// HALBORN

Razor Network

Node Security Audit

Prepared by: **Halborn**

Date of Engagement: **May 15th, 2022 - June 8th, 2022**

Visit: **Halborn.com**

DOCUMENT REVISION HISTORY

VERSION	MODIFICATION	DATE	AUTHOR
0.1	Document Creation	05/15/2022	Gokberk Gulgun
0.2	Document Updates	05/25/2022	Gokberk Gulgun
0.3	Document Review	06/08/2022	Gabi Urrutia
1.0	Remediation Plan	07/21/2022	Gokberk Gulgun
1.1	Remediation Plan Review	07/22/2022	Gabi Urrutia

CONTACTS

CONTACT	COMPANY	EMAIL
Rob Behnke	Halborn	Rob.Behnke@halborn.com
Steven Walbroehl	Halborn	Steven.Walbroehl@halborn.com
Gabi Urrutia	Halborn	Gabi.Urrutia@halborn.com
Gokberk Gulgun	Halborn	Gokberk.Gulgun@halborn.com

EXECUTIVE OVERVIEW

1.1 INTRODUCTION

Razor Network engaged Halborn to conduct a security assessment on their **Golang** Node implementation beginning on May 15th and ending on June 8th, 2022.

The security assessment was scoped to the GitHub repository of Go. An audit of the security risk and implications regarding the changes introduced by the development team at Razor Network prior to its production release, shortly following the assessment's deadline.

Though this security audit's outcome is satisfactory, only the most essential aspects were tested and verified to achieve objectives and deliverable set in the scope due to time and resource constraints. It is essential to note the use of the best practices for secure structure development.

1.2 AUDIT SUMMARY

The team at Halborn was provided nearly four weeks for the engagement and assigned two full-time security engineers to audit the security of the Razor Go Repository. The security engineer is blockchain and smart-contract security experts with advanced penetration testing, smart-contract hacking, and deep knowledge of multiple blockchain protocols.

The purpose of this audit to achieve the following:

- Ensure that Razor Node functions are intended.
- Identify potential security issues with the Razor Node.

In summary, Halborn identified few security risks that were mostly addressed by Razor Team.

1.3 TEST APPROACH & METHODOLOGY

Halborn performed a combination of manual and automated security testing to balance efficiency, timeliness, practicality, and accuracy in regard to the scope of the Razor Node. While manual testing is recommended to uncover flaws in logic, process, and implementation; automated testing techniques help enhance coverage of structures and can quickly identify items that do not follow security best practices. The following phases and associated tools were used throughout the term of the audit:

- Research into architecture and purpose.
- Static Analysis of security for scoped repository, and imported functions. (staticcheck, gosec, unconvert, ineffassign and semgrep)
- Manual Assessment for discovering security vulnerabilities on codebase.
- Ensuring correctness of the codebase.
- Dynamic Analysis on Razor Go functions and data types.
- Property based coverage-guided fuzzing. (gofuzz).

RISK METHODOLOGY:

Vulnerabilities or issues observed by Halborn are ranked based on the risk assessment methodology by measuring the **LIKELIHOOD** of a security incident and the **IMPACT** should an incident occur. This framework works for communicating the characteristics and impacts of technology vulnerabilities. The quantitative model ensures repeatable and accurate measurement while enabling users to see the underlying vulnerability characteristics that were used to generate the Risk scores. For every vulnerability, a risk level will be calculated on a scale of 5 to 1 with 5 being the highest likelihood or impact.

RISK SCALE - LIKELIHOOD

5 - Almost certain an incident will occur.
4 - High probability of an incident occurring.
3 - Potential of a security incident in the long term.

2 - Low probability of an incident occurring.
1 - Very unlikely issue will cause an incident.

RISK SCALE - IMPACT

5 - May cause devastating and unrecoverable impact or loss.
4 - May cause a significant level of impact or loss.
3 - May cause a partial impact or loss to many.
2 - May cause temporary impact or loss.
1 - May cause minimal or un-noticeable impact.

The risk level is then calculated using a sum of these two values, creating a value of 10 to 1 with 10 being the highest level of security risk.

CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL

10 - CRITICAL
9 - 8 - HIGH
7 - 6 - MEDIUM
5 - 4 - LOW
3 - 1 - VERY LOW AND INFORMATIONAL

1.4 SCOPE

IN-SCOPE:
The security assessment was scoped to Razor Go repository.

Commit ID: 41d326ad9374af7042931ddedf671ad20fbb1039

OUT-OF-SCOPE:
External libraries.

FIX Commit TREE/ID :

Commit ID: Commit ID

2. ASSESSMENT SUMMARY & FINDINGS OVERVIEW

CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL
0	0	2	3	5

LIKELIHOOD

IMPACT

🟨	🟨	🟥	🟥	🟪
🟩	🟨 (HAL-01)	🟨	🟥	🟥
🟩 (HAL-03) (HAL-05)	🟩	🟨 (HAL-02)	🟨	🟥
⬜	🟩 (HAL-04)	🟩	🟨	🟨
⬜ (HAL-06) (HAL-07) (HAL-08) (HAL-09) (HAL-10)	⬜	🟩	🟩	🟨

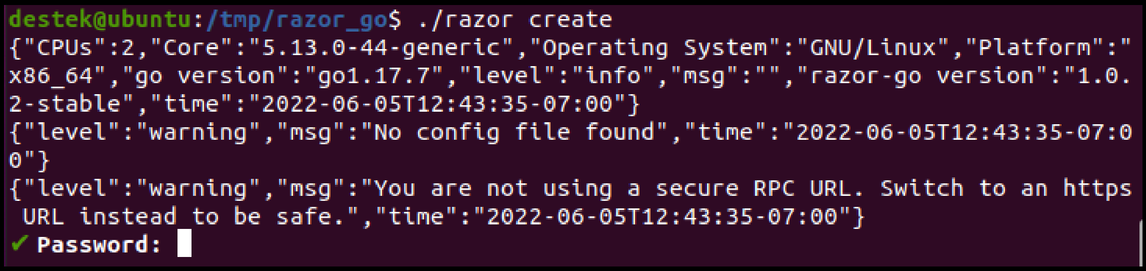
SECURITY ANALYSIS	RISK LEVEL	REMEDIATION DATE
WEAK PASSWORD POLICY	Medium	SOLVED - 07/21/2022
USE OF WEAK RANDOM GENERATOR	Medium	SOLVED - 07/21/2022
DOCKER IMAGE RUN AS ROOT	Low	RISK ACCEPTED
WEAK TLS CONFIGURATION	Low	SOLVED - 07/21/2022
GAS PRICE IS NOT CALCULATED DYNAMICALLY	Low	SOLVED - 07/21/2022
HTTP DEFAULT TRANSPORT FEATURES ARE NOT DEFINED	Informational	ACKNOWLEDGED
MISSING GO COMPILER BUILD DIRECTIVES	Informational	SOLVED - 07/21/2022
IOUTIL READALL FUNCTION IS DEPRECATED AFTER GO 1.16	Informational	SOLVED - 07/21/2022
PASSWORD STORED CLEAR TEXT IN THE MEMORY	Informational	ACKNOWLEDGED
CONTRACT CONFIGURATION FILE CAN BE DESIGNED MORE MODULAR	Informational	ACKNOWLEDGED

FINDINGS & TECH DETAILS

3.1 (HAL-01) WEAK PASSWORD POLICY - MEDIUM

A key storage mechanism is only as strong as its password. For this reason, it is important to require users to have strong passwords. Lack of password complexity significantly reduces the search space when trying to guess user's passwords, making brute-force attacks easier. The node does not validate that users should have strong passwords on the account generation, which makes it easier for attackers to compromise user accounts.

Screenshot:



```
destek@ubuntu:/tmp/razor_go$ ./razor create
{"CPUs":2,"Core":"5.13.0-44-generic","Operating System":"GNU/Linux","Platform":"
x86_64","go version":"go1.17.7","level":"info","msg":"","razor-go version":"1.0.
2-stable","time":"2022-06-05T12:43:35-07:00"}
{"level":"warning","msg":"No config file found","time":"2022-06-05T12:43:35-07:0
0"}
{"level":"warning","msg":"You are not using a secure RPC URL. Switch to an https
 URL instead to be safe.","time":"2022-06-05T12:43:35-07:00"}
✓ Password:
```

Recommendation:

- Allow all characters to be used for passwords to avoid shortening the key space for brute-force guessing.
- Disallow short password lengths. 12 characters is generally considered a good minimum password length.
- Allow for a large maximum password length.

Remediation Plan:

SOLVED: The Razor Team fixed the issue in the following commit.

3.2 (HAL-02) USE OF WEAK RANDOM GENERATOR - MEDIUM

Description:

When a non-cryptographic PRNG is used in a cryptographic context, it can expose the cryptography to certain types of attacks. Often a pseudo-random number generator (PRNG) is not designed for cryptography. Every so often, a poor source of randomness is enough or preferable for algorithms that use random numbers. Weak generators generally take less processing power and/or do not use the precious, finite, entropy sources on a system. While such PRNGs might have very useful features, these same features could be used to break the cryptography.

Code Location:

Listing 1: Locations

```
./cmd/dispute.go:14:     "math/rand"
./cmd/propose.go:12:     "math/rand"
./utils/math.go:8:   "math/rand"
```

Risk Level:

Likelihood - 3
Impact - 3

Recommendation:

Consider replacing math/rand with crypto/rand.

Remediation Plan:

SOLVED: The Razor Team fixed the issue in the following commit.

3.3 (HAL-03) DOCKER IMAGE RUN AS ROOT - LOW

Description:

Docker containers usually run with root privileges by default. This allows for unrestricted container management, which means you can do things like to install system packages, edit configuration files, bind privileged ports. During the static analysis, it has been observed that docker image is maintained via root user.

Code Location:

Listing 2: Dockerfile

```
FROM golang:1.17-alpine AS go
FROM ethereum/client-go:alltools-v1.10.7 AS ethereum

FROM node:16.2.0-alpine AS builder

COPY --from=ethereum /usr/local/bin/abigen /usr/local/bin/
COPY --from=go /usr/local/go/ /usr/local/go/

## Attaching current dir to workdir
WORKDIR /app
COPY . /app

## Install and Cleanup

RUN PATH="/usr/local/go/bin:${PATH}" \
    && apk add --update --no-cache python3 && ln -sf python3 /usr/bin/python \
    && apk add --update make gcc musl musl-dev g++ libc-dev bash linux-headers \
    && apk add --no-cache jq \
    && npm install \
    && npm run build-noargs \
    && cp build/bin/razor /usr/local/bin/


```

```
FROM alpine:latest
RUN apk add --update bash
COPY --from=builder /usr/local/bin/razor /usr/local/bin/
ENTRYPOINT [ "razor" ]
```

Risk Level:

Likelihood - 1
Impact - 3

Recommendation:

It is recommended to build dockerfile and run container as a non-root user.

Listing 3: Reference

```
USER 1001: this is a non-root user UID, and here it is assigned to
↳  the image to run the current container as an unprivileged user.
↳ By doing so, the added security and other restrictions mentioned
↳ above are applied to the container.
```

Remediation Plan:

RISK ACCEPTED: The Razor Team accepted the risk of this finding.

3.4 (HAL-04) WEAK TLS CONFIGURATION - LOW

Description:

Disabling TLS certificate checking, an application may be vulnerable to man-in-the-middle attacks.

Code Location:

metrics/server.go

Listing 4: TLS Configuration

```
metrics/server.go
        Found an HTTP server without TLS. Use 'http.
↳ ListenAndServeTLS' instead.
         Fix  http.ListenAndServeTLS(portNumber, certFile,
↳ keyFile, nil)
        23 return http.ListenAndServe(portNumber, nil)
}
```

Risk Level:

Likelihood - 2
Impact - 2

Recommendation:

It is not recommended to use InsecureSkipVerify in production code.

Remediation Plan:

SOLVED: The Razor Team fixed the issue in the following commit.

3.5 (HAL-05) GAS PRICE IS NOT CALCULATED DYNAMICALLY - LOW

Description:

During the code review, It has been noticed gas price is taken as an argument on the node. The system is not designed with dynamic gas calculation. That can leads to transaction failure on the node side.

Code Location:

Listing 5: CLI Parameters

```go
func init() {
    cobra.OnInitialize(initConfig)

    rootCmd.PersistentFlags().StringVarP(&Provider, "provider", "p
↳ ", "", "provider name")
    rootCmd.PersistentFlags().Int64VarP(&ChainId, "chainId", "c",
↳ 0, "chainId")
    rootCmd.PersistentFlags().Float32VarP(&GasMultiplier, "
↳ gasmultiplier", "g", -1, "gas multiplier value")
    rootCmd.PersistentFlags().Int32VarP(&BufferPercent, "buffer",
↳ "b", 0, "buffer percent")
    rootCmd.PersistentFlags().Int32VarP(&WaitTime, "wait", "w",
↳ -1, "wait time")
    rootCmd.PersistentFlags().Int32VarP(&GasPrice, "gasprice", "",
↳  -1, "gas price")
    rootCmd.PersistentFlags().StringVarP(&LogLevel, "logLevel", ""
↳ , "", "log level")
    rootCmd.PersistentFlags().Float32VarP(&GasLimitMultiplier, "
↳ gasLimit", "", -1, "gas limit percentage increase")
    rootCmd.PersistentFlags().StringVarP(&LogFile, "logFile", "",
↳ "", "name of log file")
    rootCmd.Flags().BoolP("toggle", "t", false, "Help message for
↳ toggle")
}
```

Risk Level:

Likelihood - 1
Impact - 3

Recommendation:

It is recommended to calculate gas dynamically. **eth_gasPrice** and **eth_-MaxPriorityFeePerGas** should be considered on the implementation.

Remediation Plan:

SOLVED: The Razor Team solved the issue with dynamic calculation of gas price in the code.

3.6 (HAL-06) HTTP DEFAULT TRANSPORT FEATURES ARE NOT DEFINED - INFORMATIONAL

Description:

By default, the golang http client performs the connection pooling. When the request completes, that connection remains open until the idle connection timeout (default is 90 seconds) If another request came, that uses the same established connection instead of creating a new connection, after the idle connection time, the connection will return to pool. By increasing connection per host and total number of idle connections, this will increase the performance an serve more requests with minimal server resources.

Code Location:

Listing 6: HTTP Transport Features

```
  utils/api.go
func (*UtilsStruct) GetDataFromAPI(url string) ([]byte, error) {
    client := http.Client{
        Timeout: 60 * time.Second,
    }
    var body []byte
    err := retry.Do(
        func() error {
            response, err := client.Get(url)
            if err != nil {
                return err
            }
            defer response.Body.Close()
            if response.StatusCode != 200 {
                return errors.New("unable to reach API")
            }
            body, err = IoutilInterface.ReadAll(response.Body)
            if err != nil {
                return err
            }
```

```
                return nil
            }, RetryInterface.RetryAttempts(core.MaxRetries))
        if err != nil {
            return nil, err
        }
        return body, nil
}
```

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

Connection pool size and connection per host count can be increased as per server resources and requirements.

Listing 7: HTTP Transport Features

```
t := http.DefaultTransport(*http.Transport).Clone()
t.MaxIdleConns = 100
t.MaxConnsPerHost = 100 t.MaxIdleConnsPerHost = 100

httpClient = &http.Client{
    Timeout: 10 * time.Second,
    Transport: t,
}
```

Remediation Plan:

ACKNOWLEDGED: The Razor Team acknowledged this issue.

3.7 (HAL-07) MISSING GO COMPILER BUILD DIRECTIVES - INFORMATIONAL

Description:

During the code review, It has been observed that some of the 'Go' compiler build flags are not configured. The use of compiler flags and compiler sequences can optimize and improve the performance of specific types of the applications.

Enabling compiler build flags could make binary build faster and outputs a smaller and probably more efficient binary. Therefore, the flags should be reviewed and enabled.

Code Location:

Listing 8: Makefile

```
GO ?= go
SHELL = /bin/bash
BIN_DIR = ./build/bin
RAZOR = ${BIN_DIR}/razor

all: fetch_bindings install_razor set_config
build: install_razor set_config
build-noargs: fetch_bindings install_razor
setup: fetch_bindings

fetch_bindings:
    @echo "Installing contract dependencies..."
    @echo ""
    @${SHELL} generate-bindings.sh
    @echo "Contract bindings generated...."
    @echo ""

install_razor:
    @echo "Installing razor node...."
    ${GO} build -o ./build/bin/razor main.go
    @echo "Razor node installed."
    @echo ""
```

```

set_config:
    @echo "Setup initial config"
    @${SHELL} config.sh
    @echo ""
    @echo "Razor node is set up and ready to use"

```

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

Consider reviewing all compiler flags and use relevant flags on the build progress.

Listing 9: Flags

```
-a
    Force rebuilding of packages that are already up-to-date.
-ldflags -s -w
    The -w turns off DWARF debugging information.
    The -s turns off generation of the Go symbol table.
- trimpath
     The -trimpath Remove all file system paths from the resulting
↳ executable.
- gcflags
     Arguments to pass on each go tool compile invocation.
```

Remediation Plan:

SOLVED: The Razor Team fixed the issue in the following commit.

3.8 (HAL-08) IOUTIL READALL FUNCTION IS DEPRECATED AFTER GO 1.16 - INFORMATIONAL

Description:

With the two proposals accepted (golang/go#42026 and golang/go#40025), the package ioutil will be deprecated, and new code is encouraged to use the respective implementations in the packages io and os.

Code Location:

Listing 10: IOUTIL Usage

```
./utils/struct-utils.go:11: "io/ioutil"
./utils/struct-utils.go:399:    return ioutil.ReadAll(body)
./utils/struct-utils.go:404:    return ioutil.ReadFile(filename)
./utils/struct-utils.go:409:    return ioutil.WriteFile(filename,
↳ data, perm)
./accounts/accountUtils.go:9:   "io/ioutil"
./accounts/accountUtils.go:56:  return ioutil.ReadFile(filename)
```

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

Consider changing **ioutil.ReadAll(...)** with **io.ReadAll(...)**.

Remediation Plan:

SOLVED: The Razor Team fixed the issue with the following commit.

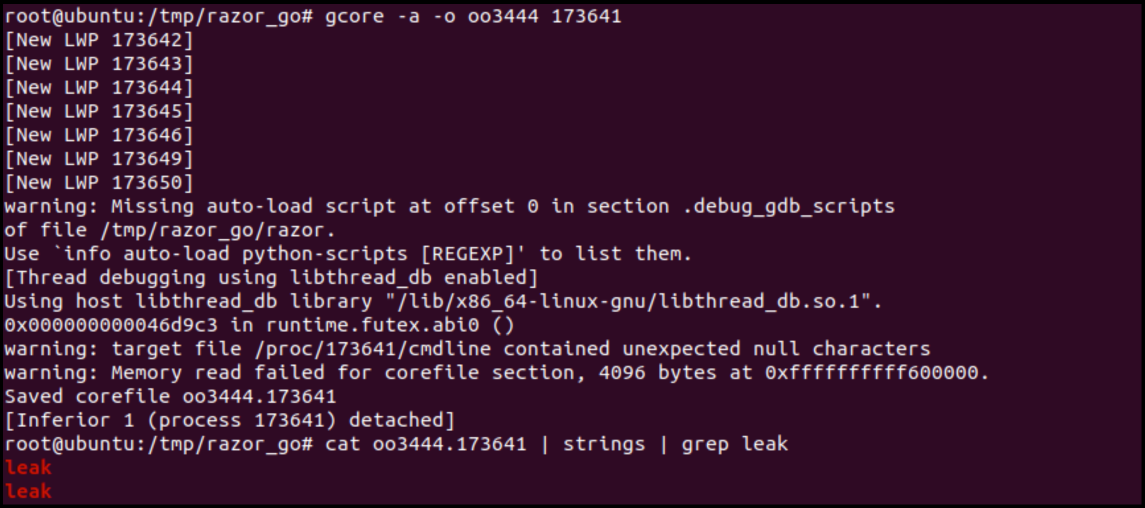
3.9 (HAL-09) PASSWORD STORED CLEAR TEXT IN THE MEMORY - INFORMATIONAL

Description:

The Razor executable store password as a clear text in the memory. In some cases, it is possible for an attacker to acquire the necessary privileges to dump the memory contents of an arbitrary user process by exploiting a vulnerable service. In summary, the password should not be reachable by dumping the memory.

MEMORY ANALYSIS:

During the test, gcore and strings tools were used to dumping the memory. The following screenshot explains what an attacker can get from the system without reading any important files.



```
root@ubuntu:/tmp/razor_go# gcore -a -o oo3444 173641
[New LWP 173642]
[New LWP 173643]
[New LWP 173644]
[New LWP 173645]
[New LWP 173646]
[New LWP 173649]
[New LWP 173650]
warning: Missing auto-load script at offset 0 in section .debug_gdb_scripts
of file /tmp/razor_go/razor.
Use `info auto-load python-scripts [REGEXP]' to list them.
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".
0x000000000046d9c3 in runtime.futex.abi0 ()
warning: target file /proc/173641/cmdline contained unexpected null characters
warning: Memory read failed for corefile section, 4096 bytes at 0xffffffffff600000.
Saved corefile oo3444.173641
[Inferior 1 (process 173641) detached]
root@ubuntu:/tmp/razor_go# cat oo3444.173641 | strings | grep leak
leak
leak
```

Figure 1: Password Dump

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

It is strongly recommended using the memguard software to store secure credentials in the memory.

Remediation Plan:

ACKNOWLEDGED: The Razor Team acknowledged this issue.

3.10 (HAL-10) CONTRACT CONFIGURATION FILE CAN BE DESIGNED MORE MODULAR - INFORMATIONAL

Description:

In the Razor Node, Contracts are parsed from the configuration file. However, If the Node created wrong Configuration File, Node could not interact with the chain.

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

For the contracts addresses, one HTTP API can be developed, and It can be parsed with HTTP request.

Remediation Plan:

ACKNOWLEDGED: The Razor Team acknowledged this issue.

AUTOMATED TESTING

Description:

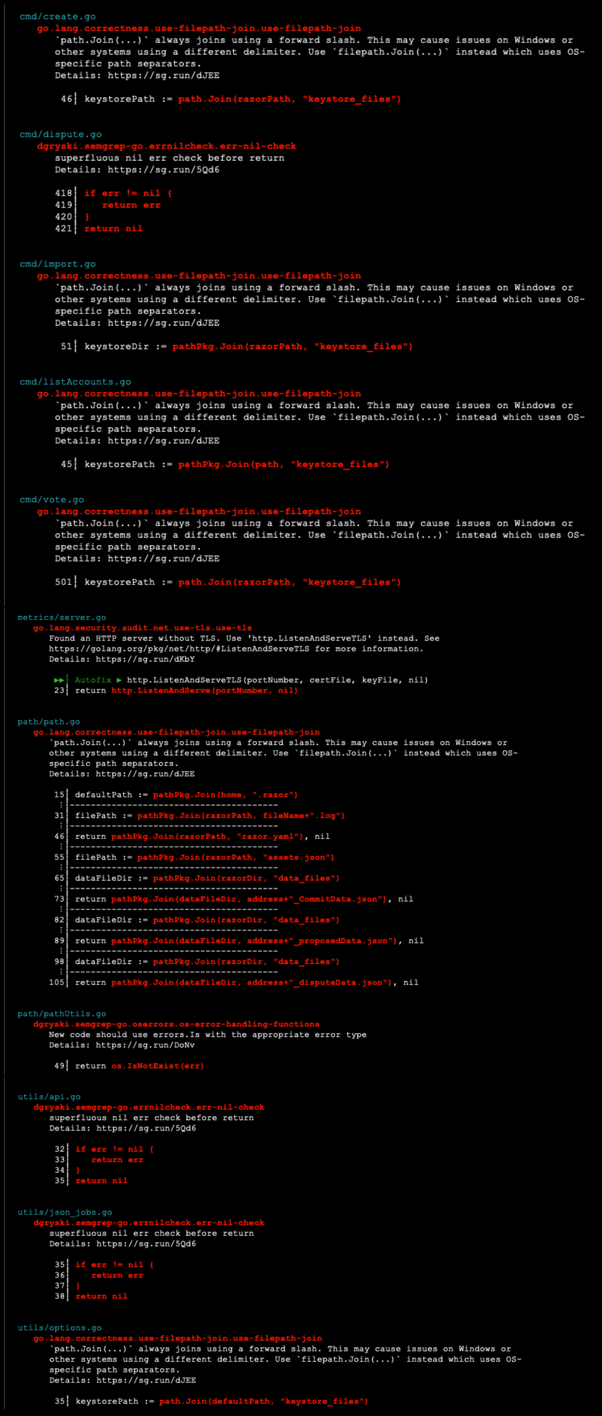
Halborn used automated testing techniques to enhance coverage of certain areas of the scoped component. Among the tools used were staticcheck, gosec ineffassign, unconvert and LGTM. After Halborn verified all the contracts and scoped structures in the repository and was able to compile them correctly, these tools were leveraged on scoped structures. With these tools, Halborn can statically verify security related issues across the entire codebase.

Semgrep - Security Analysis Output Sample:

Listing 11: Rule Set

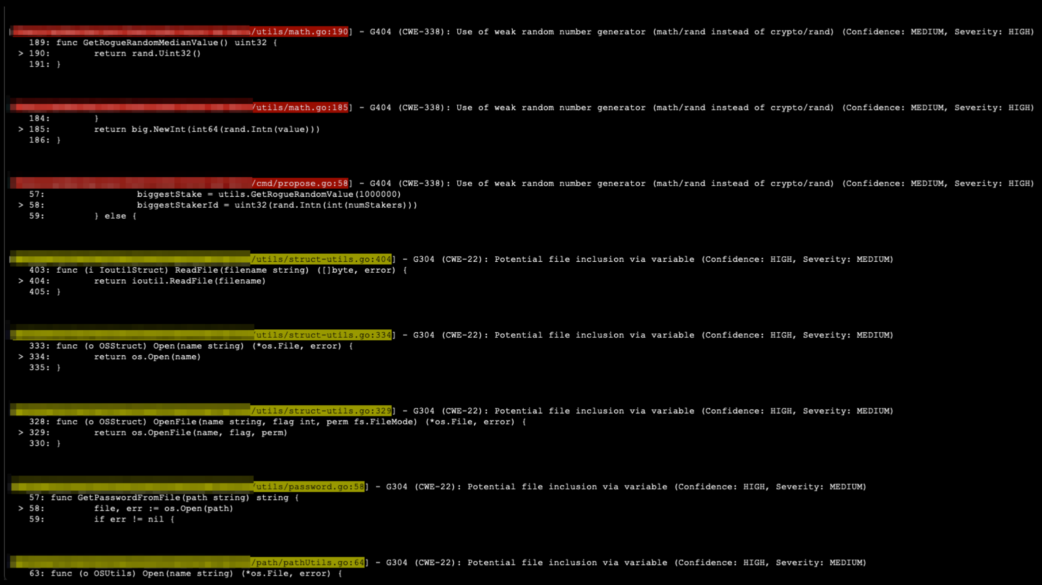
```
semgrep --config "p/dgryski.semgrep-go" x --exclude='*_test.go' --
↳ max-lines-per-finding 1000 --no-git-ignore -o dgryski.semgrep
semgrep --config "p/owasp-top-ten"       x --exclude='*_test.go' --
↳ max-lines-per-finding 1000 --no-git-ignore -o owasp-top-ten.
↳ semgrep
semgrep --config "p/r2c-security-audit" x --exclude='*_test.go' --
↳ max-lines-per-finding 1000 --no-git-ignore -o r2c-security-audit.
↳ semgrep
semgrep --config "p/r2c-ci"              x --exclude='*_test.go' --
↳ max-lines-per-finding 1000 --no-git-ignore -o r2c-ci.semgrep
semgrep --config "p/ci"                  x --exclude='*_test.go' --
↳ max-lines-per-finding 1000 --no-git-ignore -o ci.semgrep
semgrep --config "p/golang"              x --exclude='*_test.go' --
↳ max-lines-per-finding 1000 --no-git-ignore -o golang.semgrep
semgrep --config "p/trailofbits"         x --exclude='*_test.go' --
↳ max-lines-per-finding 1000 --no-git-ignore -o trailofbits.semgrep
```

Semgrep Results:

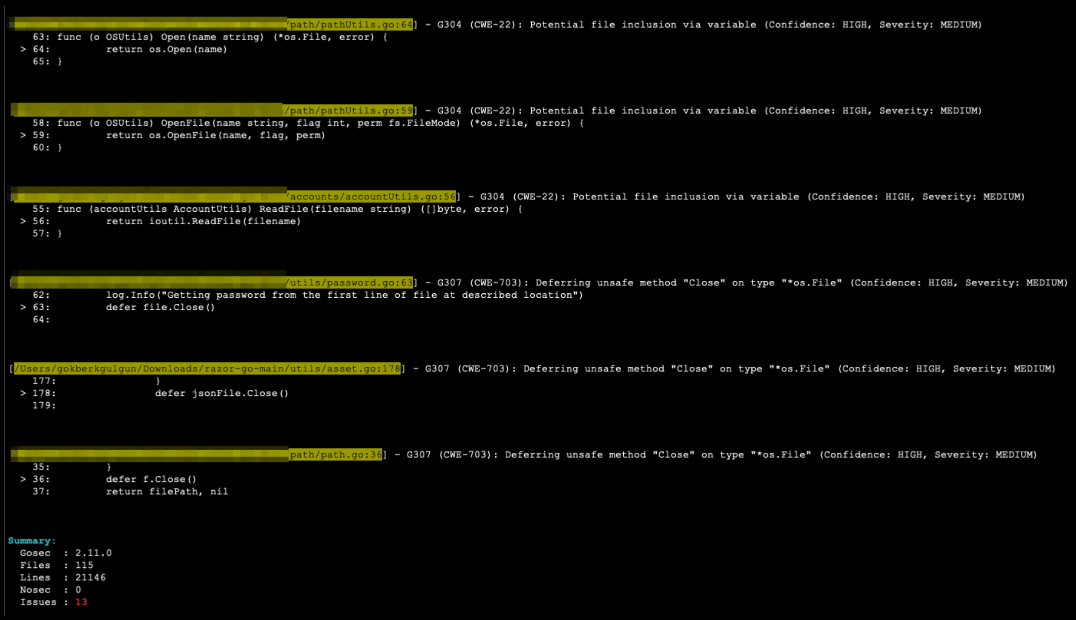


```
cmd/create.go
   go.lang.correctness.use-filepath-join.use-filepath-join
      `path.Join(...)` always joins using a forward slash. This may cause issues on Windows or
      other systems using a different delimiter. Use `filepath.Join(...)` instead which uses OS-
      specific path separators.
      Details: https://sg.run/dJEE

       46┆ keystorePath := path.Join(razorPath, "keystore_files")


cmd/dispute.go
   dgryski.semgrep-go.errnilcheck.err-nil-check
      superfluous nil err check before return
      Details: https://sg.run/5Qd6

      418┆ if err != nil {
      419┆    return err
      420┆ }
      421┆ return nil


cmd/import.go
   go.lang.correctness.use-filepath-join.use-filepath-join
      `path.Join(...)` always joins using a forward slash. This may cause issues on Windows or
      other systems using a different delimiter. Use `filepath.Join(...)` instead which uses OS-
      specific path separators.
      Details: https://sg.run/dJEE

       51┆ keystoreDir := pathPkg.Join(razorPath, "keystore_files")


cmd/listAccounts.go
   go.lang.correctness.use-filepath-join.use-filepath-join
      `path.Join(...)` always joins using a forward slash. This may cause issues on Windows or
      other systems using a different delimiter. Use `filepath.Join(...)` instead which uses OS-
      specific path separators.
      Details: https://sg.run/dJEE

       45┆ keystorePath := pathPkg.Join(path, "keystore_files")


cmd/vote.go
   go.lang.correctness.use-filepath-join.use-filepath-join
      `path.Join(...)` always joins using a forward slash. This may cause issues on Windows or
      other systems using a different delimiter. Use `filepath.Join(...)` instead which uses OS-
      specific path separators.
      Details: https://sg.run/dJEE

      501┆ keystorePath := path.Join(razorPath, "keystore_files")


metrics/server.go
   go.lang.security.audit.net.use-tls.use-tls
      Found an HTTP server without TLS. Use 'http.ListenAndServeTLS' instead. See
      https://golang.org/pkg/net/http/#ListenAndServeTLS for more information.
      Details: https://sg.run/dKbY

       ▶▶┆ Autofix ▶ http.ListenAndServeTLS(portNumber, certFile, keyFile, nil)
       23┆ return http.ListenAndServe(portNumber, nil)


path/path.go
   go.lang.correctness.use-filepath-join.use-filepath-join
      `path.Join(...)` always joins using a forward slash. This may cause issues on Windows or
      other systems using a different delimiter. Use `filepath.Join(...)` instead which uses OS-
      specific path separators.
      Details: https://sg.run/dJEE

       15┆ defaultPath := pathPkg.Join(home, ".razor")
        ⋮┆----------------------------------------
       31┆ filePath := pathPkg.Join(razorPath, fileName+".log")
        ⋮┆----------------------------------------
       46┆ return pathPkg.Join(razorPath, "razor.yaml"), nil
        ⋮┆----------------------------------------
       55┆ filePath := pathPkg.Join(razorPath, "assets.json")
        ⋮┆----------------------------------------
       65┆ dataFileDir := pathPkg.Join(razorDir, "data_files")
        ⋮┆----------------------------------------
       73┆ return pathPkg.Join(dataFileDir, address+"_CommitData.json"), nil
        ⋮┆----------------------------------------
       82┆ dataFileDir := pathPkg.Join(razorDir, "data_files")
        ⋮┆----------------------------------------
       89┆ return pathPkg.Join(dataFileDir, address+"_proposedData.json"), nil
        ⋮┆----------------------------------------
       98┆ dataFileDir := pathPkg.Join(razorDir, "data_files")
        ⋮┆----------------------------------------
      105┆ return pathPkg.Join(dataFileDir, address+"_disputeData.json"), nil


path/pathUtils.go
   dgryski.semgrep-go.oserrors.os-error-handling-functions
      New code should use errors.Is with the appropriate error type
      Details: https://sg.run/DoNv

       49┆ return os.IsNotExist(err)


utils/api.go
   dgryski.semgrep-go.errnilcheck.err-nil-check
      superfluous nil err check before return
      Details: https://sg.run/5Qd6

       32┆ if err != nil {
       33┆    return err
       34┆ }
       35┆ return nil


utils/json_jobs.go
   dgryski.semgrep-go.errnilcheck.err-nil-check
      superfluous nil err check before return
      Details: https://sg.run/5Qd6

       35┆ if err != nil {
       36┆    return err
       37┆ }
       38┆ return nil


utils/options.go
   go.lang.correctness.use-filepath-join.use-filepath-join
      `path.Join(...)` always joins using a forward slash. This may cause issues on Windows or
      other systems using a different delimiter. Use `filepath.Join(...)` instead which uses OS-
      specific path separators.
      Details: https://sg.run/dJEE

       35┆ keystorePath := path.Join(defaultPath, "keystore_files")
```

Gosec - Security Analysis Output Sample:

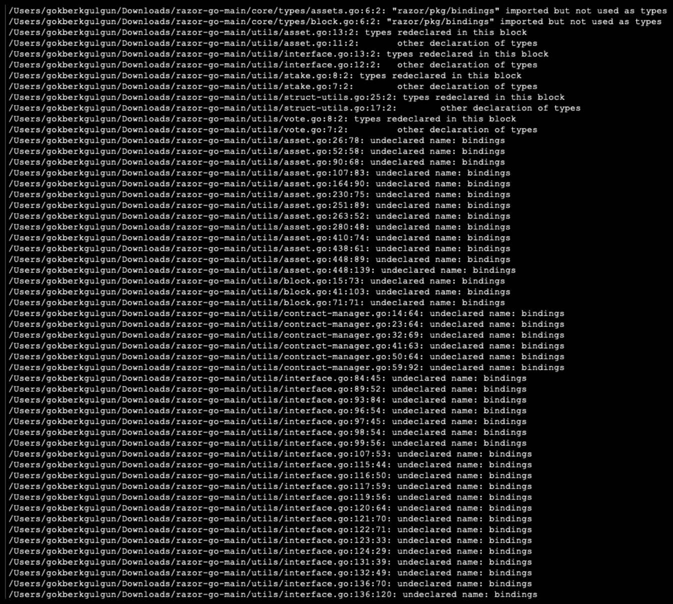


```
[[redacted]/utils/math.go:190] - G404 (CWE-338): Use of weak random number generator (math/rand instead of crypto/rand) (Confidence: MEDIUM, Severity: HIGH)
   189: func GetRogueRandomMedianValue() uint32 {
 > 190:         return rand.Uint32()
   191: }


[redacted]/utils/math.go:185] - G404 (CWE-338): Use of weak random number generator (math/rand instead of crypto/rand) (Confidence: MEDIUM, Severity: HIGH)
   184:         }
 > 185:         return big.NewInt(int64(rand.Intn(value)))
   186: }


[redacted]/cmd/propose.go:58] - G404 (CWE-338): Use of weak random number generator (math/rand instead of crypto/rand) (Confidence: MEDIUM, Severity: HIGH)
   57:                  biggestStake = utils.GetRogueRandomValue(1000000)
 > 58:                  biggestStakerId = uint32(rand.Intn(int(numStakers)))
   59:          } else {


[[redacted]/utils/struct-utils.go:404] - G304 (CWE-22): Potential file inclusion via variable (Confidence: HIGH, Severity: MEDIUM)
   403: func (i IoutilStruct) ReadFile(filename string) ([]byte, error) {
 > 404:         return ioutil.ReadFile(filename)
   405: }


[redacted]utils/struct-utils.go:334] - G304 (CWE-22): Potential file inclusion via variable (Confidence: HIGH, Severity: MEDIUM)
   333: func (o OSStruct) Open(name string) (*os.File, error) {
 > 334:         return os.Open(name)
   335: }


[redacted]/utils/struct-utils.go:329] - G304 (CWE-22): Potential file inclusion via variable (Confidence: HIGH, Severity: MEDIUM)
   328: func (o OSStruct) OpenFile(name string, flag int, perm fs.FileMode) (*os.File, error) {
 > 329:         return os.OpenFile(name, flag, perm)
   330: }


[redacted]utils/password.go:58] - G304 (CWE-22): Potential file inclusion via variable (Confidence: HIGH, Severity: MEDIUM)
   57: func GetPasswordFromFile(path string) string {
 > 58:         file, err := os.Open(path)
   59:         if err != nil {


[redacted]/path/pathUtils.go:64] - G304 (CWE-22): Potential file inclusion via variable (Confidence: HIGH, Severity: MEDIUM)
   63: func (o OSUtils) Open(name string) (*os.File, error) {
```



```
[redacted]path/pathUtils.go:64] - G304 (CWE-22): Potential file inclusion via variable (Confidence: HIGH, Severity: MEDIUM)
   63: func (o OSUtils) Open(name string) (*os.File, error) {
 > 64:         return os.Open(name)
   65: }


[redacted]/path/pathUtils.go:59] - G304 (CWE-22): Potential file inclusion via variable (Confidence: HIGH, Severity: MEDIUM)
   58: func (o OSUtils) OpenFile(name string, flag int, perm fs.FileMode) (*os.File, error) {
 > 59:         return os.OpenFile(name, flag, perm)
   60: }


[[redacted]accounts/accountUtils.go:56] - G304 (CWE-22): Potential file inclusion via variable (Confidence: HIGH, Severity: MEDIUM)
   55: func (accountUtils AccountUtils) ReadFile(filename string) ([]byte, error) {
 > 56:         return ioutil.ReadFile(filename)
   57: }


[[redacted]/utils/password.go:63] - G307 (CWE-703): Deferring unsafe method "Close" on type "*os.File" (Confidence: HIGH, Severity: MEDIUM)
   62:         log.Info("Getting password from the first line of file at described location")
 > 63:         defer file.Close()
   64:


[/Users/gokberkgulgun/Downloads/razor-go-main/utils/asset.go:178] - G307 (CWE-703): Deferring unsafe method "Close" on type "*os.File" (Confidence: HIGH, Severity: MEDIUM)
   177:                 }
 > 178:                 defer jsonFile.Close()
   179:


[redacted]path/path.go:36] - G307 (CWE-703): Deferring unsafe method "Close" on type "*os.File" (Confidence: HIGH, Severity: MEDIUM)
   35:         }
 > 36:         defer f.Close()
   37:         return filePath, nil


Summary:
  Gosec  : 2.11.0
  Files  : 115
  Lines  : 21146
  Nosec  : 0
  Issues : 13
```

Unconvert - Security Analysis Output Sample:



```
/Users/gokberkgulgun/Downloads/razor-go-main/core/types/assets.go:6:2: "razor/pkg/bindings" imported but not used as types
/Users/gokberkgulgun/Downloads/razor-go-main/core/types/block.go:6:2: "razor/pkg/bindings" imported but not used as types
/Users/gokberkgulgun/Downloads/razor-go-main/utils/asset.go:13:2: types redeclared in this block
/Users/gokberkgulgun/Downloads/razor-go-main/utils/asset.go:11:2:       other declaration of types
/Users/gokberkgulgun/Downloads/razor-go-main/utils/interface.go:13:2: types redeclared in this block
/Users/gokberkgulgun/Downloads/razor-go-main/utils/interface.go:12:2:   other declaration of types
/Users/gokberkgulgun/Downloads/razor-go-main/utils/stake.go:8:2: types redeclared in this block
/Users/gokberkgulgun/Downloads/razor-go-main/utils/stake.go:7:2:        other declaration of types
/Users/gokberkgulgun/Downloads/razor-go-main/utils/struct-utils.go:25:2: types redeclared in this block
/Users/gokberkgulgun/Downloads/razor-go-main/utils/struct-utils.go:17:2:        other declaration of types
/Users/gokberkgulgun/Downloads/razor-go-main/utils/vote.go:8:2: types redeclared in this block
/Users/gokberkgulgun/Downloads/razor-go-main/utils/vote.go:7:2:         other declaration of types
/Users/gokberkgulgun/Downloads/razor-go-main/utils/asset.go:26:78: undeclared name: bindings
/Users/gokberkgulgun/Downloads/razor-go-main/utils/asset.go:52:58: undeclared name: bindings
/Users/gokberkgulgun/Downloads/razor-go-main/utils/asset.go:90:68: undeclared name: bindings
/Users/gokberkgulgun/Downloads/razor-go-main/utils/asset.go:107:83: undeclared name: bindings
/Users/gokberkgulgun/Downloads/razor-go-main/utils/asset.go:164:90: undeclared name: bindings
/Users/gokberkgulgun/Downloads/razor-go-main/utils/asset.go:230:75: undeclared name: bindings
/Users/gokberkgulgun/Downloads/razor-go-main/utils/asset.go:251:89: undeclared name: bindings
/Users/gokberkgulgun/Downloads/razor-go-main/utils/asset.go:263:52: undeclared name: bindings
/Users/gokberkgulgun/Downloads/razor-go-main/utils/asset.go:280:48: undeclared name: bindings
/Users/gokberkgulgun/Downloads/razor-go-main/utils/asset.go:410:74: undeclared name: bindings
/Users/gokberkgulgun/Downloads/razor-go-main/utils/asset.go:438:61: undeclared name: bindings
/Users/gokberkgulgun/Downloads/razor-go-main/utils/asset.go:448:89: undeclared name: bindings
/Users/gokberkgulgun/Downloads/razor-go-main/utils/asset.go:448:139: undeclared name: bindings
/Users/gokberkgulgun/Downloads/razor-go-main/utils/block.go:15:73: undeclared name: bindings
/Users/gokberkgulgun/Downloads/razor-go-main/utils/block.go:41:103: undeclared name: bindings
/Users/gokberkgulgun/Downloads/razor-go-main/utils/block.go:71:71: undeclared name: bindings
/Users/gokberkgulgun/Downloads/razor-go-main/utils/contract-manager.go:14:64: undeclared name: bindings
/Users/gokberkgulgun/Downloads/razor-go-main/utils/contract-manager.go:23:64: undeclared name: bindings
/Users/gokberkgulgun/Downloads/razor-go-main/utils/contract-manager.go:32:69: undeclared name: bindings
/Users/gokberkgulgun/Downloads/razor-go-main/utils/contract-manager.go:41:63: undeclared name: bindings
/Users/gokberkgulgun/Downloads/razor-go-main/utils/contract-manager.go:50:64: undeclared name: bindings
/Users/gokberkgulgun/Downloads/razor-go-main/utils/contract-manager.go:59:92: undeclared name: bindings
/Users/gokberkgulgun/Downloads/razor-go-main/utils/interface.go:84:45: undeclared name: bindings
/Users/gokberkgulgun/Downloads/razor-go-main/utils/interface.go:89:52: undeclared name: bindings
/Users/gokberkgulgun/Downloads/razor-go-main/utils/interface.go:93:84: undeclared name: bindings
/Users/gokberkgulgun/Downloads/razor-go-main/utils/interface.go:96:54: undeclared name: bindings
/Users/gokberkgulgun/Downloads/razor-go-main/utils/interface.go:97:45: undeclared name: bindings
/Users/gokberkgulgun/Downloads/razor-go-main/utils/interface.go:98:54: undeclared name: bindings
/Users/gokberkgulgun/Downloads/razor-go-main/utils/interface.go:99:56: undeclared name: bindings
/Users/gokberkgulgun/Downloads/razor-go-main/utils/interface.go:107:53: undeclared name: bindings
/Users/gokberkgulgun/Downloads/razor-go-main/utils/interface.go:115:44: undeclared name: bindings
/Users/gokberkgulgun/Downloads/razor-go-main/utils/interface.go:116:50: undeclared name: bindings
/Users/gokberkgulgun/Downloads/razor-go-main/utils/interface.go:117:59: undeclared name: bindings
/Users/gokberkgulgun/Downloads/razor-go-main/utils/interface.go:119:56: undeclared name: bindings
/Users/gokberkgulgun/Downloads/razor-go-main/utils/interface.go:120:64: undeclared name: bindings
/Users/gokberkgulgun/Downloads/razor-go-main/utils/interface.go:121:70: undeclared name: bindings
/Users/gokberkgulgun/Downloads/razor-go-main/utils/interface.go:122:71: undeclared name: bindings
/Users/gokberkgulgun/Downloads/razor-go-main/utils/interface.go:123:33: undeclared name: bindings
/Users/gokberkgulgun/Downloads/razor-go-main/utils/interface.go:124:29: undeclared name: bindings
/Users/gokberkgulgun/Downloads/razor-go-main/utils/interface.go:131:39: undeclared name: bindings
/Users/gokberkgulgun/Downloads/razor-go-main/utils/interface.go:132:49: undeclared name: bindings
/Users/gokberkgulgun/Downloads/razor-go-main/utils/interface.go:136:70: undeclared name: bindings
/Users/gokberkgulgun/Downloads/razor-go-main/utils/interface.go:136:120: undeclared name: bindings
```

THANK YOU FOR CHOOSING

// HALBORN