HACKEN

HAVAH BLOCKCHAIN SECURITY ANALYSIS

Hacken OÜ
Parda 4, Kesklinn, Tallinn
10151 Harju Maakond, Eesti
Kesklinna, Estonia
support@hacken.io

Intro

This report may contain confidential information about IT systems and the intellectual property of the Customer, as well as information about potential vulnerabilities and methods of their exploitation.

The report can be disclosed publicly after prior consent by WEB3 SOLUTIONS PTE. LTD. Any subsequent publication of this report shall be without mandatory consent.

Name	Goloop blockchain node
Website	https://havah.io
Repository	https://github.com/havah-project/goloop-havah
Commit	5c68f490dca03ab8f5cfee295d67dade8e0419bc
Platform	L1
Network	HAVAH
Languages	Java, Golang
Methods	Automated Code analysis, Manual review, Issues simulation
Auditor	Yaroslav Bratashchuk (y.bratashchuk@hacken.io)
Approver	Andriy Kashcheyev (a.kashcheyev@hacken.io)
Timeline	14.09.2022 - 21.10.2022

Hacken OÜ
Parda 4, Kesklinn, Tallinn
10151 Harju Maakond, Eesti
Kesklinna, Estonia
support@hacken.io

Table of contents

Hacken OÜ
Parda 4, Kesklinn, Tallinn
10151 Harju Maakond, Eesti
Kesklinna, Estonia
support@hacken.io

Summary

ICONLOOP Blockchain project Loopchain is Layer-1 blockchain protocol implementation and is used for ICON project https://github.com/icon-project.

HAVAH is an interchain NFT platform built by WEB3 SOLUTIONS, ICONLOOP and 2BYTES. **HAVAH** blockchain node repository is https://github.com/havah-project/goloop-havah.

Hacken performed the security audit for this project against commit hash: 5c68f490dca03ab8f5cfee295d67dade8e0419bc

The codebase statistics:

```
-------------------------------------------------------------------------------
Language                     files          blank        comment           code
-------------------------------------------------------------------------------
Go                             416          10983           3003          78946
Java                           592          10527          11544          56719
Python                          73           2108           1598           6899
Markdown                        16           1794              0           6787
Bourne Shell                    24            249            114           1335
YAML                             3             53             28           1200
JSON                            34              0              0            924
Gradle                          18             94             24            634
Dockerfile                       8             48             28            289
make                             4             54             26            233
C                                1             22             39            141
DOS Batch                        1             24              2             74
JavaScript                       1              0              3             41
C/C++ Header                     1              6             27             20
-------------------------------------------------------------------------------
SUM:                          1192          25962          16436         154242
-------------------------------------------------------------------------------
```

The scope of the Audit included the following subsystems:

- Java Execution Environment (VM implementation) https://github.com/havah-project/goloop-havah/tree/main/javaee
- Havah platform (Golang)
- Golang packages that Havah depends on

The main emphasis was on Java Iconloop VM implementation.

Overall the codebase and implementation of the features included in the scope of the Audit are of satisfactory quality.

Few minor and medium-severity issues have been discovered. All issues have been acknowledged and partially already addressed by the HAVAH development team.

No high-severity security issues were discovered.

Documentation quality

The total Documentation Quality score is **8.0** out of **10**.

The project is well documented. The most important types and operations are documented using code comments. A few things could be polished, like intro documentation on the `goloop` and `gochain` and the difference, which may not be very clear for those who just got acquainted with the project. Also, some intro to the `goloop cli` would be helpful together with a bit more verbose description for each command and subcommand.

Hacken OÜ
Parda 4, Kesklinn, Tallinn
10151 Harju Maakond, Eesti
Kesklinna, Estonia
support@hacken.io

Code quality

The total Code Quality score is **9.5** out of **10**.

Code quality score is calculated against Java part of the project. A number of non-fixed issues (even with low severity) contribute to low score. Golang part of the project has poor test coverage and lack of comments.

Almost every critical function is tested both by unit and integration tests. Some functions or cli commands description doesn't cover enough details or is missing at all, but for an experienced developer, it is not hard to figure it out. In order to make the code more strict - we suggest using a linter in the CD pipeline.

Architecture quality

The architecture quality score is **9.0** out of **10**.

Node is well designed. It utilizes language paradigms. All entities and components are loosely coupled, which makes it easy to test them. RPC API is also well-designed, but goloop CLI could be more verbose when it comes to using it.

Security score

The security score is **10.0** out of **10**.

As a result of the analysis, the code contained **2** Medium, **3** Low severity security-related issues.

All security scope issues have been addressed and fixed by the team.

The overall project Total score is **8.0** out of **10**.

Non-fixed raised `Code Quality` related issues have been acknowledged by the team. The total score includes test coverage metrics only for Java Execution Environment module: **56**% Unit tests coverage is a very important metric and plays a significant role in the total score when coverage is below 70%.

Hacken OÜ
Parda 4, Kesklinn, Tallinn
10151 Harju Maakond, Eesti
Kesklinna, Estonia
support@hacken.io

Analysis

Static code analysis for VM (Java Execution Environment)

Category	Score	Details
Reliability	E	33 coding errors (bugs) found that should be fixed
Security	A	0 vulnerabilities found
Maintainability	A	3.5k places that could be confusing and hard to maintain
Security Review	E	22 occurrences of security-sensitive code that require manual review to assess whether vulnerability exists or not

The most crucial reliability bugs are mentioned in the issues. The rest of them are not critical or false positives.

Almost all maintainability complaints are due to missing comments, unused method parameters, and cognitive complexity, which is not critical.

All security-sensitive issues are related to printing stack traces when catching exceptions. This should be reviewed before delivering the code to production.

Tests Coverage

VM (Java Execution Environment) Coverage Summary

56.5% of lines are covered with unit tests

Package	Class, %	Method, %	Line, %
a	81.8% (9/11)	33.6% (50/149)	32% (117/366)
collection	0% (0/1)	0% (0/8)	0% (0/69)
e.s.java.lang	100% (1/1)	100% (2/2)	100% (4/4)
example	0% (0/7)	0% (0/90)	0% (0/420)
example.token	0% (0/3)	0% (0/37)	0% (0/100)
example.util	0% (0/3)	0% (0/21)	0% (0/43)
foundation.icon.ee	100% (1/1)	100% (2/2)	100% (2/2)
foundation.icon.ee.io	60% (3/5)	34.3% (36/105)	22.2% (89/401)
foundation.icon.ee.ipc	38.5% (5/13)	51.3% (40/78)	66.2% (301/455)
foundation.icon.ee.logger	100% (2/2)	26.9% (21/78)	44.4% (63/142)
foundation.icon.ee.score	100% (10/10)	81.5% (66/81)	78.7% (402/511)
foundation.icon.ee.struct	100% (14/14)	98.9% (91/92)	92% (335/364)

foundation.icon.ee.tooling.abi	100% (10/10)	87.9% (58/66)	81.9% (326/398)
foundation.icon.ee.tooling.deploy	100% (1/1)	85.7% (6/7)	81.5% (44/54)
foundation.icon.ee.types	78.9% (15/19)	75% (105/140)	70% (231/330)
foundation.icon.ee.util	90.9% (20/22)	60.8% (62/102)	75.4% (617/818)
foundation.icon.ee.util.xxhash	0% (0/9)	0% (0/60)	0% (0/175)
governance	0% (0/2)	0% (0/40)	0% (0/44)
i	72% (18/25)	61.9% (83/134)	66.1% (271/410)
org.aion.avm	62.5% (5/8)	52.3% (34/65)	57.5% (131/228)
org.aion.avm.core	92.1% (35/38)	71.5% (176/246)	72.4% (845/1167)
org.aion.avm.core.arraywrapping	100% (8/8)	86.6% (84/97)	77% (553/718)
org.aion.avm.core.classgeneration	100% (2/2)	86.7% (13/15)	97.2% (137/141)
org.aion.avm.core.classloading	100% (2/2)	78.6% (11/14)	84.2% (85/101)
org.aion.avm.core.dappreading	50% (1/2)	75% (3/4)	86.8% (33/38)
org.aion.avm.core.exceptionwrapping	100% (3/3)	73.3% (11/15)	77.9% (53/68)
org.aion.avm.core.instrument	82.4% (14/17)	77.2% (71/92)	85.1% (372/437)
org.aion.avm.core.miscvisitors	95.7% (22/23)	84% (100/119)	77.9% (320/411)
org.aion.avm.core.persistence	88.2% (15/17)	77.2% (112/145)	69.7% (560/804)
org.aion.avm.core.rejection	55.6% (5/9)	32.9% (25/76)	51.9% (80/154)
org.aion.avm.core.shadowing	100% (6/6)	80% (32/40)	74.5% (102/137)
org.aion.avm.core.stacktracking	100% (3/3)	100% (12/12)	100% (48/48)
org.aion.avm.core.types	90.5% (19/21)	71.9% (120/167)	77.4% (613/792)
org.aion.avm.core.util	100% (8/8)	69% (40/58)	72.5% (253/349)
org.aion.avm.core.verification	100% (2/2)	83.3% (5/6)	76.5% (26/34)
org.aion.avm.tooling.deploy	100% (6/6)	81.1% (30/37)	72.9% (113/155)
org.aion.avm.tooling.deploy.eliminator	100% (12/12)	95.4% (62/65)	91.2% (271/297)
org.aion.avm.tooling.deploy.renamer	100% (6/6)	78.4% (29/37)	80.2% (178/222)
org.aion.avm.utilities	100% (3/3)	90% (18/20)	75.2% (115/153)
org.aion.avm.utilities.analyze	100% (6/6)	89.5% (17/19)	94.7% (144/152)
org.aion.parallel	100% (1/1)	73.3% (11/15)	84% (21/25)
org.slf4j.impl	100% (1/1)	80% (4/5)	86.7% (13/15)
p.score	50% (2/4)	52.6% (41/78)	30.7% (103/335)
pi	90.9% (10/11)	44.4% (91/205)	46.4% (274/590)

s.java.lang	100% (26/26)	19.1% (128/669)	16.6% (280/1686)
s.java.lang.invoke	100% (2/2)	83.3% (5/6)	46.9% (38/81)
s.java.math	100% (4/4)	23.5% (27/115)	24% (93/387)
s.java.util	80% (4/5)	37.5% (15/40)	46.6% (34/73)
s.java.util.concurrent	100% (1/1)	14.3% (2/14)	43.9% (18/41)
s.score	100% (3/3)	33.3% (6/18)	31.8% (7/22)
score	40% (2/5)	6.2% (4/64)	7.4% (5/68)
test	0% (0/2)	0% (0/10)	0% (0/123)
testcases	0% (0/8)>	0% (0/74)	0% (0/169)

56.5% coverage is considered below high code quality, especially for blockchain projects which run in a hostile environment and have a number of attack vectors.

We recommend targeting at least 80% test coverage overall for the project and 100% for critical subsystems (VM, consensus, P2P, RPC)

Integration tests in `goloop-testsuite`

Golang packages coverage summary

39.5% of lines are covered with unit tests. The most critical code logic is covered well by unit and integration tests. Still, we recommend improving test coverage.

Low test coverage percentage has a strong negative influence on the total project score.

Hacken OÜ
Parda 4, Kesklinn, Tallinn
10151 Harju Maakond, Eesti
Kesklinna, Estonia
support@hacken.io

Issues

Incomplete Cleanup or Missing Release of Resources after Effective Lifetime

Methods do not properly "clean up" and remove temporary or supporting resources after they have been used.

ID	LOOP-12
Scope	VM, Security
Severity	**MEDIUM**
Status	Fixed (4d5b58)
Vulnerability Type	Incomplete Cleanup

Details

Connections, streams, files, and other classes that implement the `Closeable` interface, need to be closed after use. Further, that `close` call must be made in a `finally` block otherwise, an exception could keep the call from being made.

Failure to properly close resources will result in a resource leak which could bring first the application and then perhaps the box the application is on to their knees.

Places of occurrences (within javaee):

- exec/src/java/foundation/icon/ee/ipc/Proxy.java#L71
- tooling/src/java/foundation/icon/ee/tooling/deploy/OptimizedJarBuilder.java#L129
- tooling/src/java/org/aion/avm/tooling/deploy/JarOptimizer.java#L174
- tooling/src/java/org/aion/avm/tooling/deploy/renamer/Renamer.java#L214

Recommendation

Make a `close` call in a `finally` block.

Untrusted dependency package for secp256k1 implementation

Goloop uses `seccp256k1-go` package with almost zero reputation.

ID	LOOP-2
Scope	Cryptography, Security
Severity	**MEDIUM**
Status	Fixed (be7ad7ac)
Vulnerability Type	Use of Unmaintained Third-Party Components

Details

Goloop relies on a third-party library `seccp256k1-go` that is not actively supported or maintained by the original developer.

Reliance on components that are no longer maintained can make it difficult or impossible to fix significant bugs, vulnerabilities, or quality issues. This issue makes it more difficult to maintain the software, which indirectly affects security by making it more difficult or time-consuming to find and/or fix vulnerabilities. It also might make it easier to introduce vulnerabilities.

Recommendation

Avoid using unmaintained third-party libraries. We recommend migrating to https://github.com/ethereum/go-ethereum/tree/master/crypto/secp256k1 or https://github.com/decred/dcrd/tree/master/dcrec/secp256k1.

Operation on a Resource after Expiration or Release

Sdk package uses, accesses, or otherwise operates on a resource after that resource has expired, released, or revoked.

ID	LOOP-8
Scope	Cryptography, Security
Severity	**MEDIUM**
Status	Fixed (3de20605e14)
Vulnerability Type	Operation on a Resource after Expiration or Release

Details

Bouncy Castle Cryptography APIs of version 1.60 is used heavily throughout goloop java codebase. At least 2 vulnerabilities (CVE-2020-15522, CVE-2020-28052) have been discovered since version 1.60.

Recommendation

Use the latest release of Bouncy Castle package (1.70).

Dead Code

Packages contain dead code, which can never be executed.

ID	LOOP-6
Scope	Code Quality
Severity	**MEDIUM**
Status	Partially fixed (3de20605, 42150af0c)
Vulnerability	Dead Code

Details

Dead code is source code that can never be executed in a running program. The surrounding code makes it impossible for a section of code to ever be executed.

Hacken OÜ
Parda 4, Kesklinn, Tallinn
10151 Harju Maakond, Eesti
Kesklinna, Estonia
support@hacken.io

We found 27 dead code occurrences throughout golang codebase. Below is a concatenated output generated by (tools): staticcheck, govet, deadcode:

```
service/log.go:12:2: `logMsg` is unused
    logMsg   = configLogLevel >= LogLevelMsg
    ^
service/log.go:13:2: `logDebug` is unused
    logDebug = configLogLevel >= LogLevelDebug
    ^
service/manager.go:613:6: `boolToJSON` is unused
func boolToJSON(v bool) interface{} {
     ^
chain/chain.go:584:6: `registerTaskFactory` is unused
func registerTaskFactory(name string, factory TaskFactory) {
     ^
common/crypto/key.go:54:2: `publicKeyUncompressed` is unused
    publicKeyUncompressed byte = 0x4 // x coord + y coord
    ^
common/crypto/signature.go:16:2: `invalidV` is unused
    invalidV        = 0xff
    ^
common/db/bucket.go:76:6: `nonNilBytes` is unused
func nonNilBytes(bz []byte) []byte {
     ^
common/wallet/plugin.go:36:6: `builder` is unused
type builder func(params map[string]string) (interface{}, error)
     ^
common/containerdb/common.go:202:6: `must` is unused
func must(b []byte, e error) error {
     ^
common/trie/ompt/rlp.go:45:6: `rlpIsList` is unused
func rlpIsList(buf []byte) bool {
     ^
common/trie/ompt/rlp.go:91:6: `rlpLen` is unused
func rlpLen(b []byte) (int, error) {
     ^
common/trie/ompt/rlp.go:246:6: `rlpEncodeObjects` is unused
func rlpEncodeObjects(olist ...interface{}) ([]byte, error) {
     ^
common/legacy/loopchain.go:137:2: `accountGeneral` is unused
    accountGeneral  = 0
    ^
common/legacy/loopchain.go:138:2: `accountGenesis` is unused
    accountGenesis  = 1
    ^
common/legacy/loopchain.go:139:2: `accountTreasury` is unused
    accountTreasury = 2
    ^
service/transaction/transaction_v3.go:20:2: `configCheckDataOnPreValidate` is unused
    configCheckDataOnPreValidate = false
    ^
service/state/contract.go:152:2: `contractSnapshotImplEntries` is unused
    contractSnapshotImplEntries = 7
    ^
service/contract/contractmanager.go:339:2: `tmpRoot` is unused
    tmpRoot                = "tmp"
    ^
service/contract/contractmanager.go:342:2: `tryTmpNum` is unused
    tryTmpNum              = 10
    ^
service/contract/systemscore.go:202:2: `addressType` is unused
    addressType      = reflect.TypeOf((*module.Address)(nil)).Elem()
    ^
common/ipc/server.go:102:2: unreachable: unreachable code
    s.listener.Close()
    ^
server/wsevent.go:195:2: unreachable: unreachable code
    return eventIndexes, false
    ^
common/trie/ompt/hash.go:112:2: unreachable: unreachable code
```

```
    return nil
    ^
common/trie/cache/nodecache_test.go:66:2: SA4006: this value of `data` is never used
    data, ok = cache.Get(n2[0:3], h2)
    ^
```

Recommendation

Deadcode occurrences indicate poor code hygiene and possible accumulation of technical debt.

Expected Behavior Violation

`foundation.icon.test.cases.MultiSigWalletTest` is failing, which could mean that logic behind it does not perform according to its specification.

ID	LOOP-13
Scope	VM, Security
Severity	**MEDIUM**
Status	Fixed (2b24914d)
Vulnerability Type	Expected Behavior Violation

Details

Execution of `testJavaScore` integration tests fail. The output of a failed test is:

```
foundation.icon.test.cases.MultiSigWalletTest > testJava() STANDARD_OUT
    --> deploy: MultiSigWallet
       scoreAddr = cxdd778ca85ef4367eeb6cbfaa732dc107b773498a
    <-- deploy
    --> setup: initial wallets
       Address of owner: hxb6924a53147b62381b632ba8048bec9f90cc01f1
       Address of Alice: hx0ddb2e0b0c45e6d66a0f39c34a4a292f14bba65e
       Address of Bob:   hxed4b11871da33b08d7caf8547f1d41b0e5712b99
       transfer: 5 icx to multiSigWallet
       ICX balance of cxdd778ca85ef4367eeb6cbfaa732dc107b773498a: 5000000000000000000
    <-- setup
    --> call: submitIcxTransaction() - send 2 icx to Bob
       confirmTransaction() by Alice
foundation.icon.test.cases.MultiSigWalletTest > testJava() FAILED
    java.io.IOException at MultiSigWalletTest.java:123
```

Recommendation

A test should not fail. If it is meant to fail, then "failing conditions" should be checked and be successfully asserted as an error.

Improper Handling of Highly Compressed Data (Data Amplification)

Packages incorrectly handles a compressed input with a very high compression ratio that produces a large output.

ID	LOOP-7
Scope	Code Quality
Severity	**MEDIUM**

Status	Fixed (b894cc51)
Vulnerability	Improper Handling of Highly Compressed Data (Data Amplification)

Details

An example of data amplification is a "decompression bomb," a small ZIP file that can produce a large amount of data when it is decompressed.

We found 1 occurrence throughout golang codebase. Below is the output generated by gosec (tool)

```
service/contract/contractmanager.go:402:16: G110: Potential DoS vulnerability via decompression bomb
    if _, err := io.Copy(out, in); err != nil {
                    ^
```

Recommendation

Replace

```
    if _, err := io.Copy(out, in); err != nil {
        return errors.WithCode(err, errors.CriticalIOError)
    }
```

with

```
    _, err := io.CopyN(out, in, 1024) // 1024 is an example
    if err != nil {
        if err == io.EOF {
            // handle eof
        }
        return errors.WithCode(err, errors.CriticalIOError)
    }
```

Multiplication of durations

Packages contain cases where two time.Duration values are multiplied in possibly erroneous ways.

ID	LOOP-5
Scope	Code Quality
Severity	MEDIUM
Status	Fixed (0fbfa03a)
Vulnerability	Business logic error

Details

Using durationcheck tool we spotted 2 places where durations are multiplied in a possibly erroneous way.

```
service/sync/client.go:28:27: Multiplication of durations: `time.Millisecond * p.expired`
    p.timer = time.AfterFunc(time.Millisecond*p.expired, func() {
                             ^
service/sync/client.go:52:27: Multiplication of durations: `time.Millisecond * p.expired`
    p.timer = time.AfterFunc(time.Millisecond*p.expired, func() {
                             ^
```

Recommendation

Replace

```
    p.timer = time.AfterFunc(time.Millisecond*p.expired, func() {...}
```

with

```
    p.timer = time.AfterFunc(time.Millisecond*time.Duration(p.expired), func() {...} // p.expired is of type int
```

NULL Pointer Dereference

A "NullPointerException" could be thrown within java packages.

ID	LOOP-9
Scope	VM, Security
Severity	**MEDIUM**
Status	Fixed (f876adb3)
Vulnerability Type	NULL Pointer Dereference

Details

A reference to `null` should never be dereferenced/accessed. Doing so will cause a `NullPointerException` to be thrown. At best, such an exception will cause abrupt program termination. At worst, it could expose debugging information that would be useful to an attacker, or it could allow an attacker to bypass security measures.

Note that when they are present, this rule takes advantage of `@CheckForNull` and `@Nonnull` annotations defined in JSR-305 to understand which values are and are not nullable except when `@Nonnull` is used on the parameter to equals, which by contract should always work with null.

Places of occurrences (within javaee):

- rt/src/java/s/java/lang/String.java#L39 (`value` is nullable)
- rt/src/java/s/java/lang/String.java#L57 (`bytes` is nullable)
- rt/src/java/s/java/lang/String.java#L65 (`buffer` is nullable)
- rt/src/java/s/java/lang/String.java#L73 (`builder` is nullable)
- rt/src/java/s/java/lang/String.java#L127 (`sb` is nullable)
- rt/src/java/s/java/lang/String.java#L136 (`cs` is nullable)
- rt/src/java/s/java/lang/String.java#L157 (`anotherString` is nullable)
- rt/src/java/s/java/lang/String.java#L166 (`str` is nullable)
- rt/src/java/s/java/lang/String.java#L187 (`prefix` is nullable)
- rt/src/java/s/java/lang/String.java#L196 (`prefix` is nullable)
- rt/src/java/s/java/lang/String.java#L205 (`prefix` is nullable)
- rt/src/java/s/java/lang/String.java#L352 (`a` is nullable)
- rt/src/java/s/java/lang/String.java#L373 (`a` is nullable)
- rt/src/java/s/java/lang/StringBuffer.java#L26 (`str` is nullable)
- rt/src/java/s/java/lang/StringBuffer.java#L125 (`str` is nullable)
- rt/src/java/s/java/lang/StringBuffer.java#L236 (`str` is nullable)

- rt/src/java/s/java/lang/StringBuilder.java#L35 (`str` is nullable)
- rt/src/java/s/java/lang/StringBuilder.java#L204 (`str` is nullable)

Recommendation

Do not deference null pointer or do not use null in a case where an object is required, including the following cases:

- Calling the instance method of a null object
- Accessing or modifying the field of a null object
- Taking the length of null as if it were an array
- Accessing or modifying the elements of null as if it were an array
- Throwing null as if it were a Throwable value

Fix Comment

The motivation for current code behavior is to be as close to JRE String behavior as possible. Since `new java.lang.String((char[])null)` throws NPE, goloop instrumented String should also throw NPE exception. The Smart Contract execution environment will throw an exception, and the failure is recorded in the blockchain. And VM will continue executing the next smart contract.

Poor file permissions used

File permissions are set to allow anyone to modify those files.

ID	LOOP-3
Scope	Code Quality
Severity	**MEDIUM**
Status	Fixed (9414ab6)
Vulnerability	Incorrect Default Permissions

Details

We found 1 occurrence throughout golang codebase. Below is the output generated by gosec (tool)

`service/contract/contractmanager.go:417:12: G306: Expect WriteFile permissions to be 0600 or less`:

```
    if err := ioutil.WriteFile(sPath, code, 0755); err != nil {
              ^
```

Recommendation

Change file permission to 0600.

Returning NULL over an empty value

Methods return `null` instead of empty values.

ID	LOOP-10
Scope	VM, Code Quality
Severity	**MEDIUM**
Status	Fixed (Dummy implementation)
Vulnerability Type	NULL Pointer Dereference

Details

Empty arrays and collections should be returned instead of `null`. Returning `null` instead of an actual array, collection or map forces callers of the method to explicitly test for nullity, making them more complex and less readable. Moreover, in many cases, `null` is used as a synonym for empty.

Calling `toString()` or `clone()` on an object should always return a string or an object. Returning `null` instead contravenes the method's implicit contract.

Places of occurrences (within javaee):

- api/src/java/score/Address.java#L55
- api/src/java/score/Address.java#L86

Recommendation

Do not use a `null` where an object is required. For functions that return an array, it is preferable to return an empty array over a null value.

Returning NULL over an empty value

Methods return `null` instead of empty values.

ID	LOOP-10
Scope	VM, Code Quality
Severity	**MEDIUM**
Status	Fixed (Dummy implementation)
Vulnerability Type	NULL Pointer Dereference

Details

Empty arrays and collections should be returned instead of `null`. Returning `null` instead of an actual array, collection or map forces callers of the method to explicitly test for nullity, making them more complex and less readable. Moreover, in many cases, `null` is used as a synonym for empty.

Calling `toString()` or `clone()` on an object should always return a string or an object. Returning `null` instead contravenes the method's implicit contract.

Places of occurrences (within javaee):

- api/src/java/score/Address.java#L55
- api/src/java/score/Address.java#L86

Recommendation

Do not use a `null` where an object is required. For functions that return an array, it is preferable to return an empty array instead of a null value.

Inconsistent error handling and potential weaknesses

Packages do not use a standardized method for handling errors throughout the code, which might introduce inconsistent error handling and resultant weaknesses.

ID	LOOP-4
Scope	Code Quality
Severity	**LOW**
Status	New
Vulnerability Type	Missing Standardized Error Handling Mechanism

Details

If the application handles errors individually, on a one-by-one basis, this is likely to result in inconsistent error handling. The causes of errors may be lost. Also, detailed information about the causes of an error may be unintentionally logged or returned to the client.

We found 79 places where errors must be handled throughout golang codebase. Tools used: nilerr, staticcheck, errcheck, errorlint, errchkjson

```
chain/chain.go:503:21: Error return value of `c._waitResultOf` is not checked
        go c._waitResultOf(task)
                           ^
chain/taskpruning.go:224:13: Error return value of `os.Rename` is not checked
            os.Rename(dbbk, target)
                      ^
chain/taskpruning.go:243:13: Error return value of `os.Rename` is not checked
            os.Rename(gsbk, gsfile)
                      ^
havah/chainscore_basic.go:68:23: Error return value of `as.MigrateForRevision` is not checked
    as.MigrateForRevision(s.cc.ToRevision(int(code.Int64())))
                          ^
service/manager.go:739:35: Error return value of `state.NewWorldSnapshotWithBuilder` is not checked
    state.NewWorldSnapshotWithBuilder(e.Builder(), r.StateHash, vh, ess)
                                      ^
service/manager.go:752:35: Error return value of `state.NewWorldSnapshotWithBuilder` is not checked
    state.NewWorldSnapshotWithBuilder(e.Builder(), r.StateHash, vh, es)
                                      ^
service/memberlist.go:66:47: Error return value of `itr.Next` is not checked
    for itr := m2.Iterator(); itr.Has(); itr.Next() {
                                                ^
service/transactionlist_test.go:163:7: Error return value of `l.Add` is not checked
    l.Add(tx2, false)
         ^
service/transactionlist_test.go:164:7: Error return value of `l.Add` is not checked
    l.Add(tx1, false)
         ^
service/transactionlist_test.go:170:7: Error return value of `l.Add` is not checked
```

```
    l.Add(tx2, false)
         ^
service/transactionmanager.go:90:46: Error return value of `itr.Next` is not checked
    for itr := l.Iterator(); itr.Has(); itr.Next() {
                                                ^
service/transactionpool.go:253:42: Error return value of `i.Next` is not checked
    for i := txs.Iterator(); i.Has(); i.Next() {
                                           ^
service/transition.go:525:54: Error return value of `itr.Next` is not checked
        for itr := receipts.Iterator(); itr.Has(); itr.Next() {
                                                            ^
service/transition.go:547:43: Error return value of `i.Next` is not checked
    for i := list.Iterator(); i.Has(); i.Next() {
                                          ^
service/transition.go:707:40: Error return value of `i.Next` is not checked
    for i := l.Iterator(); i.Has(); i.Next() {
                                        ^
server/websocket.go:119:14: Error return value of `c.WriteJSON` is not checked
        c.WriteJSON(&wsResponse)
                    ^
server/websocket.go:130:14: Error return value of `c.WriteJSON` is not checked
        c.WriteJSON(&wsResponse)
                    ^
server/wsblock.go:101:51: Error return value of `rit.Next` is not checked
                   for rit := rl.Iterator(); rit.Has(); rit.Next() {
                                                                 ^
server/wsevent.go:95:49: Error return value of `rit.Next` is not checked
            for rit := rl.Iterator(); rit.Has(); rit.Next() {
                                                          ^
common/address_test.go:139:22: Error return value of `got.SetStringStrict` is not checked
        got.SetStringStrict(addr)
                            ^
common/address_test.go:141:17: Error return value of `got2.SetString` is not checked
        got2.SetString(addr)
                       ^
common/log/filter.go:50:21: Error return value of `f.fileWriter.Write` is not checked
        f.fileWriter.Write(buf)
                           ^
common/trie/mta/accumulator_test.go:51:11: Error return value of `a.Recover` is not checked
    a.Recover()
           ^
common/codec/msgpack.go:53:20: Error return value of `r.real.DecodeNil` is not checked
            r.real.DecodeNil()
                           ^
common/containerdb/arraydb_test.go:31:13: Error return value of `arraydb.Put` is not checked
    arraydb.Put("Value1")
             ^
common/containerdb/arraydb_test.go:32:13: Error return value of `arraydb.Put` is not checked
    arraydb.Put("Value2")
             ^
common/containerdb/arraydb_test.go:33:13: Error return value of `arraydb.Set` is not checked
    arraydb.Set(1, "Value3")
             ^
common/containerdb/dictdb_test.go:32:11: Error return value of `dict2.Set` is not checked
    dict2.Set(1, 1)
           ^
common/containerdb/storestate_test.go:99:11: Error return value of `array.Put` is not checked
    array.Put(obj1)
           ^
common/containerdb/storestate_test.go:100:11: Error return value of `array.Put` is not checked
    array.Put(obj2)
           ^
common/containerdb/vardb_test.go:51:8: Error return value of `db.Set` is not checked
    db.Set(int(1))
         ^
common/containerdb/vardb_test.go:58:11: Error return value of `db.Delete` is not checked
    db.Delete()
           ^
common/trie/trie_manager/manager_test.go:50:14: Error return value of `trie.Delete` is not checked
        trie.Delete([]byte(poolKey[i]))
                    ^
```

```
common/trie/trie_manager/manager_test.go:114:13: Error return value of `tree.Delete` is not checked
    tree.Delete([]byte("dogglesworth"))
                ^
common/trie/trie_manager/manager_test.go:120:13: Error return value of `tree.Delete` is not checked
    tree.Delete([]byte("dog"))
                ^
common/trie/trie_manager/manager_test.go:217:13: Error return value of `trie2.Reset` is not checked
    trie2.Reset(snapshot) // have doe, dog, dogglesworth
                ^
common/trie/trie_manager/manager_test.go:237:13: Error return value of `trie2.Reset` is not checked
    trie2.Reset(snapshot)
                ^
common/trie/trie_manager/manager_test.go:280:8: Error return value of `tr.Set` is not checked
    tr.Set([]byte{0x00}, []byte{0xFF})
          ^
common/trie/trie_manager/manager_test.go:281:8: Error return value of `tr.Set` is not checked
    tr.Set([]byte{0x00, 0x01}, []byte{0xFE})
          ^
common/trie/trie_manager/manager_test.go:282:8: Error return value of `tr.Set` is not checked
    tr.Set([]byte{0x00, 0x01, 0x00}, []byte{unchanged})
          ^
common/trie/trie_manager/manager_test.go:418:10: Error return value of `trie.Set` is not checked
    trie.Set([]byte(k), []byte(v))
           ^
common/trie/trie_manager/manager_test.go:422:13: Error return value of `trie.Delete` is not checked
    trie.Delete([]byte(k))
                ^
common/trie/trie_manager/manager_test.go:743:14: Error return value of `mutable.Set` is not checked
        mutable.Set([]byte(k), []byte(v))
                 ^
common/trie/trie_manager/manager_test.go:745:17: Error return value of `mutableObj.Set` is not checked
        mutableObj.Set([]byte(k), &Obj{value: []byte(v)})
                    ^
common/trie/trie_manager/manager_test.go:804:10: Error return value of `m1.Set` is not checked
            m1.Set([]byte(s), to)
                 ^
common/trie/trie_manager/manager_test.go:865:10: Error return value of `m1.Set` is not checked
            m1.Set([]byte(s), to)
                 ^
common/trie/trie_manager/manager_test.go:922:13: Error return value of `mutable.Set` is not checked
    mutable.Set([]byte("test"), []byte("value"))
                ^
common/trie/trie_manager/manager_test.go:1037:17: Error return value of `mutable.Reset` is not checked
            mutable.Reset(ss)
                        ^
service/eeproxy/javaee.go:208:21: Error return value of `e.cmd.Process.Wait` is not checked
        e.cmd.Process.Wait()
                          ^
service/eeproxy/manager.go:71:9: Error return value of `p.Kill` is not checked
        p.Kill()
              ^
service/platform/basic/chainscore.go:872:3: error is not nil (line 871) but it returns nil
        return nil
        ^
server/wsblock.go:126:2: error is not nil (line 68) but it returns nil
    return nil
    ^
server/wsevent.go:52:3: error is not nil (line 50) but it returns nil
        return nil
        ^
server/wsevent.go:118:2: error is not nil (line 78) but it returns nil
    return nil
    ^
havah/chainscore_basic.go:66:3: error is not nil (line 65) but it returns nil
        return nil
        ^
havah/hvh/extension.go:411:5: error is not nil (line 410) but it returns nil
                return nil
                ^
common/trie/ompt/mpt.go:378:4: error is not nil (line 375) but it returns nil
            return nil
```

```
            ^
service/contract/callhandler.go:314:3: error is not nil (line 312) but it returns nil
        return nil
        ^
chain/gs/genesis.go:185:3: SA5001: should check returned error before deferring fd.Close()
        defer fd.Close()
        ^
chain/taskbackup.go:149:3: SA5001: should check returned error before deferring fd.Close()
        defer fd.Close()
        ^
chain/imports/servicemanager.go:347:15: Error return value of `encoding/json.Marshal` is not checked: unsafe type `inter
            rjbs, _ := json.Marshal(mrjsn)
                       ^
chain/imports/servicemanager.go:350:16: Error return value of `encoding/json.Marshal` is not checked: unsafe type `inter
            nrjbs, _ := json.Marshal(mnrjsn)
                        ^
service/scoreapi/info.go:118:11: Error return value of `encoding/json.Marshal` is not checked: unsafe type `interface{}
    bs, _ := json.Marshal(jso)
           ^
service/transaction/transactionhandler.go:321:6: Error return value of `(*encoding/json.Encoder).Encode` is not checked
    _ = je.Encode(jso)
        ^
server/v3/validation_test.go:39:11: Error return value of `encoding/json.MarshalIndent` is not checked: unsafe type `int
    bs, _ := json.MarshalIndent(&callParam, "", "\t")
           ^
server/v3/validation_test.go:82:11: Error return value of `encoding/json.MarshalIndent` is not checked: unsafe type `int
    bs, _ := json.MarshalIndent(&txParam, "", "\t")
           ^
common/ipc/server.go:81:7: comparing with != will fail on wrapped errors. Use errors.Is to check for a specific error
            if err != io.EOF {
               ^
common/legacy/loopchain.go:35:6: comparing with == will fail on wrapped errors. Use errors.Is to check for a specific er
        if err == leveldb.ErrNotFound {
           ^
common/legacy/loopchain.go:42:6: comparing with == will fail on wrapped errors. Use errors.Is to check for a specific er
        if err == leveldb.ErrNotFound {
           ^
common/legacy/loopchain.go:53:6: comparing with == will fail on wrapped errors. Use errors.Is to check for a specific er
        if err == leveldb.ErrNotFound {
           ^
server/error.go:13:15: type assertion on error will fail on wrapped errors. Use errors.As to check for specific errors
    if je, ok := err.(*jsonrpc.Error); ok {
                 ^
server/error.go:17:15: type assertion on error will fail on wrapped errors. Use errors.As to check for specific errors
    if he, ok := err.(*echo.HTTPError); ok {
                 ^
server/wsblock.go:132:59: non-wrapping format verb for fmt.Errorf. Use `%w` to format errors
            return fmt.Errorf("fail to compile idx:%d, err:%v", i, err)
                                                                ^
common/codec/codec.go:646:21: comparing with != will fail on wrapped errors. Use errors.Is to check for a specific erro
            if err != nil && err != ErrNilValue {
                             ^
common/errors/errors.go:290:12: type assertion on error will fail on wrapped errors. Use errors.As to check for specific
        _, ok := err.(ErrorCoder)
                 ^
common/errors/errors.go:316:16: type switch on error will fail on wrapped errors. Use errors.As to check for specific er
    switch obj := err.(type) {
                  ^
common/errors/errors_test.go:91:6: comparing with != will fail on wrapped errors. Use errors.Is to check for a specific
        if c != e2 {
           ^
common/trie/cache/branchcache.go:136:46: non-wrapping format verb for fmt.Errorf. Use `%w` to format errors
            err = fmt.Errorf("fail to mkdir err:%+v", mkErr)
                                                     ^
common/trie/cache/branchcache.go:140:44: non-wrapping format verb for fmt.Errorf. Use `%w` to format errors
        err = fmt.Errorf("fail to stat err:%+v", sErr)
```

Recommendation

Define a strategy for handling errors of different severities, such as fatal errors versus basic log events. Use or create built-in language features or an external package that provides an easy-to-use API and define coding standards for the detection and handling of errors. Don't leave errors unhandled.

Use of inappropriate types

Int64 is used where uint64 is more appropriate.

ID	LOOP-1
Scope	Code Quality
Severity	**LOW**
Status	New

Details

- Type `Planet` could use `uint64` for the `height` field.

```
type Planet struct {
    dirty bool
    flags  PlanetFlag
    owner  *common.Address
    usdt   *big.Int // priceInUSDT
    price  *big.Int // priceInHVH
    height int64    // regTime
}
```

- Type `planetReward` could use `uint64` for `lastTN` field.

```
type planetReward struct {
    // Total received reward
    total *big.Int
    // The last term number when the reward is claimed
    lastTN int64
    // Reward to claim at this moment
    current *big.Int
}
```

- Type `WorldContext` could use `uint64` return type for the `Blockheight()` method.

```
type WorldContext interface {
    WorldState
    // ...
    BlockHeight() int64
    // ...
}
```

Recommendation

We assume some type selection decisions were made with regard to the `common/containerdb` package types. We suggest updating that package to support needed types. Using appropriate types could reduce some sanity checks throughout the code.