.org/x/crypto
    Found in: golang.org/x/crypto@v0.6.0
    Fixed in: golang.org/x/crypto@v0.31.0

Vulnerability #2: GO-2024-2611
    Infinite loop in JSON unmarshaling in google.golang.org/protobuf
  More info: https://pkg.go.dev/vuln/GO-2024-2611
  Module: google.golang.org/protobuf
    Found in: google.golang.org/protobuf@v1.28.1
    Fixed in: google.golang.org/protobuf@v1.33.0

Vulnerability #3: GO-2023-2402
    Man-in-the-middle attacker can compromise integrity of secure
channel in
    golang.org/x/crypto
  More info: https://pkg.go.dev/vuln/GO-2023-2402
  Module: golang.org/x/crypto
    Found in: golang.org/x/crypto@v0.6.0
    Fixed in: golang.org/x/crypto@v0.17.0

Your code is affected by 0 vulnerabilities.
This scan also found 0 vulnerabilities in packages you import and 3
vulnerabilities in modules you require, but your code doesn't appear to
call
these vulnerabilities.
```

Result: Confirmed

Fix Result: Fixed

4. CONCLUSION

In this audit, we thoroughly analyzed **go-helios** Blockchain implementation. The problems found are described and explained in detail in Section 3. The problems found in the audit have been communicated to the project leader. We therefore consider the audit result to be **PASSED**. To improve this report, we greatly appreciate any constructive feedbacks or suggestions, on our methodology, audit findings, or potential gaps in scope/coverage.

5. APPENDIX

5.1 Basic Coding Assessment

5.1.1 Apply Verification Control

- Description: The security of apply verification
- Result: Not found
- Severity: Critical

5.1.2 Authorization Access Control

- Description: Permission checks for external integral functions
- Result: Not found
- Severity: Critical

5.1.3 Forged Transfer Vulnerability

- Description: Assess whether there is a forged transfer notification vulnerability in the contract
- Result: Not found
- Severity: Critical

5.1.4 Transaction Rollback Attack

- Description: Assess whether there is transaction rollback attack vulnerability in the contract.
- Result: Not found
- Severity: Critical

5.1.5 Transaction Block Stuffing Attack

- Description: Assess whether there is transaction blocking attack vulnerability.
- Result: Not found
- Severity: Critical

5.1.6 Soft Fail Attack Assessment

- Description: Assess whether there is soft fail attack vulnerability.
- Result: Not found
- Severity: Critical

5.1.7 Hard Fail Attack Assessment

- Description: Examine for hard fail attack vulnerability
- Result: Not found
- Severity: Critical

5.1.8 Abnormal Memo Assessment

- Description: Assess whether there is abnormal memo vulnerability in the contract.
- Result: Not found
- Severity: Critical

5.1.9 Abnormal Resource Consumption

- Description: Examine whether abnormal resource consumption in contract processing.
- Result: Not found
- Severity: Critical

5.1.10 Random Number Security

- Description: Examine whether the code uses insecure random number.
- Result: Not found
- Severity: Critical

5.2 Advanced Code Scrutiny

5.2.1 Cryptography Security

- Description: Examine for weakness in cryptograph implementation.
- Results: Not Found
- Severity: High

5.2.2 Account Permission Control

- Description: Examine permission control issue in the contract
- Results: Not Found
- Severity: Medium

5.2.3 Malicious Code Behavior

- Description: Examine whether sensitive behavior present in the code
- Results: Not found
- Severity: Medium

5.2.4 Sensitive Information Disclosure

- Description: Examine whether sensitive information disclosure issue present in the code.
- Result: Not found
- Severity: Medium

5.2.5 System API

- Description: Examine whether system API application issue present in the code
- Results: Not found
- Severity: Low

6. 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 Services Agreement, or the scope of services, and terms and conditions provided to the Company in connection with the Agreement. This report provided in connection with the Services set forth in the Agreement shall be used by the Company only to the extent permitted under the terms and conditions set forth in the Agreement. This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes without ExVul's prior written consent.

This report is not, nor should be considered, an "endorsement" or "disapproval" of any particular project or team. This report is not, nor should be considered, an indication of the economics or value of any "product" or "asset" created by any team or project that contracts ExVul to perform a security assessment. This report does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors, business, business model or legal compliance.

This report should not be used in any way to make decisions around investment or involvement with any particular project. This report in no way provides investment advice, nor should be leveraged as investment advice of any sort. This report represents an extensive assessing process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.

Blockchain technology and cryptographic assets present a high level of ongoing risk. ExVul's position is that each company and individual are responsible for their own due diligence and continuous security. ExVul's goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies, and in no way claims any guarantee of security or functionality of the technology we agree to analyze.

7. REFERENCES

[1] MITRE. CWE- 191: Integer Underflow (Wrap or Wraparound).

https://cwe.mitre.org/data/ definitions/191.html.

[2] MITRE. CWE- 197: Numeric Truncation Error.

https://cwe.mitre.org/data/definitions/197. html.

[3] MITRE. CWE-400: Uncontrolled Resource Consumption.

https://cwe.mitre.org/data/ definitions/400.html.

[4] MITRE. CWE-440: Expected Behavior Violation.

https://cwe.mitre.org/data/definitions/440. html.

[5] MITRE. CWE-684: Protection Mechanism Failure.

https://cwe.mitre.org/data/definitions/ 693.html.

[6] MITRE. CWE CATEGORY: 7PK - Security Features.

https://cwe.mitre.org/data/definitions/ 254.html.

[7] MITRE. CWE CATEGORY: Behavioral Problems.

https://cwe.mitre.org/data/definitions/438. html.

[8] MITRE. CWE CATEGORY: Numeric Errors.

https://cwe.mitre.org/data/definitions/189.html.

[9] MITRE. CWE CATEGORY: Resource Management Errors.

https://cwe.mitre.org/data/ definitions/399.html.

[10] OWASP. Risk Rating Methodology.

https://www.owasp.org/index.php/OWASP_Risk_Rating_Methodology

www.exvul.com

contact@exvul.com

@EXVULSEC

github.com/EXVUL-Sec

ExVul