RouterProtocol

GoBridge Audit

Prepared by: **Halborn**

Date of Engagement: **September 15th, 2021 - November 25th, 2021**

Visit: **Halborn.com**

DOCUMENT REVISION HISTORY

VERSION	MODIFICATION	DATE	AUTHOR
0.1	Document Creation	09/18/2021	Ataberk Yavuzer
0.2	Document Updates	10/18/2021	Gokberk Gulgun
0.3	Document Updates	10/19/2021	Ataberk Yavuzer
0.4	Draft Review	10/31/2021	Gabi Urrutia
1.0	Remediation Plan	11/25/2021	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

EXECUTIVE OVERVIEW

1.1 INTRODUCTION

Router Protocol engaged Halborn to conduct a security assessment on a **Router Bridge** beginning on September 15th, 2021 and ending November 25th, 2021. The security assessment was scoped to the Bridge provided in the Github repository Router Protocol Bridge Repository Halborn conducted this audit to measure security risk and identify any new vulnerabilities introduced during the final stages of development before the production release.

In summary, some issues were found by auditors that were mostly addressed by the Router Protocol team.

1.2 AUDIT SUMMARY

The team at Halborn was provided six weeks for the engagement and assigned two full time security engineers to audit the security of the Bridge. Security engineers are 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 Bridge functions operate as intended.
- Identify potential security issues with the Bridge.

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

In summary, Halborn identified some security risks that should be addressed by the RouterProtocol 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 Bridge audit. While manual testing is recommended to uncover flaws in logic, process,and implementation; automated testing techniques help enhance coverage of Bridge 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.
- Bridge manual code review and walkthrough.
- Manual Assessment of use and safety for the critical variables and functions in scope to identify any arithmetic related vulnerability classes.
- Statical Analysis scanning of Bridge files for vulnerabilities, security hotspots or bugs. (gosec, ineffassign, unconvert and staticcheck)
- Testnet deployment (ganache-cli)

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. It’s quantitative model ensures repeatable and accurate measurement while enabling users to see the underlying vulnerability characteristics that was 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 **Router Bridge** of the following repository:

https://github.com/router-protocol/router-bridge

Branch: feature/upgradeable_contracts

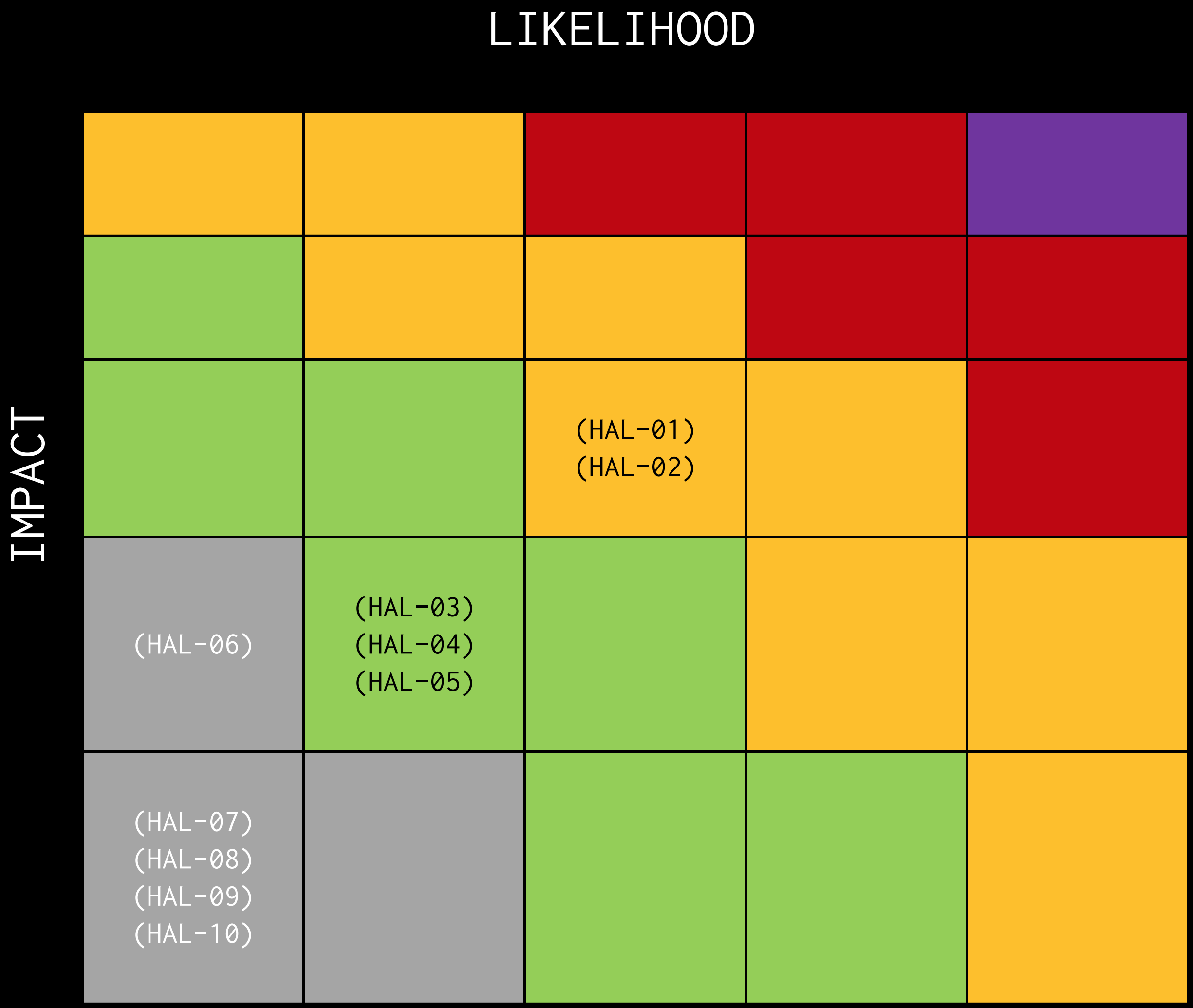
Commit ID: e39d4b5b250706278d3606459e4d24cb8512813c

OUT-OF-SCOPE:

External libraries, test scripts, bindings and e2e directories on the repository.

2. ASSESSMENT SUMMARY & FINDINGS OVERVIEW

CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL
0	0	2	3	5



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

SECURITY ANALYSIS	RISK LEVEL	REMEDIATION DATE
(HAL01) - SUPPORTED CHAIN IDS ARE NOT FULLY IMPLEMENTED	Medium	SOLVED - 11/25/2021
(HAL02) - SOME PROPOSAL METHODS ONLY CONTROLLED ON GOERLI NETWORK	Medium	SOLVED - 11/25/2021
(HAL03) - DELETED FUNCTIONALITIES STILL LOCATED ON THE CODE BASE	Medium	SOLVED - 11/25/2021
(HAL04) - HARDCODED KEYSTORE VARIABLE ON DOCKER CONFIG	Low	SOLVED - 11/25/2021
(HAL05) - ESTIMATED GAS VARIABLE RESTRICTED TO ZERO	Low	NOT APPLICABLE
(HAL06) - UNHANDLED ERRORS	Low	SOLVED - 11/25/2021
(HAL07) - REDUNDANT CODE	Informational	SOLVED - 11/25/2021
(HAL08) - OPEN TODOS	Informational	NOT APPLICABLE
(HAL09) - LOG DEBUGGING IS ENABLED ON PRODUCTION ENVIRONMENT	Informational	FUTURE RELEASE
(HAL10) - CONFUSION ON THE HANDLER CONTRACT NAMES	Informational	ACKNOWLEDGED

FINDINGS & TECH DETAILS

3.1 (HAL-01) SUPPORTED CHAIN IDS ARE NOT FULLY IMPLEMENTED - MEDIUM

Description:

Ethereum compatible networks have two identifiers, a network ID and a chain ID. Peer-to-peer communication between nodes uses the network ID, while the transaction signature process uses the chain ID. It depends on the environment variable, the chainToNetwork is configured only for Polygon and Ethereum. However, on the production environment avalanche network supported too.

Code Location:

Listing 1: /chains/ethereum/writer-methods.go

```go
        if destStableTokenAddress != destTokenAddress {
            chainIndexStart := int(0)

            chainToNetworkId := make(map[string]string)
            if os.Getenv("IS_PRODUCTION") == "true" {
                chainToNetworkId["0"] = "1"
                chainToNetworkId["1"] = "137"
                chainIndexStart = 1
            } else {
                chainToNetworkId["0"] = "3"
                chainToNetworkId["1"] = "42"
                chainToNetworkId["2"] = "80001"
                chainToNetworkId["3"] = "97"
                chainToNetworkId["4"] = "256"
                chainToNetworkId["5"] = "43113"
                chainIndexStart = 0
            }
```

Risk Level:

Likelihood - 3
Impact - 3

Recommendations:

Ensure that all mainnet chain ids are added into if statement on the production environment.

Remediation Plan:

SOLVED: The Router Protocol team solved the issue.

Commit ID: **da2467ed70bc5b57d3a9c1a6e649504c502b89b6**

3.2 (HAL-02) SOME PROPOSAL METHODS ONLY CONTROLLED ON GOERLI NETWORK - MEDIUM

Description:

On the writer methods, Chain ID is checked and interaction is completed through erc20 handler contract. However, If we take a look at the defined id, the id belongs to Goerli test network.

Code Location:

Code Location

Listing 2: /chains/ethereum/writer-methods.go

```go
func (w *writer) proposalIsComplete(srcId msg.ChainId, nonce msg.Nonce, dataHash [32]byte) bool {
    if w.cfg.id == 5 || strconv.Itoa(int(w.cfg.id)) == "5" {

        address := w.cfg.erc20HandlerContract
        instance, err := NewBridge(address, w.conn.Client())
        prop, err := instance.GetProposal(w.conn.CallOpts(), uint8(srcId), uint64(nonce), dataHash)

        if err != nil {
            w.log.Error("Failed to check proposal existence", "err", err)
            return false
        }

        return prop.Status == PassedStatus || prop.Status == TransferredStatus || prop.Status == CancelledStatus

    } else {
        prop, err := w.bridgeContract.GetProposal(w.conn.CallOpts(), uint8(srcId), uint64(nonce), dataHash)

        if err != nil {
            w.log.Error("Failed to check proposal existence", "err", err)
            return false
        }
        return prop.Status == PassedStatus || prop.Status == TransferredStatus || prop.Status == CancelledStatus

    }

}
```

Risk Level:

Likelihood - 3
Impact - 3

Recommendations:

Consider reviewing the handler interaction. It is recommended to standardize all handler contract interactions.

Remediation Plan:

SOLVED: The Goerli network is for testing proposal. The code was commented.

Commit ID: **da2467ed70bc5b57d3a9c1a6e649504c502b89b6**

3.3 (HAL-03) DELETED FUNCTIONALITIES STILL LOCATED ON THE CODE BASE - LOW

Description:

On the code base, genericHandler and erc721Handler are deleted from the repository. However, the handlers are still located on the different functions.

Code Location:

createGenericDepositProposal
createErc721Proposal

Listing 3: /chains/ethereum/writer-methods.go

```go
func (w *writer) createGenericDepositProposal(m msg.Message) bool {
    w.log.Info("Creating generic proposal", "src", m.Source, "nonce", m.DepositNonce)
    ...
}

func (w *writer) createErc721Proposal(m msg.Message) bool {
    w.log.Info("Creating erc721 proposal", "src", m.Source, "nonce", m.DepositNonce)
    ...
}
```

Risk Level:

Likelihood - 2
Impact - 2

Recommendations:

Consider reviewing the unused handlers from the code base. It is recommended to delete redundant code.

Remediation Plan:

SOLVED: The Router Protocol team commented and removed those functions in commit ID: **da2467ed70bc5b57d3a9c1a6e649504c502b89b6**.

3.4 (HAL-04) HARDCODED KEYSTORE VARIABLE ON DOCKER CONFIG - LOW

Description:

Compose is a tool for defining and running multi-container Docker applications. With Compose, you use a YAML file to configure your application's services. Then, with a single command, you create and start all the services from your configuration.

There are some environment variables on the docker-compose.yml file, which is the configuration file that enables the composing process to be performed. During the tests, it was seen that the keystore password stored on docker-compose.yml file directly.

Code Location:

Listing 4: docker-compose.yml

```yaml
# SPDX-License-Identifier: LGPL-3.0-only

version: '3'
services:
  bridge:
    build:
      context: .
    container_name: bridge
    environment:
      - KEYSTORE_PASSWORD=password
    command: --config /config/config.json
    volumes:
      - ./config:/config
      - ./keys:/keys/
    network_mode: host
```

Risk Level:

Likelihood - 2

Impact - 2

Recommendations:

It is recommended to use third-party software such as AWS Secret Manager or HashiCorp Vault to obtain important information such as passwords.

Remediation Plan:

SOLVED: The issue was solved by the Router Protocol team.

Commit ID: **da2467ed70bc5b57d3a9c1a6e649504c502b89b6**

3.5 (HAL-05) ESTIMATED GAS VARIABLE RESTRICTED TO ZERO - LOW

Description:

During the code review, It has been observed that estimatedGasArray and estimatedGas are defined as zero value. The variables are not changed in the time flow of executeProposal workflow.

Code Location:

Listing 5: /chains/ethereum/writer-methods.go

```go
flagsLength := len(jsonResponse.Key.TokenAddresses) - 1
flags = make([]*big.Int, 0)
parts := make([]*big.Int, 0)
estimatedGasArr := make([]*big.Int, 0)

part := big.NewInt(10)
estimatedGas := big.NewInt(0)

if int(m.Destination) == 1-chainIndexStart {
    // flags[0] = int(17716740096)
    // flags of size (length(path) - 1)
    flag := big.NewInt(70386460917760)

    for i := 0; i < flagsLength; i++ {
        flags = append(flags, flag)
        parts = append(parts, part)
        estimatedGasArr = append(estimatedGasArr, estimatedGas)
    }
} else if int(m.Destination) == 2-chainIndexStart {
    flag := big.NewInt(17716742144)
    for i := 0; i < flagsLength; i++ {
        flags = append(flags, flag)
        parts = append(parts, part)
        estimatedGasArr = append(estimatedGasArr, estimatedGas)
```

```
                        }
                    }
```

Risk Level:

Likelihood - 2
Impact - 2

Recommendations:

EstimatedGasArr and EstimatedGas should adjust dynamically according to type of the chain.

Remediation Plan:

NOT APPLICABLE: That is functionality to deduct erc20 tokens as fees. But they do not deduct any fees and thus pass 0.

3.6 (HAL-06) UNHANDLED ERRORS - INFORMATIONAL

Description:

Error handling on the applications is important for the proper operation of the applications. Otherwise, as a result of not managing these errors properly, it will not be possible to predict what the problem is. During the tests, it was determined that although some error variables on the code were defined, they were not handled in any way.

Code Location:

No error handlers are defined for the following methods:

Listing 6: /chains/ethereum/writer-methods.go (Lines 541)

```go
541 json.Unmarshal(responseData, &jsonResponse)
```

Listing 7: /cmd/router-bridge/main.go (Lines 125)

```go
125 viper.ReadInConfig()
```

Risk Level:

Likelihood - 1
Impact - 2

Recommendations:

It is recommended to handle errors for code applicability and proper operation.

Remediation Plan:

SOLVED: The Router Protocol team solved the issue.

Commit ID: **da2467ed70bc5b57d3a9c1a6e649504c502b89b6**

3.7 (HAL-07) REDUNDANT CODE - INFORMATIONAL

Description:

During the test, it was determined that one of the variables on the code was not used, although it was defined on the code. This situation does not pose any risk in terms of security. But it is important for the readability and applicability of the code.

Code Location:

According to the code below, the topic_string variable used for no purpose since the array32 variable is also unused on that function.

Listing 8: /chains/ethereum/writer-methods.go (Lines 235,236,237)

```go
234 func buildQueryForProposalNew(contract ethcommon.Address, sig
        utils.EventSig, startBlock *big.Int, endBlock *big.Int) eth.
        FilterQuery {
235     topic_string := ethcommon.Hex2BytesFixed("0
            x968626a768e76ba1363efe44e322a6c4900c5f084e0b45f35e294dfddaa9e0d5
            ", 32)
236     var array32 [32]byte
237     copy(array32[:], topic_string)
238     query := eth.FilterQuery{
239         FromBlock: startBlock,
240         ToBlock:   endBlock,
241         Addresses: []ethcommon.Address{contract},
242         Topics:    nil,
243     }
244     return query
245 }
```

Risk Level:

Likelihood - 1
Impact - 1

Recommendations:

It is suggested to remove unused variable from the code to improve readability.

Remediation Plan:

SOLVED: The Router Protocol team solved the issue.

Commit ID: **da2467ed70bc5b57d3a9c1a6e649504c502b89b6**

3.8 (HAL-08) OPEN TODOS - INFORMATIONAL

Description:

During the audit, it was seen that the same parts of the code still has many open TODOs. It means, the planned works on these codes are still not done.

Code Location:

Listing 9

```
https://github.com/router-protocol/router-bridge/blob/e39d4b5b250706278d3606459e4d24cb8512813c/chains/substrate/connection.go#L182
https://github.com/router-protocol/router-bridge/blob/e39d4b5b250706278d3606459e4d24cb8512813c/chains/substrate/connection.go#L228
https://github.com/router-protocol/router-bridge/blob/e39d4b5b250706278d3606459e4d24cb8512813c/chains/substrate/events.go#L41
https://github.com/router-protocol/router-bridge/blob/e39d4b5b250706278d3606459e4d24cb8512813c/shared/substrate/client.go#L18
https://github.com/router-protocol/router-bridge/blob/e39d4b5b250706278d3606459e4d24cb8512813c/shared/substrate/query.go#L21
```

Risk Level:

Likelihood - 1
Impact - 1

Recommendations:

Consider updating the code base such that these TODOs have been completed before deploy.

Remediation Plan:

NOT APPLICABLE: The Router Protocol team claims that TODOs are in Substrate files and they do not use them.

3.9 (HAL-09) LOG DEBUGGING IS ENABLED ON PRODUCTION ENVIRONMENT - INFORMATIONAL

Description:

Logging is often neglected by developers when thinking of security considerations. However, proper logging practice can provide the crucial forensics needed to investigate after a breach, and perhaps more importantly, a change to detect security issues as they happen. However, the debugging logs should be disabled on the production. Logs should be classified according to error severity.

Code Location:

Listing 10

```
https://github.com/router-protocol/router-bridge/blob/
    e39d4b5b250706278d3606459e4d24cb8512813c/chains/ethereum/chain.
    go#L204
https://github.com/router-protocol/router-bridge/blob/
    e39d4b5b250706278d3606459e4d24cb8512813c/chains/ethereum/
    listener.go#L84
https://github.com/router-protocol/router-bridge/blob/
    e39d4b5b250706278d3606459e4d24cb8512813c/chains/ethereum/
    listener.go#L130
https://github.com/router-protocol/router-bridge/blob/
    e39d4b5b250706278d3606459e4d24cb8512813c/chains/ethereum/
    listener.go#L166
https://github.com/router-protocol/router-bridge/blob/
    e39d4b5b250706278d3606459e4d24cb8512813c/chains/ethereum/
    listener.go#L192
https://github.com/router-protocol/router-bridge/blob/
    e39d4b5b250706278d3606459e4d24cb8512813c/chains/ethereum/
    listener.go#L199
https://github.com/router-protocol/router-bridge/blob/
    e39d4b5b250706278d3606459e4d24cb8512813c/chains/ethereum/
    writer_methods.go#L388
```

```
https://github.com/router-protocol/router-bridge/blob/
    e39d4b5b250706278d3606459e4d24cb8512813c/chains/ethereum/
    writer_methods.go#L427
https://github.com/router-protocol/router-bridge/blob/
    e39d4b5b250706278d3606459e4d24cb8512813c/chains/ethereum/writer
    .go#L44
https://github.com/router-protocol/router-bridge/blob/
    e39d4b5b250706278d3606459e4d24cb8512813c/chains/substrate/
    connection.go#L71
https://github.com/router-protocol/router-bridge/blob/
    e39d4b5b250706278d3606459e4d24cb8512813c/chains/substrate/
    connection.go#L79
https://github.com/router-protocol/router-bridge/blob/
    e39d4b5b250706278d3606459e4d24cb8512813c/chains/substrate/
    connection.go#L86
https://github.com/router-protocol/router-bridge/blob/
    e39d4b5b250706278d3606459e4d24cb8512813c/config/config.go#L103
https://github.com/router-protocol/router-bridge/blob/
    e39d4b5b250706278d3606459e4d24cb8512813c/config/config.go#L118
```

Risk Level:

Likelihood - 1
Impact - 1

Recommendations:

Consider to use golang logging mechanisms like a logger.info()- logger.error() instead of debug.

Remediation Plan:

PENDING: The Router Protocol team will remove debug logs before going live on the mainnet.

3.10 (HAL-10) CONFUSION ON THE HANDLER CONTRACT NAMES - INFORMATIONAL

Description:

During the manual code review, It has been seen that erc20handler are re-named as NewERC20HandlerUpgradeable. Although the instances are re-named correctly, the contract address variables are still named as erc20HandlerContract.

Code Location:

Listing 11: chains/ethereum/writer-methods.go (Lines 405)

```
404             handlerAddress := w.cfg.erc20HandlerContract
405             instance, err := ERC20Handler.
                    NewERC20HandlerUpgradeable(handlerAddress, w.conn.
                    Client())
```

Risk Level:

Likelihood - 1
Impact - 1

Recommendations:

Ensure that variables are configured correctly. The Erc20HandlerUpgradeable should be interacted with upgradeable contract.

Remediation Plan:

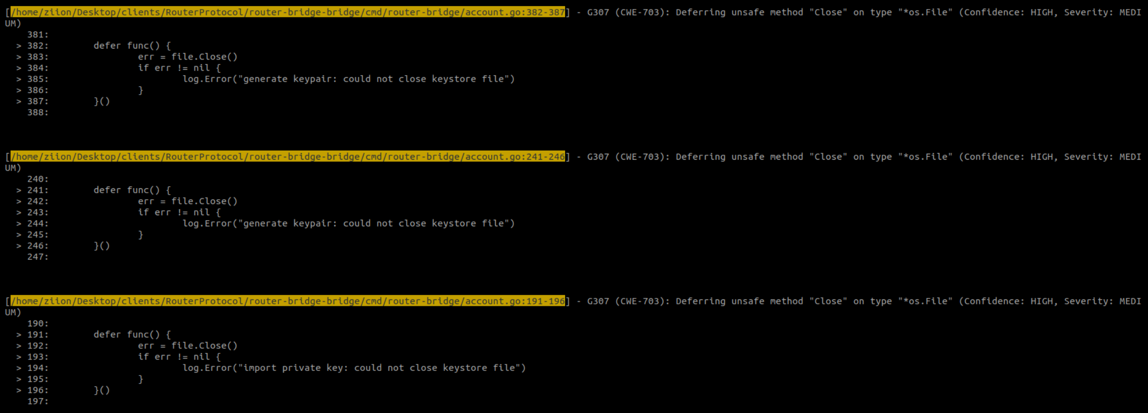
ACKNOWLEDGED: It makes it easier to work with the older code. The contract bindings are updated but the working of the bridge remains mostly the same, so we use the same variable names.

AUTOMATED TESTING

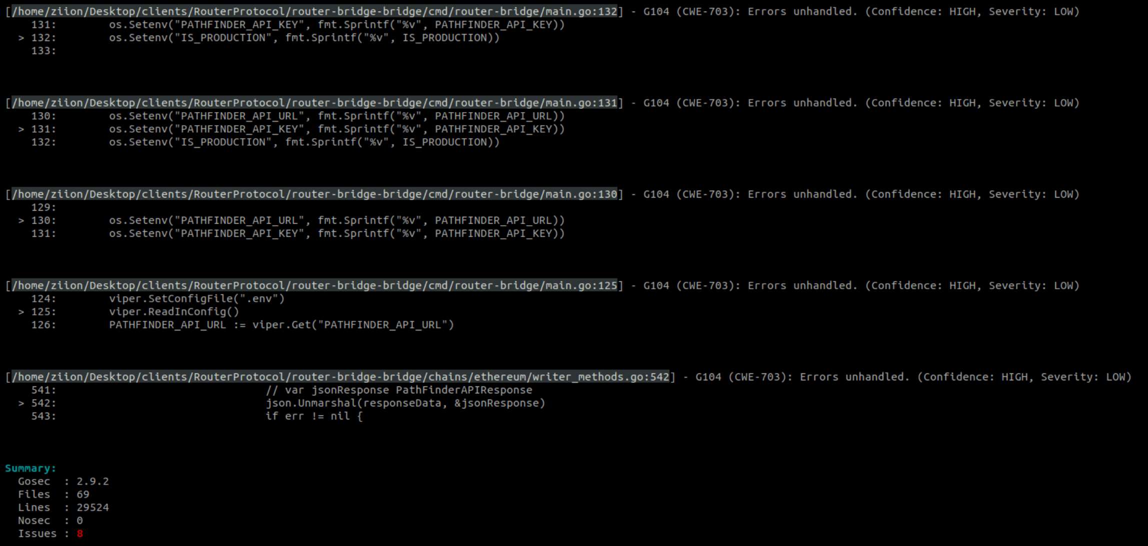
Description:

Halborn used automated testing techniques to enhance coverage of certain areas of the scoped component. Among the tools used were staticcheck, gosec ineffassign, unconvert. 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.

Gosec - Security Analysis Output Sample:

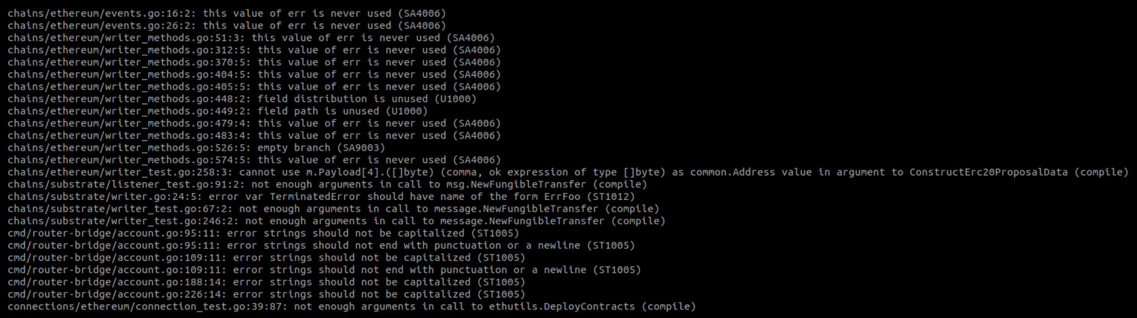


```
[/home/ziion/Desktop/clients/RouterProtocol/router-bridge-bridge/cmd/router-bridge/account.go:382-387] - G307 (CWE-703): Deferring unsafe method "Close" on type "*os.File" (Confidence: HIGH, Severity: MEDI
UM)
    381:
  > 382:        defer func() {
  > 383:                err = file.Close()
  > 384:                if err != nil {
  > 385:                        log.Error("generate keypair: could not close keystore file")
  > 386:                }
  > 387:        }()
    388:


[/home/ziion/Desktop/clients/RouterProtocol/router-bridge-bridge/cmd/router-bridge/account.go:241-246] - G307 (CWE-703): Deferring unsafe method "Close" on type "*os.File" (Confidence: HIGH, Severity: MEDI
UM)
    240:
  > 241:        defer func() {
  > 242:                err = file.Close()
  > 243:                if err != nil {
  > 244:                        log.Error("generate keypair: could not close keystore file")
  > 245:                }
  > 246:        }()
    247:


[/home/ziion/Desktop/clients/RouterProtocol/router-bridge-bridge/cmd/router-bridge/account.go:191-196] - G307 (CWE-703): Deferring unsafe method "Close" on type "*os.File" (Confidence: HIGH, Severity: MEDI
UM)
    190:
  > 191:        defer func() {
  > 192:                err = file.Close()
  > 193:                if err != nil {
  > 194:                        log.Error("import private key: could not close keystore file")
  > 195:                }
  > 196:        }()
    197:
```



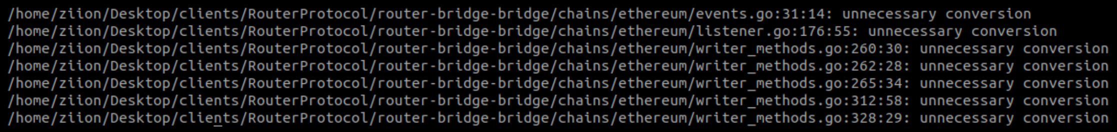
```
[/home/ziion/Desktop/clients/RouterProtocol/router-bridge-bridge/cmd/router-bridge/main.go:132] - G104 (CWE-703): Errors unhandled. (Confidence: HIGH, Severity: LOW)
    131:        os.Setenv("PATHFINDER_API_KEY", fmt.Sprintf("%v", PATHFINDER_API_KEY))
  > 132:        os.Setenv("IS_PRODUCTION", fmt.Sprintf("%v", IS_PRODUCTION))
    133:


[/home/ziion/Desktop/clients/RouterProtocol/router-bridge-bridge/cmd/router-bridge/main.go:131] - G104 (CWE-703): Errors unhandled. (Confidence: HIGH, Severity: LOW)
    130:        os.Setenv("PATHFINDER_API_URL", fmt.Sprintf("%v", PATHFINDER_API_URL))
  > 131:        os.Setenv("PATHFINDER_API_KEY", fmt.Sprintf("%v", PATHFINDER_API_KEY))
    132:        os.Setenv("IS_PRODUCTION", fmt.Sprintf("%v", IS_PRODUCTION))


[/home/ziion/Desktop/clients/RouterProtocol/router-bridge-bridge/cmd/router-bridge/main.go:130] - G104 (CWE-703): Errors unhandled. (Confidence: HIGH, Severity: LOW)
    129:
  > 130:        os.Setenv("PATHFINDER_API_URL", fmt.Sprintf("%v", PATHFINDER_API_URL))
    131:        os.Setenv("PATHFINDER_API_KEY", fmt.Sprintf("%v", PATHFINDER_API_KEY))


[/home/ziion/Desktop/clients/RouterProtocol/router-bridge-bridge/cmd/router-bridge/main.go:125] - G104 (CWE-703): Errors unhandled. (Confidence: HIGH, Severity: LOW)
    124:        viper.SetConfigFile(".env")
  > 125:        viper.ReadInConfig()
    126:        PATHFINDER_API_URL := viper.Get("PATHFINDER_API_URL")


[/home/ziion/Desktop/clients/RouterProtocol/router-bridge-bridge/chains/ethereum/writer_methods.go:542] - G104 (CWE-703): Errors unhandled. (Confidence: HIGH, Severity: LOW)
    541:                                // var jsonResponse PathFinderAPIResponse
  > 542:                                json.Unmarshal(responseData, &jsonResponse)
    543:                                if err != nil {


Summary:
  Gosec  : 2.9.2
  Files  : 69
  Lines  : 29524
  Nosec  : 0
  Issues : 8
```

Staticcheck - Security Analysis Output Sample:



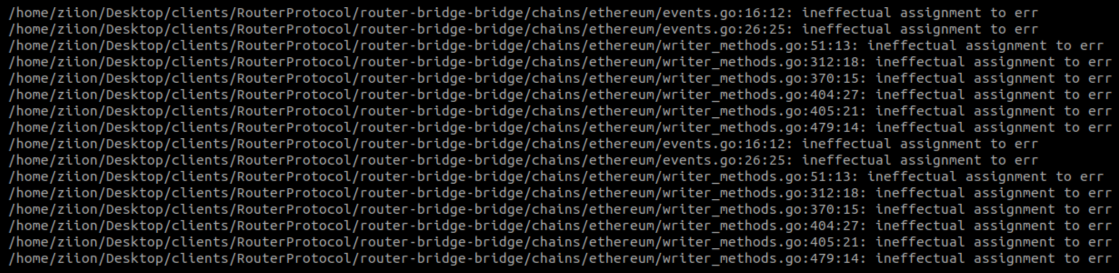
```
chains/ethereum/events.go:16:2: this value of err is never used (SA4006)
chains/ethereum/events.go:26:2: this value of err is never used (SA4006)
chains/ethereum/writer_methods.go:51:3: this value of err is never used (SA4006)
chains/ethereum/writer_methods.go:312:5: this value of err is never used (SA4006)
chains/ethereum/writer_methods.go:370:5: this value of err is never used (SA4006)
chains/ethereum/writer_methods.go:404:5: this value of err is never used (SA4006)
chains/ethereum/writer_methods.go:405:5: this value of err is never used (SA4006)
chains/ethereum/writer_methods.go:448:2: field distribution is unused (U1000)
chains/ethereum/writer_methods.go:449:2: field path is unused (U1000)
chains/ethereum/writer_methods.go:479:4: this value of err is never used (SA4006)
chains/ethereum/writer_methods.go:483:4: this value of err is never used (SA4006)
chains/ethereum/writer_methods.go:526:5: empty branch (SA9003)
chains/ethereum/writer_methods.go:574:5: this value of err is never used (SA4006)
chains/ethereum/writer_test.go:258:3: cannot use m.Payload[4].([]byte) (comma, ok expression of type []byte) as common.Address value in argument to ConstructErc20ProposalData (compile)
chains/substrate/listener_test.go:91:2: not enough arguments in call to msg.NewFungibleTransfer (compile)
chains/substrate/writer.go:24:5: error var TerminatedError should have name of the form ErrFoo (ST1012)
chains/substrate/writer_test.go:67:2: not enough arguments in call to message.NewFungibleTransfer (compile)
chains/substrate/writer_test.go:246:2: not enough arguments in call to message.NewFungibleTransfer (compile)
cmd/router-bridge/account.go:95:11: error strings should not be capitalized (ST1005)
cmd/router-bridge/account.go:95:11: error strings should not end with punctuation or a newline (ST1005)
cmd/router-bridge/account.go:109:11: error strings should not be capitalized (ST1005)
cmd/router-bridge/account.go:109:11: error strings should not end with punctuation or a newline (ST1005)
cmd/router-bridge/account.go:188:14: error strings should not be capitalized (ST1005)
cmd/router-bridge/account.go:226:14: error strings should not be capitalized (ST1005)
connections/ethereum/connection_test.go:39:87: not enough arguments in call to ethutils.DeployContracts (compile)
```

Unconvert - Security Analysis Output Sample:



```
/home/ziion/Desktop/clients/RouterProtocol/router-bridge-bridge/chains/ethereum/events.go:31:14: unnecessary conversion
/home/ziion/Desktop/clients/RouterProtocol/router-bridge-bridge/chains/ethereum/listener.go:176:55: unnecessary conversion
/home/ziion/Desktop/clients/RouterProtocol/router-bridge-bridge/chains/ethereum/writer_methods.go:260:30: unnecessary conversion
/home/ziion/Desktop/clients/RouterProtocol/router-bridge-bridge/chains/ethereum/writer_methods.go:262:28: unnecessary conversion
/home/ziion/Desktop/clients/RouterProtocol/router-bridge-bridge/chains/ethereum/writer_methods.go:265:34: unnecessary conversion
/home/ziion/Desktop/clients/RouterProtocol/router-bridge-bridge/chains/ethereum/writer_methods.go:312:58: unnecessary conversion
/home/ziion/Desktop/clients/RouterProtocol/router-bridge-bridge/chains/ethereum/writer_methods.go:328:29: unnecessary conversion
```

Ineffassign - Security Analysis Output Sample:



```
/home/ziion/Desktop/clients/RouterProtocol/router-bridge-bridge/chains/ethereum/events.go:16:12: ineffectual assignment to err
/home/ziion/Desktop/clients/RouterProtocol/router-bridge-bridge/chains/ethereum/events.go:26:25: ineffectual assignment to err
/home/ziion/Desktop/clients/RouterProtocol/router-bridge-bridge/chains/ethereum/writer_methods.go:51:13: ineffectual assignment to err
/home/ziion/Desktop/clients/RouterProtocol/router-bridge-bridge/chains/ethereum/writer_methods.go:312:18: ineffectual assignment to err
/home/ziion/Desktop/clients/RouterProtocol/router-bridge-bridge/chains/ethereum/writer_methods.go:370:15: ineffectual assignment to err
/home/ziion/Desktop/clients/RouterProtocol/router-bridge-bridge/chains/ethereum/writer_methods.go:404:27: ineffectual assignment to err
/home/ziion/Desktop/clients/RouterProtocol/router-bridge-bridge/chains/ethereum/writer_methods.go:405:21: ineffectual assignment to err
/home/ziion/Desktop/clients/RouterProtocol/router-bridge-bridge/chains/ethereum/writer_methods.go:479:14: ineffectual assignment to err
/home/ziion/Desktop/clients/RouterProtocol/router-bridge-bridge/chains/ethereum/events.go:16:12: ineffectual assignment to err
/home/ziion/Desktop/clients/RouterProtocol/router-bridge-bridge/chains/ethereum/events.go:26:25: ineffectual assignment to err
/home/ziion/Desktop/clients/RouterProtocol/router-bridge-bridge/chains/ethereum/writer_methods.go:51:13: ineffectual assignment to err
/home/ziion/Desktop/clients/RouterProtocol/router-bridge-bridge/chains/ethereum/writer_methods.go:312:18: ineffectual assignment to err
/home/ziion/Desktop/clients/RouterProtocol/router-bridge-bridge/chains/ethereum/writer_methods.go:370:15: ineffectual assignment to err
/home/ziion/Desktop/clients/RouterProtocol/router-bridge-bridge/chains/ethereum/writer_methods.go:404:27: ineffectual assignment to err
/home/ziion/Desktop/clients/RouterProtocol/router-bridge-bridge/chains/ethereum/writer_methods.go:405:21: ineffectual assignment to err
/home/ziion/Desktop/clients/RouterProtocol/router-bridge-bridge/chains/ethereum/writer_methods.go:479:14: ineffectual assignment to err
```

As a result of the tests completed with tools above, some results were obtained and these results were reviewed by Halborn. In line with the reviewed results, it was decided that some vulnerabilities were false-positive and these results were not included in the report. The actual vulnerabilities are already included in the findings on the report.

THANK YOU FOR CHOOSING

HALBORN