BoostyLabs - Tricorn Bridge - Server

Golang Security Assessment

Prepared by: **Halborn**

Date of Engagement: **January 23rd, 2023 - April 26th, 2023**

Visit: **Halborn.com**

DOCUMENT REVISION HISTORY 10
CONTACTS 11
1 EXECUTIVE OVERVIEW 12
1.1 INTRODUCTION 13
1.2 ASSESSMENT SUMMARY 13
1.3 TEST APPROACH & METHODOLOGY 14
2 RISK METHODOLOGY 15
2.1 EXPLOITABILITY 16
2.2 IMPACT 17
2.3 SEVERITY COEFFICIENT 19
2.4 SCOPE 21
3 ASSESSMENT SUMMARY & FINDINGS OVERVIEW 23
4 FINDINGS & TECH DETAILS 25
4.1 (HAL-01) PRIVATE SSH KEYS COMMITTED TO GIT REPOSITORY - CRITICAL(10) 27
Description 27
Code Location 27
Proof of concept 27
BVSS 28
Recommendation 28
Remediation Plan 28
4.2 (HAL-02) PANIC ON EVENT PARSING - HIGH(8.4) 29
BVSS 29
Description 29
Code Location 29

Proof-of-concept 30

BVSS 30

Recommendation 30

Remediation Plan 30

4.3 (HAL-03) TOKENID IS HARD-CODED IN TRANSFERS - HIGH(7.5) 31

Description 31

Code Location 31

Proof-of-concept 33

BVSS 33

Recommendation 33

Remediation Plan 33

4.4 (HAL-04) USERS CANNOT WITHDRAW FUNDS DUE TO HARD-CODED COMMISSION VALUES - HIGH(7.0) 34

Description 34

Code Location 34

Proof-of-concept 38

BVSS 38

Recommendation 38

Remediation Plan 38

4.5 (HAL-05) DEPOSITS CAN BE LOCKED DUE TO WITHDRAWALS BY CONTRACT OWNER - MEDIUM(6.2) 39

Code Location 39

Proof-of-concept 39

BVSS 40

Recommendation 40

Remediation Plan 40

4.6 (HAL-06) BRIDGE SUPPORTS UNENCRYPTED TRAFFIC - MEDIUM(6.2) 41
Description 41
Code Location 41
BVSS 42
Recommendation 42
Remediation Plan 42
4.7 (HAL-07) ADMINISTRATOR LOGIN PROTOCOLS ARE ACCESSIBLE FOR TEST SERVER - LOW(3.1) 43
Description 43
Code Location 43
Proof of Concept 45
BVSS 45
Recommendation 45
Remediation Plan 45
4.8 (HAL-08) UI ELEMENT FOR TRANSACTION FEE IS HARD-CODED - LOW(3.1) 46
Description 46
Code Location 46
Proof of concept 47
BVSS 47
Recommendation 47
Remediation Plan 47
4.9 (HAL-09) PRIVATE ETHERSCAN API KEY EXPOSED IN GIT REPOSITORY - LOW(2.5) 48
Description 48

Code Location 48

Proof of concept 49

BVSS 49

Recommendation 49

Remediation Plan 49

4.10 (HAL-10) LACK OF RATE LIMITING ON THE API - LOW(3.1) 50

Description 50

Proof of concept 50

BVSS 50

Recommendation 50

Remediation Plan 51

4.11 (HAL-11) BRIDGE TRANSACTIONS ARE SIGNED BY A SINGLE PRIVATE KEY (CENTRALIZATION RISK) - LOW(3.9) 52

Description 52

Code Location 52

Proof of Concept 52

BVSS 52

Recommendation 53

Remediation Plan 53

4.12 (HAL-12) LOSS TO PROTOCOL YIELD OR AVAILABILITY DUE TO COMMISSION RATES BETWEEN BRIDGE AND SMART CONTRACT BECOMING DESYNCED - LOW(3.6) 54

Description 54

Code Location 55

Proof of Concept 56

BVSS 56

Recommendation 56

Remediation Plan 56
4.13 (HAL-13) POSSIBLE DIVISION BY ZERO - LOW(2.1) 57
Description 57
Code Location 57
BVSS 58
Recommendation 58
Remediation Plan 58
4.14 (HAL-14) INSECURE PASSWORD POLICY - LOW(3.8) 59
BVSS 59
Description 59
Code Location 59
BVSS 60
Recommendation 60
Remediation Plan 60
4.15 (HAL-15) USE OF VULNERABLE DEPENDENCIES - LOW(4.4) 61
Description 61
Code Location 61
BVSS 62
Recommendation 63
Remediation Plan 63
4.16 (HAL-16) CONVERSION OF STRINGS TO NUMBERS LEAD TO INTEGER OVER-FLOWS - LOW(3.1) 64
BVSS 64
Description 64
Code Location 64
BVSS 67

Recommendation 67
Remediation Plan 67
4.17 (HAL-17) INFORMATION DISCLOSURE DUE TO WEB SERVER ERROR HANDLING - INFORMATIONAL(0.0) 68
BVSS 68
Description 68
Code Location 68
BVSS 69
Recommendation 70
Remediation Plan 70
4.18 (HAL-18) REVERSE.STRINGS DOES NOT FUNCTION CORRECTLY - INFORMATIONAL(0.0) 71
Description 71
Code Location 71
BVSS 72
Recommendation 72
Remediation Plan 72
4.19 (HAL-19) OUTDATED REFERENCES TO A REPOSITORY THAT HAS BEEN RENAMED - INFORMATIONAL(0.0) 73
Description 73
Code Location 73
BVSS 73
Recommendation 74
Remediation Plan 74
4.20 (HAL-20) UNIT TESTS MISSING - INFORMATIONAL(0.0) 75

Description 75

Code Location 75

BVSS 78

Recommendation 78

Remediation Plan 78

4.21 (HAL-21) LACK OF FUZZ TESTS - INFORMATIONAL(0.0) 79

Description 79

BVSS 79

Recommendation 79

Remediation Plan 79

4.22 (HAL-22) GO VERSION 1.18 IS UNSUPPORTED - INFORMATIONAL(0.0) 80

Description 80

Code Location 80

BVSS 80

Recommendation 80

Remediation Plan 80

4.23 (HAL-23) GO VERSIONS PRIOR TO 1.20 ARE MORE VULNERABLE TO SIDE-CHANNEL ATTACKS - INFORMATIONAL(0.0) 81

Description 81

Code Location 81

BVSS 81

Recommendation 81

Remediation Plan 81

4.24 (HAL-24) INCORRECT ERROR MESSAGE FOR HISTORY ENDPOINT IN GATEWAY - INFORMATIONAL(0.0) 82

Description 82

Code Location 82
BVSS 82
Recommendation 82
Remediation Plan 83
4.25 (HAL-25) TODOS IN CODEBASE - INFORMATIONAL(0.0) 84
Description 84
Code Location 84
BVSS 86
Recommendation 86
Remediation Plan 86
4.26 (HAL-26) INCONSISTENCY IN API VALUES - INFORMATIONAL(0.0) 87
Description 87
Code Location 87
BVSS 88
Recommendation 88
Remediation Plan 88
5 AUTOMATED TESTING 89
5.1 Automated Testing -- Overview 90
5.2 codeql 90
5.3 gosec 90
5.4 govulncheck 91
5.5 nancy 91
5.6 npm audit 92
5.7 Fuzz Testing 93
Event Parsing 93
String Reversal 95

DOCUMENT REVISION HISTORY

VERSION	MODIFICATION	DATE	AUTHOR
0.1	Document Creation	04/22/2023	John Saigle
0.2	Document Updates	04/25/2023	John Saigle
0.3	Document Review	04/25/2023	Gokberk Gulgun
0.4	Document Review	04/26/2023	Gabi Urrutia
1.0	Remediation Plan	06/09/2023	John Saigle
1.1	Remediation Plan Update	08/03/2023	Gokberk Gulgun
1.2	Remediation Plan Review	08/06/2023	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
John Saigle	Halborn	John.Saigle@halborn.com

EXECUTIVE OVERVIEW

1.1 INTRODUCTION

BoostyLabs engaged Halborn to conduct a security assessment on their Tricorn Bridge beginning on January 23rd, 2023 and ending on April 26th, 2023. The security assessment was scoped to the Go code that contains bridging functionality for the project. Due to some internal re-organization, the assessment was paused, and the scope was revised to include additional Go code in March 2023. The assessment resumed and concluded in April 2023. In total, six weeks of work was performed by Halborn on the Go assessment.

1.2 ASSESSMENT SUMMARY

The team at Halborn was provided two weeks for the engagement and assigned a full-time security engineer to verify the security of the Tricorn Bridge. The security engineer is a blockchain and smart-contract security expert with advanced penetration testing, smart-contract hacking, and deep knowledge of multiple blockchain protocols.

The purpose of this assessment is to:

- Ensure that the Go bridge functionality operates as intended.
- Identify potential security issues with the bridge.
- Review security best practices for Go web applications.
- Identify the interfaces between Web2 and Web3 components and consider risks in both paradigms.

In summary, Halborn identified some security risks that mostly addressed by the BoostyLabs 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 custom modules. 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 assessment:

- Research into architecture and purpose.
- Static Analysis of security for scoped repository, and imported functions. (e.g., staticcheck, gosec, unconvert, codeql, ineffassign and semgrep)
- Manual Assessment for discovering security vulnerabilities on codebase.
- Ensuring correctness of the codebase.
- Dynamic Analysis of files and modules related to Tricorn.
- Custom fuzz testing, Go’s built-in fuzzing tools.

It is important to note that manual, interactive testing was limited during the assessment due to certain issues present within the codebase at different points as the scope was being revised. As a result, manual code review by auditors, as well as static analysis, were the primary methods through which the assessment was conducted.

2. RISK METHODOLOGY

Every vulnerability and issue observed by Halborn is ranked based on **two sets** of **Metrics** and a **Severity Coefficient**. This system is inspired by the industry standard Common Vulnerability Scoring System.

The two **Metric sets** are: **Exploitability** and **Impact**. **Exploitability** captures the ease and technical means by which vulnerabilities can be exploited and **Impact** describes the consequences of a successful exploit.

The **Severity Coefficients** is designed to further refine the accuracy of the ranking with two factors: **Reversibility** and **Scope**. These capture the impact of the vulnerability on the environment as well as the number of users and smart contracts affected.

The final score is a value between 0-10 rounded up to 1 decimal place and 10 corresponding to the highest security risk. This provides an objective and accurate rating of the severity of security vulnerabilities in smart contracts.

The system is designed to assist in identifying and prioritizing vulnerabilities based on their level of risk to address the most critical issues in a timely manner.

2.1 EXPLOITABILITY

Attack Origin (AO):

Captures whether the attack requires compromising a specific account.

Attack Cost (AC):

Captures the cost of exploiting the vulnerability incurred by the attacker relative to sending a single transaction on the relevant blockchain. Includes but is not limited to financial and computational cost.

Attack Complexity (AX):

Describes the conditions beyond the attacker's control that must exist in order to exploit the vulnerability. Includes but is not limited to macro situation, available third-party liquidity and regulatory challenges.

Metrics:

Exploitability Metric (m_E)	Metric Value	Numerical Value
Attack Origin (AO)	Arbitrary (AO:A)	1
	Specific (AO:S)	0.2
Attack Cost (AC)	Low (AC:L)	1
	Medium (AC:M)	0.67
	High (AC:H)	0.33
Attack Complexity (AX)	Low (AX:L)	1
	Medium (AX:M)	0.67
	High (AX:H)	0.33

Exploitability E is calculated using the following formula:

$$E = \prod m_e$$

2.2 IMPACT

Confidentiality (C):

Measures the impact to the confidentiality of the information resources managed by the contract due to a successfully exploited vulnerability. Confidentiality refers to limiting access to authorized users only.

Integrity (I):

Measures the impact to integrity of a successfully exploited vulnerability. Integrity refers to the trustworthiness and veracity of data stored and/or processed on-chain. Integrity impact directly affecting Deposit or Yield records is excluded.

Availability (A):

Measures the impact to the availability of the impacted component resulting from a successfully exploited vulnerability. This metric refers to smart contract features and functionality, not state. Availability impact directly affecting Deposit or Yield is excluded.

Deposit (D):

Measures the impact to the deposits made to the contract by either users or owners.

Yield (Y):

Measures the impact to the yield generated by the contract for either users or owners.

Metrics:

Impact Metric (m_I)	Metric Value	Numerical Value
Confidentiality (C)	None (I:N)	0
	Low (I:L)	0.25
	Medium (I:M)	0.5
	High (I:H)	0.75
	Critical (I:C)	1
Integrity (I)	None (I:N)	0
	Low (I:L)	0.25
	Medium (I:M)	0.5
	High (I:H)	0.75
	Critical (I:C)	1
Availability (A)	None (A:N)	0
	Low (A:L)	0.25
	Medium (A:M)	0.5
	High (A:H)	0.75
	Critical	1
Deposit (D)	None (D:N)	0
	Low (D:L)	0.25
	Medium (D:M)	0.5
	High (D:H)	0.75
	Critical (D:C)	1
Yield (Y)	None (Y:N)	0
	Low (Y:L)	0.25
	Medium: (Y:M)	0.5
	High: (Y:H)	0.75
	Critical (Y:H)	1

Impact I is calculated using the following formula:

$$I = max(m_I) + \frac{\sum m_I - max(m_I)}{4}$$

2.3 SEVERITY COEFFICIENT

Reversibility (R):

Describes the share of the exploited vulnerability effects that can be reversed. For upgradeable contracts, assume the contract private key is available.

Scope (S):

Captures whether a vulnerability in one vulnerable contract impacts resources in other contracts.

Coefficient (C)	Coefficient Value	Numerical Value
Reversibility (r)	None (R:N)	1
	Partial (R:P)	0.5
	Full (R:F)	0.25
Scope (s)	Changed (S:C)	1.25
	Unchanged (S:U)	1

Severity Coefficient C is obtained by the following product:

$$C = rs$$

The Vulnerability Severity Score S is obtained by:

$$S = min(10, EIC * 10)$$

The score is rounded up to 1 decimal places.

Severity	Score Value Range
Critical	9 - 10
High	7 - 8.9
Medium	4.5 - 6.9
Low	2 - 4.4
Informational	0 - 1.9

2.4 SCOPE

During the assessment, the scope was modified several times:

- **Phase 1**: Originally, the project used a public repository, casper-eth-bridge. This repository is no longer accessible to Halborn.
- **Phase 2**: A modification of the project was pushed to a new repository, golden-gate. Out of the six total weeks of assessment effort, four were spent on this version of the codebase using the commit hash 47dbfab7f191041fa58957fbae0d8e752c463dc7. The repository was renamed to tricorn.
- **Phase 3**: Additional bridging features were pushed to the codebase. The final two weeks of the assessment were spent on this phase.

The repository for the assessment is Tricorn Bridge's repository tricorn as well as the RPC definitions found in tricorn-communication. The repositories are located at the following URLs:

- Tricorn: https://github.com/BoostyLabs/tricorn
- Tricorn Communication: https://github.com/BoostyLabs/tricorn-communication

The commit hashes are as follows:

- Tricorn: 3a3f25bc8643998ad40a1dec0352dd5cc661806c
- Tricorn Communication: 3efdf854cd94635599697d82abf803c1bbfd06f3

Not all the code in the Tricorn repository was in-scope. The engagement excludes code relating to the following features:

- The currencyrates directory, and any functionality relating to the calculations of currencies and bridge oracle features
- Bridging functionality related to the Solana network

- Bridging functionality related to the Avalanche network
- Code review of key dependencies such as casper-golang-sdk

Halborn has delivered two previous assessment reports for the Tricorn bridge project that cover the Solidity and Casper smart contracts. This report covers the off-chain web application that relays messages between chains. For a more complete understanding of the security status of the project, it is best to read the previous reports along with this one.

2. REMEDIATION PULL REQUESTS :

- 310
- 336
- 375
- 340
- 308
- 311
- 305
- 304
- 296
- 300
- 301
- 302
- 270
- 338
- 337
- 339
- 341
- 326

3. ASSESSMENT SUMMARY & FINDINGS OVERVIEW

CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL
1	3	2	10	10

SECURITY ANALYSIS	RISK LEVEL	REMEDIATION DATE
(HAL-01) PRIVATE SSH KEYS COMMITTED TO GIT REPOSITORY	Critical (10)	SOLVED - 02/01/2023
(HAL-02) PANIC ON EVENT PARSING	High (8.4)	SOLVED - 07/20/2023
(HAL-03) TOKENID IS HARD-CODED IN TRANSFERS	High (7.5)	SOLVED - 07/25/2023
(HAL-04) USERS CANNOT WITHDRAW FUNDS DUE TO HARD-CODED COMMISSION VALUES	High (7.0)	SOLVED - 05/05/2023
(HAL-05) DEPOSITS CAN BE LOCKED DUE TO WITHDRAWALS BY CONTRACT OWNER	Medium (6.2)	FUTURE RELEASE
(HAL-06) BRIDGE SUPPORTS UNENCRYPTED TRAFFIC	Medium (6.2)	SOLVED - 05/08/2023
(HAL-07) ADMINISTRATOR LOGIN PROTOCOLS ARE ACCESSIBLE FOR TEST SERVER	Low (3.1)	SOLVED - 02/01/2023
(HAL-08) UI ELEMENT FOR TRANSACTION FEE IS HARD-CODED	Low (3.1)	SOLVED - 07/25/2023
(HAL-09) PRIVATE ETHERSCAN API KEY EXPOSED IN GIT REPOSITORY	Low (2.5)	SOLVED - 01/01/2023
(HAL-10) LACK OF RATE LIMITING ON THE API	Low (3.1)	SOLVED - 07/19/2023
(HAL-11) BRIDGE TRANSACTIONS ARE SIGNED BY A SINGLE PRIVATE KEY (CENTRALIZATION RISK)	Low (3.9)	FUTURE RELEASE
(HAL-12) LOSS TO PROTOCOL YIELD OR AVAILABILITY DUE TO COMMISSION RATES BETWEEN BRIDGE AND SMART CONTRACT BECOMING DESYNCED	Low (3.6)	SOLVED - 07/25/2023
(HAL-13) POSSIBLE DIVISION BY ZERO	Low (2.1)	SOLVED - 05/06/2023
(HAL-14) INSECURE PASSWORD POLICY	Low (3.8)	SOLVED - 07/27/2023
(HAL-15) USE OF VULNERABLE DEPENDENCIES	Low (4.4)	SOLVED - 05/06/2023

(HAL-16) CONVERSION OF STRINGS TO NUMBERS LEAD TO INTEGER OVERFLOWS	Low (3.1)	SOLVED - 07/26/2023
(HAL-17) INFORMATION DISCLOSURE DUE TO WEB SERVER ERROR HANDLING	Informational (0.0)	SOLVED - 05/07/2023
(HAL-18) REVERSE.STRINGS DOES NOT FUNCTION CORRECTLY	Informational (0.0)	SOLVED - 05/07/2023
(HAL-19) OUTDATED REFERENCES TO A REPOSITORY THAT HAS BEEN RENAMED	Informational (0.0)	SOLVED - 07/26/2023
(HAL-20) UNIT TESTS MISSING	Informational (0.0)	SOLVED - 05/08/2023
(HAL-21) LACK OF FUZZ TESTS	Informational (0.0)	SOLVED - 07/25/2023
(HAL-22) GO VERSION 1.18 IS UNSUPPORTED	Informational (0.0)	SOLVED - 05/06/2023
(HAL-23) GO VERSIONS PRIOR TO 1.20 ARE MORE VULNERABLE TO SIDE-CHANNEL ATTACKS	Informational (0.0)	SOLVED - 05/06/2023
(HAL-24) INCORRECT ERROR MESSAGE FOR HISTORY ENDPOINT IN GATEWAY	Informational (0.0)	SOLVED - 03/20/2023
(HAL-25) TODOS IN CODEBASE	Informational (0.0)	ACKNOWLEDGED
(HAL-26) INCONSISTENCY IN API VALUES	Informational (0.0)	SOLVED - 05/08/2023

FINDINGS & TECH DETAILS

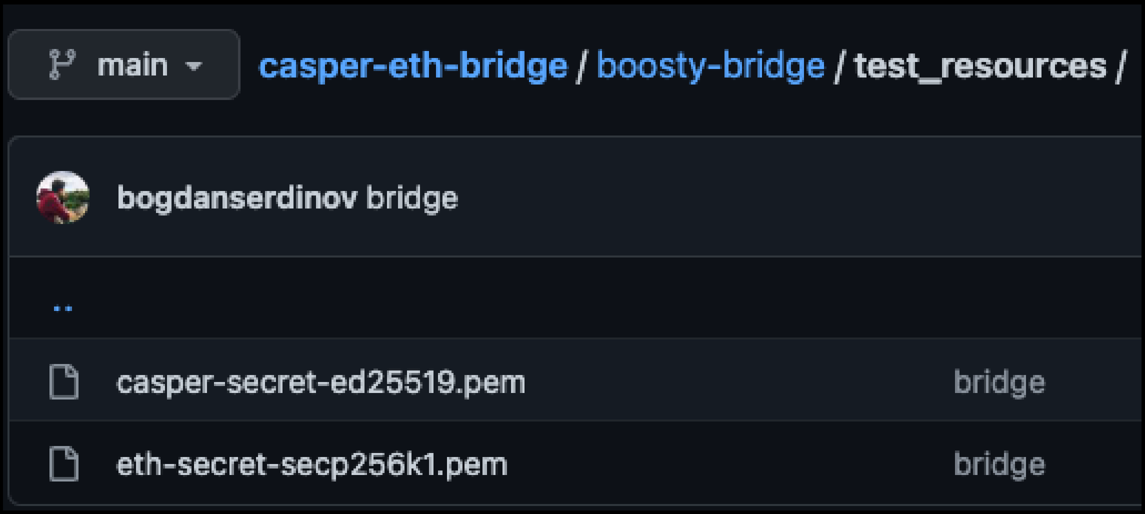
4.1 (HAL-01) PRIVATE SSH KEYS COMMITTED TO GIT REPOSITORY - CRITICAL(10)

Description:

Private SSH keys are publicly exposed on GitHub.

Code Location:

Private SSH keys are accessible by browsing the public casper-eth-bridge repository.



main casper-eth-bridge / boosty-bridge / test_resources /

bogdanserdinov bridge

..

casper-secret-ed25519.pem bridge

eth-secret-secp256k1.pem bridge

Figure 1: Private keys are accessible via GitHub

Proof of concept:

- A malicious party reads the private keys.
- The private keys are used to authenticate to the test server.
- With access to the test server, the attacker can access other sensitive information such as database passwords, API keys, and private source code.

BVSS:

AO:A/AC:L/AX:M/C:C/I:C/A:C/D:C/Y:C/R:N/S:C (10)

Recommendation:

Remove the SSH keys from GitHub. Disable their access to all servers, including test servers. Ensure that private information is not committed to git repositories.

Remediation Plan:

SOLVED: The casper-eth-bridge repository is no longer public.

4.2 (HAL-02) PANIC ON EVENT PARSING - HIGH (8.4)

BVSS:

AO:A/AC:L/AX:M/C:N/I:N/A:C/D:N/Y:N/R:N/S:C (8.4)

Description:

The code for parsing events into Go data structures can panic under certain circumstances. If the incoming event data is not in the expected format, a nil dereference error can occur, which causes a panic in Go code. If encountered in a production environment, services using this code will crash and require manual restart from an administrator.

Code Location:

The code below performs string slicing without doing bound checks. This is the source of the panic error. When an offset exceeds the length of the slice, the code will panic.

tricorn/internal/eventparsing/eventparsing.go

Listing 1: getNextParam() accesses an offset index without checking that it is within the range of e.Bytes

```go
// getNextParam returns next parameter for specified data length.
func (e *EventData) getNextParam(offset int, limit int) string {
    e.offset += offset
    param := e.Bytes[e.offset : e.offset+limit]
    e.offset += limit
    return param
}

// GetEventType returns event type from event data.
func (e *EventData) GetEventType() (int, error) {
    eventTypeHex := e.getNextParam(LengthSelectorString.Int(),
↳ LengthSelectorTag.Int())
```

```

    eventTypeBytes, err := hex.DecodeString(eventTypeHex)
    if err != nil {
        return 0, err
    }

    eventType := big.NewInt(0).SetBytes(eventTypeBytes)

    return int(eventType.Int64()), nil
}
```

Proof-of-concept:

- An event is processed by the chain that is not in the expected format.
- When attempting to parse the event, the code pictured above panics.
- Bridge services calling this code will be unavailable until an administrator restarts the service

This vulnerability was discovered using fuzz testing. Further details can be found in the Automated Testing section at the end of this report.

BVSS:

AO:A/AC:L/AX:M/C:N/I:N/A:C/D:N/Y:N/R:N/S:C (8.4)

Recommendation:

Add bounds checks to the parsing code to ensure that panics do not occur.

Remediation Plan:

SOLVED: BoostyLabs solved the issue by adding the checks.

Pull Request : 310

4.3 (HAL-03) TOKENID IS HARD-CODED IN TRANSFERS - HIGH (7.5)

Description:

The value for TokenId in transfers is hard-coded. This means that it is only possible to transfer a single ERC-20 ‘test’ token. As a result, the bridge is not fully functional, as there is no way to specify a different token based on user input.

Code Location:

bridge/service.go

Listing 2: Example 1: Building a Bridge In transfer: tokenId is hard-coded to the value 1 (Line 400)

```
399     tokenTransfer := transfers.TokenTransfer{
400         TokenID:            1, // todo: dynamically change.
401         Amount:             *amount,
402         Status:             transfers.StatusWaiting,
403         SenderNetworkID:    int64(networks.NetworkNameToID[
  ↳ senderNetworkName]),
404         SenderAddress:      senderAddress,
405         RecipientNetworkID: int64(recipientNetworkID),
406         RecipientAddress:   recipientAddress,
407     }
408
409     err = service.tokenTransfers.Create(ctx, tokenTransfer)
410     if err != nil {
411         service.log.Error(fmt.Sprintf("couldn't create token
  ↳ transfer for network name %s", request.Sender.NetworkName), Error.
  ↳ Wrap(err))
412         return BridgeInSignatureResponse{}, Error.Wrap(err)
413     }
414
415     return bridgeInSignature, service.nonces.Increment(ctx,
  ↳ senderID)
```

bridge/service.go

Listing 3: Example 2: Responding to a blockchain event log to do a 'Bridge Out' action (Lines 580,609)

```go
573     { // call BridgeOut.
574         toAddress, err := networks.StringToBytes(
 ↳ recipientNetworkID, eventFund.EventFundsIn.To.Address)
575         if err != nil {
576             service.log.Error("", Error.Wrap(err))
577             return status.Error(codes.Internal, Error.Wrap(err).
 ↳ Error())
578         }
579
580         token, err := service.networkTokens.Get(ctx,
 ↳ recipientNetworkID, 1) // TODO: add dynamic token id.
581         if err != nil {
582             service.log.Error("", Error.Wrap(err))
583             return status.Error(codes.Internal, Error.Wrap(err).
 ↳ Error())
584         }
585
586         bridgeOut, err := service.connectors[networks.Name(
 ↳ eventFund.EventFundsIn.To.NetworkName)].BridgeOut(ctx, chains.
 ↳ TokenOutRequest{
587             Amount: amount,
588             Token:  token.ContractAddress,
589             To:     toAddress,
590             From: networks.Address{
591                 NetworkName: networkName.String(),
592                 Address:     hex.EncodeToString(senderAddress),
593             },
594             TransactionID: big.NewInt(int64(transactionID)),
595         })
596         if err != nil {
597             service.log.Error("", Error.Wrap(err))
598             return status.Error(codes.Internal, Error.Wrap(err).
 ↳ Error())
599         }
600         if len(bridgeOut.Txhash) == 0 {
601             err = errs.New("couldn't send bridgeOut in network %s"
 ↳ , eventFund.EventFundsIn.To.NetworkName)
602             service.log.Error("", Error.Wrap(err))
603             return status.Error(codes.Internal, Error.Wrap(err).
 ↳ Error())
```

```
        }
    }

    tokenTransfer := transfers.TokenTransfer{
        TokenID:          1, // todo: dynamically change.
        Amount:           *amount,
        SenderAddress:    senderAddress,
        RecipientAddress: recipientAddress,
    }
```

Proof-of-concept:

- A user wishes to make a transfer using the bridge.
- When visiting the UI, the user can choose only a ‘Test’ token.
- As a result, the user cannot transfer their ERC-20 tokens.

BVSS:

AO:A/AC:L/AX:L/C:N/I:N/A:H/D:N/Y:N/R:N/S:U (7.5)

Recommendation:

Remove placeholder values and replace with dynamic values. Before pushing changes to the codebase, ensure that core operations are tested thoroughly with a range of values.

Remediation Plan:

SOLVED: BoostyLabs solved the issue by fixing the implementation.

Pull Request : 336

4.4 (HAL-04) USERS CANNOT WITHDRAW FUNDS DUE TO HARD-CODED COMMISSION VALUES - HIGH (7.0)

Description:

Users cannot withdraw their liquidity from the bridge due to a hard-coded value for 'commission'.

The expected behavior is that a user can deposit tokens to the bridge, and later request a 'Cancel Signature' in order to withdraw the tokens they have deposited. This in turn triggers a 'Transfer Out' action that corresponds to a Solidity function that allows a user to withdraw funds.

This two-step process involves a deposit and a withdraw action, each referred to as "Bridge In" and "Cancel Signature" within the Go code. For both steps, a commission that represents a gas cost of the transaction is calculated and sent as a value in a "Bridge In" transaction to Solidity. The Solidity code stores this amount separately from the contact's main token balance in a map that matches the sender's address to the commission amount.

Code Location:

The commission field for TransferOut is calculated in the EstimateTransfer function in turns uses a go-ethereum method, SuggestGasPrice.
chains/evm/service.go

Listing 4: (Lines 182,192,195)

```
181 // EstimateTransfer estimates transfer fee and time.
182 func (service *Service) EstimateTransfer(ctx context.Context) (
  ↳ chains.Estimation, error) {
183     gasPrice, err := service.ethClient.SuggestGasPrice(ctx)
184     if err != nil {
185         return chains.Estimation{}, Error.Wrap(err)
186     }
```

```go

    // TODO: change in dynamic way, as in BridgeOut method.
    gasLimit := new(big.Int).SetUint64(service.config.GasLimit)
    feeWei  := gasPrice.Mul(gasPrice, gasLimit)

    fee  := new(big.Float).Quo(new(big.Float).SetInt(feeWei), new(big.Float).SetUint64(wei))

    estimation := chains.Estimation{
        Fee:                   fee.String(),
        FeePercentage:         service.config.FeePercentage,
        EstimatedConfirmation: service.config.ConfirmationTime,
    }

    return estimation, nil
}
```

Ordinarily, the commission is calculated using the above function. However, it is commented-out in the codebase:

tricorn/bridge/service.go

Listing 5: Building a BridgeIn transaction: gasCommission is hard-coded to the value 1 and calculations are commented-out. (Lines 363,364,379)

```go
363     // TODO: uncomment after fix.
364     // destinationEstimation, err := service.connectors[recipientNetworkName].EstimateTransfer(ctx, transfers.EstimateTransfer{
365     //  SenderNetwork:    request.Sender.NetworkName,
366     //  RecipientNetwork: request.Destination.NetworkName,
367     //  TokenID:          request.TokenID,
368     //  Amount:           request.Amount,
369     // })
370     // if err != nil {
371     //  return BridgeInSignatureResponse{}, Error.Wrap(err)
372     // }
373     //
374     // gasCommission, ok := new(big.Int).SetString(destinationEstimation.Fee, 10)
375     // if !ok {
376     //  return BridgeInSignatureResponse{}, Error.New("couldn't parse gas commission")
```

```go
    // }.

    gasCommission := new(big.Int).SetInt64(1)
```

The same commission value is not commented-out for the CancelSignature function (which is the corresponding ‘withdraw’ action after the ‘deposit’ action performed by BridgeIn).
tricorn/bridge/service.go

Listing 6: Building a CancelSignature transaction: the commission is calculated via EstimateTransfer

```go
    estimation, err := service.connectors[networkName].
 ↳ EstimateTransfer(ctx, transfers.EstimateTransfer{
        SenderNetwork:    networkName.String(),
        RecipientNetwork: networkName.String(),
        TokenID:          uint32(tokenTransfer.TokenID),
        Amount:           tokenTransfer.Amount.String(),
    })
    if err != nil {
        return transfers.CancelSignatureResponse{}, Error.Wrap(err
 ↳ )
    }

    commission, ok := new(big.Int).SetString(estimation.Fee, 10)
```

The commission calculated above is eventually encoded in a “Transfer Out” transaction that corresponds to the Solidity function transferOut.

The BridgeIn commission will be equal to 1 because it is hard-coded. As a result, the TransferOut commission will always be much higher than the BridgeIn Commission by several orders of magnitude.

When the _commissionPools value reaches 0, the operation will fail. This will happen as soon as a user tries to transferOut given the difference in size between the commission values.

Ultimately, this will cause the transferOut operation to revert to Solidity when the commission is subtracted from the pool ethereum/contracts/Bridge.sol from the tricorn-smart-contracts git repository.

Listing 7: commission is subtracted from _commissionPools

```
function transferOut (
    address token,
    address recipient,
    uint256 amount,
    uint256 commission,
    uint256 nonce,
    bytes calldata signature
) external whenNotPaused {
    if (_usedNonces[nonce]) {
        revert(Errors.ALREADY_USED_SIGNATURE);
    }


    _checkTransferOutRequest(
        address(this), token,
        recipient,
        amount,
        commission,
        nonce,
        signature
    );


    _usedNonces[nonce] = true;
    _commissionPools[token] -= commission;


    uint256 totalSumForTransfer = amount + commission;
    IERC20(token).safeTransfer(recipient,totalSumForTransfer);


    // TODO:Descrease pool commission
    emit TransferOut(recipient, nonce, token,
↳ totalSumForTransfer);
}
```

Proof-of-concept:

- A user deposits tokens into the bridge via “Bridge In”. The commission value is 1.
- Later, the user wishes to withdraw their tokens via “Cancel Signature” which creates a “Transfer Out” action with a much higher commission.
- When the Solidity code executes, a revert will occur when the _commissionPool value underflows as the high commission value is subtracted from 1.
- As a result, the user’s funds are locked in the contract.

BVSS:

AO:A/AC:L/AX:L/C:N/I:N/A:M/D:C/Y:N/R:P/S:C (7.0)

Recommendation:

Remove the TODO message and fix the code used to calculate the commission for the BridgeIn transaction type.

Remediation Plan:

SOLVED: The commented code has been removed and new logic for calculating commissions is added in PR 300.

4.5 (HAL-05) DEPOSITS CAN BE LOCKED DUE TO WITHDRAWALS BY CONTRACT OWNER - MEDIUM (6.2)

As explored in HAL-04, users can withdraw funds from the protocol, but the withdrawal will fail if the balance of the _commissionPools falls to a value that is less than the commission supplied in the withdrawal transaction.

It is possible for the contract owner to withdraw tokens from the contracts in several ways, including the “Withdraw Commission” and “Bridge Out” functions. These functions will transfer funds from the commission pool or from the ERC20 token balance of the contract. If the owner withdraws too much from either source, users will not be able to withdraw their funds. This could occur because the contract balance falls below the transaction amount that the user is attempting to reclaim, or because the commission pool’s balance falls below the commission value in the transaction.

While this operation can be exploited easily by a malicious administrator, this issue does not rely on malicious action. This scenario may occur during normal operations if the owner withdraws too much at a time relative to the gas price of the networks involved.

Code Location:

Please refer to the previous Halborn reports for the details of the Solidity and Casper smart contract functionalities. Both sets of contracts are implementing the same design, and the issue here pertains to the design itself rather than any specific mistake in the code.

Proof-of-concept:

- After regular usage of the bridge, the owner withdraws funds from the commission pools.

- A spike in gas prices occurred.
- Users will be unable to withdraw their deposits (similar to the scenario outlined in HAL-04).

BVSS:

AO:A/AC:L/AX:L/C:N/I:N/A:C/D:N/Y:N/R:P/S:C (6.2)

Recommendation:

Isolate the commission for the protocol owner into a separate data structure that does not interfere with user deposits. The normal withdrawal of funds for the contract owner should not interfere with user deposits and withdrawals.

Remediation Plan:

FUTURE RELEASE: BoostyLabs will fix the issue in the future release.

4.6 (HAL-06) BRIDGE SUPPORTS UNENCRYPTED TRAFFIC - MEDIUM (6.2)

Description:

The bridge microservices allow traffic over unencrypted network connections. gRPC's servers are initialized using an optional set of server options. If the Creds option is not specified, the connections will not be encrypted.

Code Location:

An example where the code creates a gRPC server is outlined in the following code excerpt:

internal/server/grpc/server.go

Listing 8: gRPC server created with NewServer without a Creds option

```go

// NewServer is a constructor for GRPC server.
func NewServer(logger logger.Logger, registerServer func(*grpc.
↳ Server), name, address string) *grpcserver {
    grpcServer := grpc.NewServer(
        grpc.MaxRecvMsgSize(defaultGrpcMessageSize),
        grpc.MaxSendMsgSize(defaultGrpcMessageSize),
    )

    registerServer(grpcServer)

    return &grpcserver{
        log:     logger,
        address: address,
        name:    name,
        server:  grpcServer,
    }
}
```

BVSS:

AO:A/AC:L/AX:L/C:M/I:N/A:N/D:N/Y:N/R:N/S:C (6.2)

Recommendation:

Enable encryption for network connections in the bridge project.

Remediation Plan:

SOLVED: BoostyLabs added code to encrypt traffic in PR 296.

4.7 (HAL-07) ADMINISTRATOR LOGIN PROTOCOLS ARE ACCESSIBLE FOR TEST SERVER - LOW (3.1)

Description:

The documentation for the public casper-eth-bridge repository lists an IP address.

Halborn examined this web server and discovered several configuration issues, including:

- Database authentication exposed to the Internet.
- TLS connection not enabled for sensitive data.
- Sensitive ports exposed to the Internet.

Code Location:

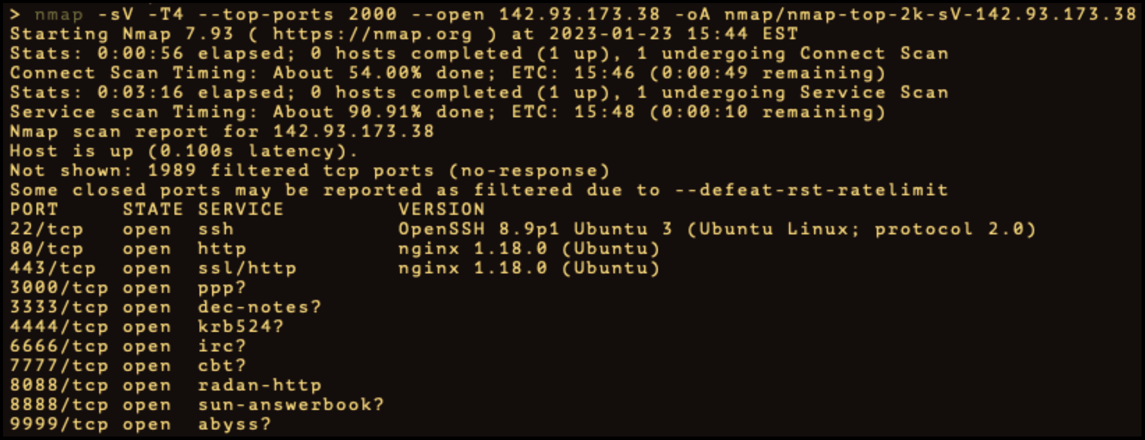
README.md

Listing 9: Excerpt from README file including the IP address

```
[Logs](http://142.93.173.38:9999/)
```

This port hosted a login form for a Dozzle logging server that was available without encryption (as seen in the above excerpt, the logs are served using the plaintext http:// protocol). The [Dozzle documentation]((https://github.com/amir20/dozzle#authentication) recommends using TLS, as otherwise passwords will be revealed.

A nmap scan was performed against the server in order to look for other exposed ports:

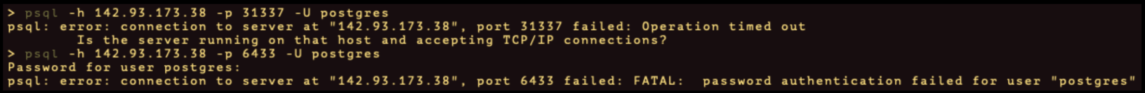


```
> nmap -sV -T4 --top-ports 2000 --open 142.93.173.38 -oA nmap/nmap-top-2k-sV-142.93.173.38
Starting Nmap 7.93 ( https://nmap.org ) at 2023-01-23 15:44 EST
Stats: 0:00:56 elapsed; 0 hosts completed (1 up), 1 undergoing Connect Scan
Connect Scan Timing: About 54.00% done; ETC: 15:46 (0:00:49 remaining)
Stats: 0:03:16 elapsed; 0 hosts completed (1 up), 1 undergoing Service Scan
Service scan Timing: About 90.91% done; ETC: 15:48 (0:00:10 remaining)
Nmap scan report for 142.93.173.38
Host is up (0.100s latency).
Not shown: 1989 filtered tcp ports (no-response)
Some closed ports may be reported as filtered due to --defeat-rst-ratelimit
PORT      STATE SERVICE          VERSION
22/tcp    open  ssh              OpenSSH 8.9p1 Ubuntu 3 (Ubuntu Linux; protocol 2.0)
80/tcp    open  http             nginx 1.18.0 (Ubuntu)
443/tcp   open  ssl/http         nginx 1.18.0 (Ubuntu)
3000/tcp open  ppp?
3333/tcp open  dec-notes?
4444/tcp open  krb524?
6666/tcp open  irc?
7777/tcp open  cbt?
8088/tcp open  radan-http
8888/tcp open  sun-answerbook?
9999/tcp open  abyss?
```

Figure 2: Exposed ports

An exposed port 22 indicates that an SSH login is possible via the Internet. If an attacker brute-forces the password, they can obtain access to a user account on the server.

Postgres is accessible to the internet on port 6433. A login was attempted as a proof-of-concept.



```
> psql -h 142.93.173.38 -p 31337 -U postgres
psql: error: connection to server at "142.93.173.38", port 31337 failed: Operation timed out
        Is the server running on that host and accepting TCP/IP connections?
> psql -h 142.93.173.38 -p 6433 -U postgres
Password for user postgres:
psql: error: connection to server at "142.93.173.38", port 6433 failed: FATAL:  password authentication failed for user "postgres"
```

Figure 3: Login is possible to postgres via the Internet. The first line shows a login attempt on a random, closed port. The second indicates that a Postgres service is listening

The remaining exposed ports overlap with many of the ones that are opened in Tricorn's deploy/docker-compose.yml file, namely:

Listing 10: Ports opened in the Docker file

```
3000
6432
6433
6666
7777
8088
8888
9999
```

Taken together, this indicates that the bridge microservices were likely running on this server. This means that it would have been possible for an attacker to brute-force the user account and database passwords for the test server.

Proof of Concept:

- An attacker runs a tool to brute-force the SSH or Postgres password for the test server.
- If successful, they could access the source code of the application and/or the contents of the database.
- If password reuse occurs, the attacker may be able to compromise other infrastructure operated by Boosty.

BVSS:

AO:A/AC:L/AX:H/C:M/I:M/A:M/D:N/Y:N/R:N/S:C (3.1)

Recommendation:

Do not expose authentication to the public Internet. Instead, ensure that only privileged connections are allowed via a VPN. Ensure that public key authentication is used for SSH rather than passwords. Use TLS to secure authentication rather than plaintext HTTP connections. Avoid reusing credentials in testing and production environments. If there has been any credential reuse, then all credentials should be changed immediately to be strong, unique passphrases. As the server has been exposed to the Internet, consider investigating the server forensically in case a breach has already occurred.

Remediation Plan:

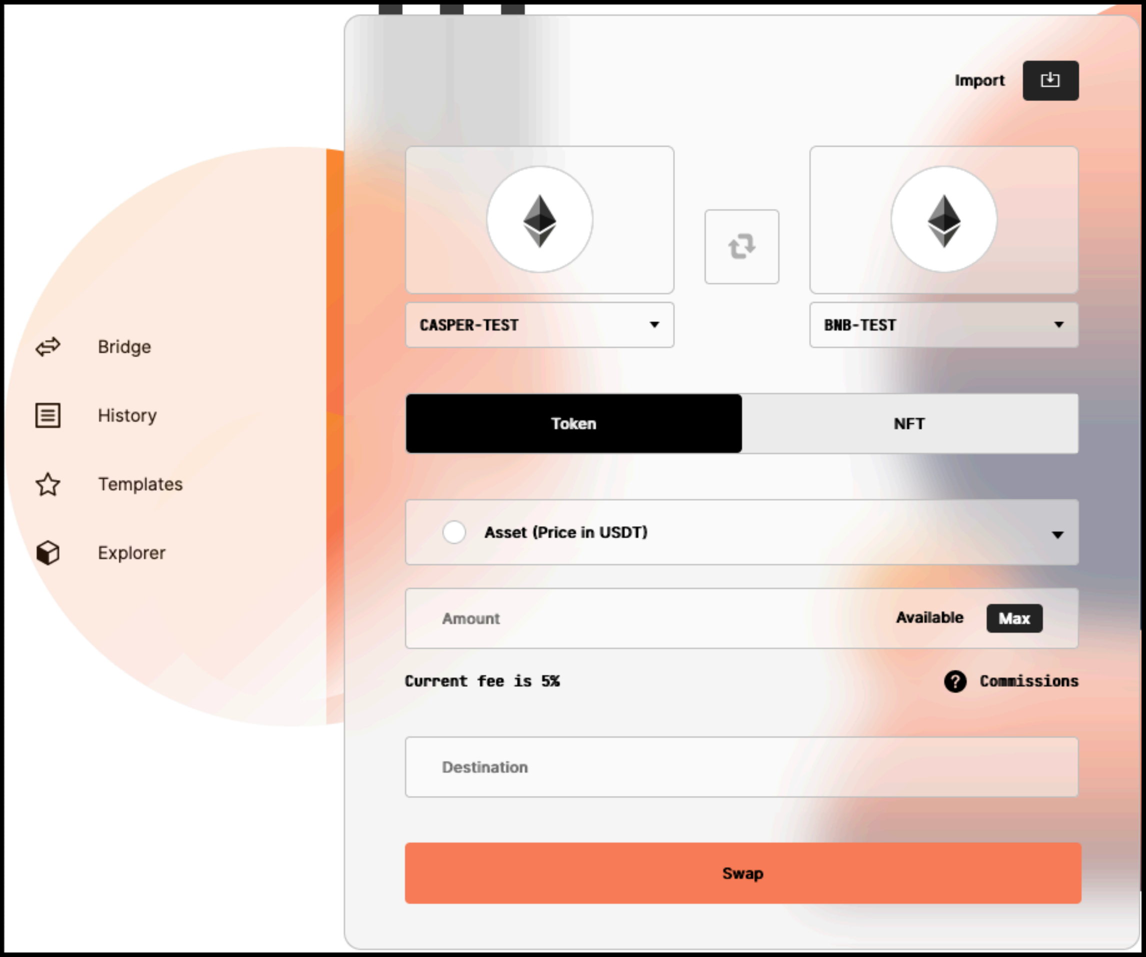
SOLVED: BoostyLabs was notified of this finding immediately during the initial stages of the assessment. The public repository listing the IP address is no longer accessible. The IP address is no longer accessible.

4.8 (HAL-08) UI ELEMENT FOR TRANSACTION FEE IS HARD-CODED - LOW (3.1)

Description:

The web application UI is hard-coded to display a fee percentage of 5%. Users will not be informed of the actual percentage as configured in the bridge or the contract.

Code Location:



web/bridge/src/apps/views/Swaps/index.tsx
{language=javascript caption="The fee is displayed using fixed

text. There is no retrieval of a dynamic value.} <span className=" swap__comission__label">Current fee is 5%</span>

During testing, the web application displayed a fee of 5% even though both the Go bridge and smart contract was configured with a value of 4%.

Proof of concept:

- At some point, the bridge fee is set to 6%.
- A user makes a bridge transaction using the web app and sees 5% quoted as the fee percentage.
- After transferring their funds, the user has paid more than they expected.

BVSS:

AO:A/AC:L/AX:L/C:N/I:N/A:N/D:L/Y:N/R:N/S:C (3.1)

Recommendation:

Populate this UI element dynamically according to the percentage value configured in the relevant smart contract(s).

Remediation Plan:

SOLVED: BoostyLabs solved the issue by changing the UI.

4.9 (HAL-09) PRIVATE ETHERSCAN API KEY EXPOSED IN GIT REPOSITORY - LOW (2.5)

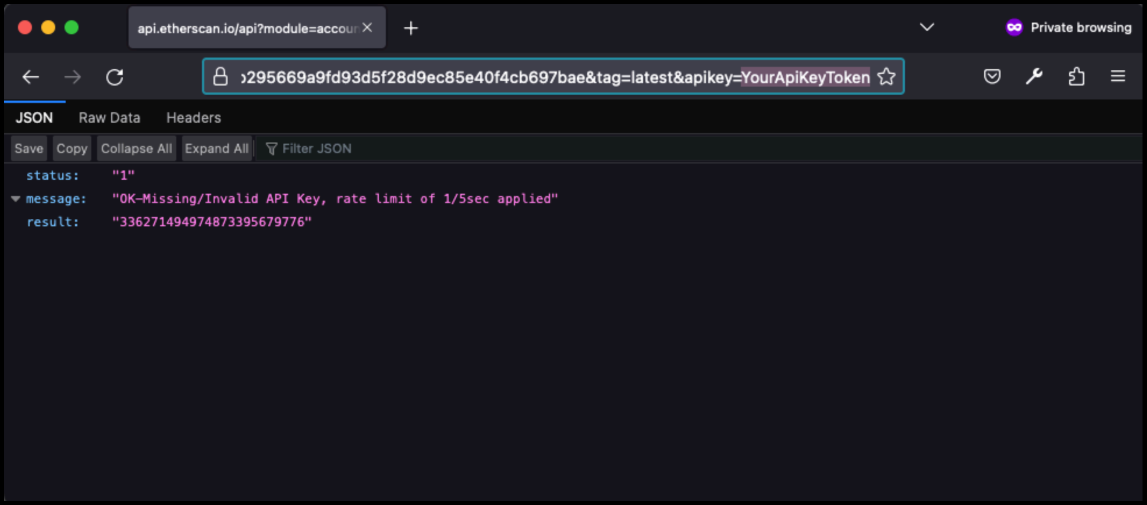
Description:

An Etherscan API key is publicly exposed on GitHub.

Code Location:

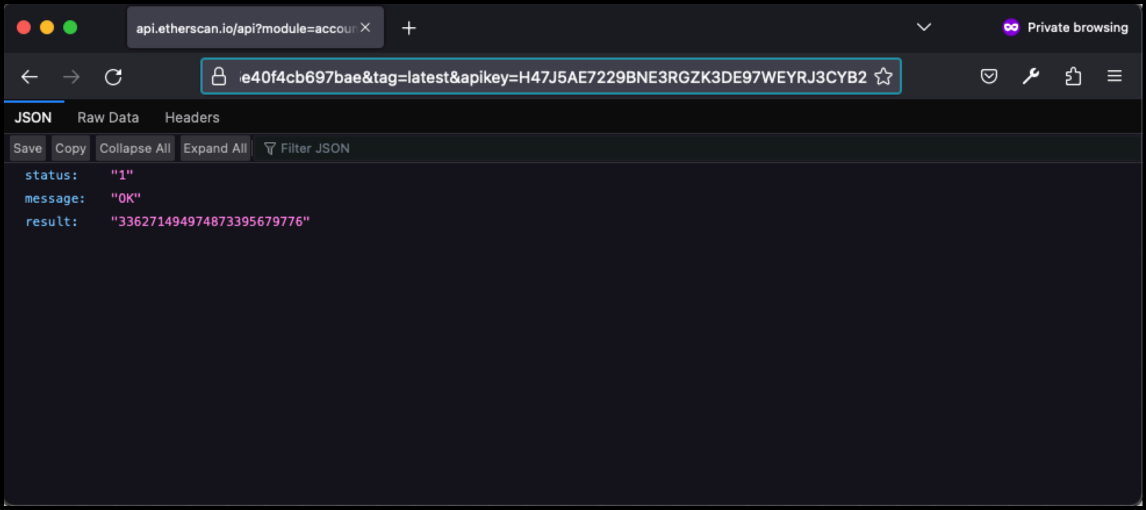
The API key was discovered in the casper-eth-bridge repository in the file boosty-smart-contracts/ethereum/.env.example.

Simple testing was performed to check that the key was active:



```
›295669a9fd93d5f28d9ec85e40f4cb697bae&tag=latest&apikey=YourApiKeyToken
```

```
status:     "1"
message:    "OK-Missing/Invalid API Key, rate limit of 1/5sec applied"
result:     "336271494974873395679776"
```



```
e40f4cb697bae&tag=latest&apikey=H47J5AE7229BNE3RGZK3DE97WEYRJ3CYB2
```

```
status:   "1"
message:  "OK"
result:   "336271494974873395679776"
```

Proof of concept:

- A malicious party discovers the API key in the repository.
- They use the API key to make exhaust the number of tokens that Boosty purchased.
- Any services relying on Etherscan will no longer be accessible.

BVSS:

AO:A/AC:L/AX:L/C:N/I:N/A:M/D:N/Y:N/R:P/S:U (2.5)

Recommendation:

Remove the API key from the repository. Deactivate it within the Etherscan website and generate a new key for operations. Ensure that no private information is committed to GitHub repositories.

Remediation Plan:

SOLVED: The casper-eth-bridge repository is no longer public.

4.10 (HAL-10) LACK OF RATE LIMITING ON THE API - LOW (3.1)

Description:

There is no rate-limiting mechanism configured for the bridge microservices. Without this protection, it may be possible for an attacker to issue a denial-of-service attack against the bridge by flooding the server with many requests. This may have the effect of halting all bridge operations. As the bridge is off-chain, it requires application and network level protections, as it does not benefit from economic incentives present in blockchain software that can reduce spam.

Proof of concept:

- An attacker spams transactions against a resource-intensive feature offered by the bridge.
- The server's resources are exhausted in trying to handle the requests.
- Other users experience degraded performance or a complete halt in operations.

BVSS:

AO:A/AC:H/AX:L/C:N/I:N/A:H/D:N/Y:N/R:N/S:C (3.1)

Recommendation:

Add a rate-limiting mechanism to the API. There are many approaches to this, including using API keys with a fixed number of allowed queries, CAPTCHA or other automation detection tools, and load balancers or other network-level configurations.

Remediation Plan:

SOLVED: Rate limiting has been added in pull request #326.

4.11 (HAL-11) BRIDGE TRANSACTIONS ARE SIGNED BY A SINGLE PRIVATE KEY (CENTRALIZATION RISK) - LOW (3.9)

Description:

The bridge software stores private keys in a database that it uses to sign outgoing transactions for supported networks. This poses a centralization risk, as anyone with access to the database can authorize transactions on behalf of the bridge. The Solidity smart contract used by the bridge exposes a function where the contract owner may withdraw all the bridge funds and commissions. As a result, a compromise of the keys used by the bridge could result in a total and unrecoverable loss of user funds.

Code Location:

Halborn delivered a separate report outlining the details of the Solidity smart contract code. Please refer to this report for further details about the privileged functions controlled by the contract owner.

Proof of Concept:

- A private key is obtained by an attacker through a technical compromise of the database server or via a malicious employee.
- The private key can be used to authorize transactions on behalf of the bridge which include the ability to transfer. funds in and out of the smart contract.

BVSS:

AO:A/AC:L/AX:H/C:N/I:N/A:H/D:H/Y:N/R:N/S:C (3.9)

Recommendation:

Migrate to a multiparty signature scheme so that no single actor in the system can control user funds.

Remediation Plan:

PENDING: BoostyLabs has stated that they will take precautions to store the private keys on AWS servers and will migrate to using a multisignature scheme in the future.

4.12 (HAL-12) LOSS TO PROTOCOL YIELD OR AVAILABILITY DUE TO COMMISSION RATES BETWEEN BRIDGE AND SMART CONTRACT BECOMING DESYNCED - LOW (3.6)

Description:

Both the Go bridge and the Solidity smart contract contain variables used to calculate the commission rate for the bridge protocol. On the Go side, this value is read from a configuration file. In Solidity, it is stored in a contract variable.

These two values serve the same purpose but are set independently. There are no function calls to set the commission rate for the Solidity variable from Go or vice-versa. As a result, these two values must be synced manually.

This may cause problems for the protocol if these values are incorrectly configured. For example, if an admin sets a commission rate lower on the bridge than on the Solidity smart contract, the transaction may fail because insufficient gas will be used in the transaction. If the reverse occurs, the user may avoid paying commission to the protocol.

This latter case may exist even if an admin does remember to configure the values correctly. There may be an interval in which the Go bridge value is updated and deployed before the transaction to change the Solidity percentage is processed and included in the chain. This creates a race condition, where a user of the protocol could monitor for a discrepancy between these values and time transactions to get a discount.

Code Location:

The Solidity contract for the bridge contains a commission percentage for the protocol (400 = 4%). There exists a function for the contract owner to change this value. (For more information, please consult the separate Halborn Solidity Assessment Report for the Tricorn smart contracts).

Within the tricorn Go project, this value is configured via environment variables for the different connectors, e.g.

Listing 11

```
root@BoostyLabs-Test:~/tricorn/configs# grep 'FEE_PERCENTAGE' ./.*
./.avalanche.env:export FEE_PERCENTAGE=0.4
./.bnb.env:export FEE_PERCENTAGE=0.4
./.casper.env:FEE_PERCENTAGE=0.4 # 0.4%
./.eth.env:FEE_PERCENTAGE=0.4
./.polygon.env:FEE_PERCENTAGE=0.4
```

The percentage is then loaded into the codebase and used to calculate the ‘Gas Commission’ for the protocol, e.g.

Listing 12: The environment variable is stored in service.config.FeePercentage

```
// EstimateTransfer estimates transfer fee and time.
func (service *Service) EstimateTransfer(ctx context.Context) (
↳ chains.Estimation, error) {
    gasPrice, err := service.ethClient.SuggestGasPrice(ctx)
    if err != nil {
        return chains.Estimation{}, Error.Wrap(err)
    }

    // TODO: change in dynamic way, as in BridgeOut method.
    gasLimit := new(big.Int).SetUint64(service.config.GasLimit)
    feeWei := gasPrice.Mul(gasPrice, gasLimit)

    fee := new(big.Float).Quo(new(big.Float).SetInt(feeWei), new(
↳ big.Float).SetUint64(wei))

    estimation := chains.Estimation{
        Fee:                       fee.String(),
        FeePercentage:          service.config.FeePercentage,
        EstimatedConfirmation:  service.config.ConfirmationTime,
    }

    return estimation, nil
}
```

Proof of Concept:

- An attacker writes code to simultaneously observe the Bridge contracts and the Bridge API for their FeePercentage values.
- On detecting a difference in the FeePercentages, an attacker submits the transaction.
- The attacker pays a smaller commission due to the differences in these values.

BVSS:

AO:A/AC:L/AX:H/C:N/I:H/A:N/D:N/Y:M/R:N/S:C (3.6)

Recommendation:

Create a mechanism to update the bridge when the smart contract value is changed and vice-versa, or use one as the source of truth. If these values are set separately, minimize the duration for which they are not equal.

Remediation Plan:

SOLVED: The commission percent is removed in PR 375.

4.13 (HAL-13) POSSIBLE DIVISION BY ZERO - LOW (2.1)

Description:

The code uses the Quo method from big.Int without checking that the divisor does not equal 0. This could cause the code to panic in some cases.

Code Location:

tricorn/chains/evm/service.go

Listing 13: If either gasPrice or gasLimit is 0, feeWei will be 0. This will cause a panic when feeWei is used in the Quo function.

```go
// EstimateTransfer estimates transfer fee and time.
func (service *Service) EstimateTransfer(ctx context.Context) (
    chains.Estimation, error) {
    gasPrice, err := service.ethClient.SuggestGasPrice(ctx)
    if err != nil {
        return chains.Estimation{}, Error.Wrap(err)
    }

    // TODO: change in dynamic way, as in BridgeOut method.
    gasLimit := new(big.Int).SetUint64(service.config.GasLimit)
    feeWei := gasPrice.Mul(gasPrice, gasLimit)

    fee := new(big.Float).Quo(new(big.Float).SetInt(feeWei), new(
    big.Float).SetUint64(wei))

    estimation := chains.Estimation{
        Fee:                   fee.String(),
        FeePercentage:         service.config.FeePercentage,
        EstimatedConfirmation: service.config.ConfirmationTime,
    }

    return estimation, nil
}
```

BVSS:

AO:A/AC:L/AX:H/C:N/I:N/A:C/D:N/Y:N/R:P/S:C (2.1)

Recommendation:

Always validate that the divisor is not 0 when using division. It is important to note that this value is not attacker-controlled, so panic would only occur if the gasLimit is misconfigured by an administrator or if there is a bug in the go-ethereum SuggestGasPrice function.

Remediation Plan:

SOLVED: Checks have been added in PR 301 to prevent division by zero.

4.14 (HAL-14) INSECURE PASSWORD POLICY - LOW (3.8)

BVSS:

AO:S/AC:L/AX:L/C:C/I:C/A:C/D:N/Y:N/R:N/S:C (3.8)

Description:

In various places in the repository, the password 1212 is used as a default or test password used to connect to the Postgres database server that holds information such as the private keys or nonce values. As these are critical values for the bridge's security, it is recommended to enforce the use of strong passwords on the application level.

Code Location:

An example of weak passwords being allowed by the application is the file tricorn/bridge/database/dbtesting/run.go. A password of 1212 is used to connect to Postgres.

Listing 14

```go
// DefaultTestConn default test conn string that is expected to
↳ work with postgres server.
const DefaultTestConn = "postgres://postgres:1212@localhost/
↳ boosty_bridge_db?sslmode=disable"

// Run method will establish connection with db, create tables in
↳ random schema, run tests.
func Run(t *testing.T, test func(ctx context.Context, t *testing.T
↳ , db bridge.DB)) {
    t.Run("Postgres", func(t *testing.T) {
        ctx := context.Background()

        options := Database{
            Name: "Postgres",
            URL:  DefaultTestConn,
        }

        masterDB, err := CreateMasterDB(ctx, t.Name(), "Test", 0,
↳ options)
        if err != nil {
            t.Fatal(err)
        }
        defer func() {
            err := masterDB.Close()
            if err != nil {
                t.Fatal(err)
            }
        }()
        err = masterDB.CreateSchema(ctx)
        if err != nil {
            t.Fatal(err)
        }

        test(ctx, t, masterDB)
    })
}
```

BVSS:

AO:S/AC:L/AX:L/C:C/I:C/A:C/D:N/Y:N/R:N/S:C (3.8)

Recommendation:

Enforce the use of secure database passwords within the application. This can help to catch any configuration errors if the database password is set to a weak value.

Remediation Plan:

SOLVED: BoostyLabs solved the issue by adding enforcement.

Pull Request : 340

4.15 (HAL-15) USE OF VULNERABLE DEPENDENCIES - LOW (4.4)

Description:

A variety of vulnerabilities exists in dependencies used by the project, such as those required in go.mod and package.json files.

Code Location:

Vulnerabilities flagged by the tool nancy:

ID	Package	Rating	Description
CVE-2022-44797	btcd	CRITICAL	Improper Restriction of Operation
sonatype-2022-39389	btcd	MEDIUM	Improper Input Validation
CVE-2022-41723	golang x/net	HIGH	Uncontrolled Resource Consumption
CVE-2021-42219	go-ethereum	HIGH	Uncontrolled Resource Consumption
CVE-2022-23328	go-ethereum	HIGH	Uncontrolled Resource Consumption
CVE-2022-37450	go-ethereum	MEDIUM	Improper Input Validation

Listing 15: Excerpt from govulncheck report

```
govulncheck is an experimental tool. Share feedback at https://go.
↳ dev/s/govulncheck-feedback.

Scanning for dependencies with known vulnerabilities...
Found 12 known vulnerabilities.

Vulnerability #1: GO-2022-0524
  Calling Reader.Read on an archive containing a large number of
  concatenated 0-length compressed files can cause a panic due to
  stack exhaustion.

  Call stacks in your code:
      currencyrates/chainlink/chainlinkcurrencyrates.go:64:28:
↳ tricorn/currencyrates/chainlink.Service.GetPrice calls io/ioutil.
↳ ReadAll, which eventually calls compress/gzip.Reader.Read
```

```

  Found in: compress/gzip@go1.18.3
  Fixed in: compress/gzip@go1.18.4
  More info: https://pkg.go.dev/vuln/GO-2022-0524

Vulnerability #2: GO-2023-1570
  Large handshake records may cause panics in crypto/tls. Both
  clients and servers may send large TLS handshake records which
  ...
```

Listing 16: npm audit --no-dev report for tricorn/web/bridge/package.json

```
# npm audit report

json5  2.0.0 - 2.2.1
Severity: high
Prototype Pollution in JSON5 via Parse Method - https://github.com
↳ /advisories/GHSA-9c47-m6qq-7p4h
fix available via `npm audit fix`
node_modules/json5

webpack  5.0.0 - 5.75.0
Severity: high
Cross-realm object access in Webpack 5 - https://github.com/
↳ advisories/GHSA-hc6q-2mpp-qw7j
fix available via `npm audit fix`
node_modules/webpack

2 high severity vulnerabilities

To address all issues, run:
  npm audit fix
```

BVSS:

AO:A/AC:L/AX:L/C:N/I:L/A:L/D:L/Y:L/R:N/S:U (4.4)

Recommendation:

Where possible, keep dependencies patched in order to reduce the risk of the system being attacked using known vulnerabilities. It is recommended that Tricorn Bridge run the nancy, govulncheck, and npm audit tools regularly and fix as many warnings as possible.

Remediation Plan:

SOLVED: Dependencies are upgraded in [PR 302] (https://github.com/BoostyLabs/tricorn/p
and nancy and govulncheck have been added to the CI pipeline.

4.16 (HAL-16) CONVERSION OF STRINGS TO NUMBERS LEAD TO INTEGER OVERFLOWS - LOW (3.1)

BVSS:

AO:A/AC:L/AX:M/C:L/I:L/A:L/D:N/Y:N/R:N/S:C (3.1)

Description:

In many locations in the code, numbers are parsed from strings using the strconv.Atoi function. Later, these values used in type conversions. This can change their sign or truncate values.

Code Location:

One example of this programming pattern exists in the API request for querying supported networks. The following code excerpt and series of HTTP requests and responses demonstrate that an integer overflow occurs due to parsing that is insufficiently validated:

bridge/gateway/controllers/networks.go

Listing 17: One example of the dangerous code pattern: network-is parsed from a string and converted to uint32.

```go
func (controller *Networks) SupportedTokens(w http.ResponseWriter,
↳ r *http.Request) {
    V
    ctx := r.Context()
    w.Header().Set("Content-Type", "application/json")

    params := mux.Vars(r)
    networkID, err := strconv.Atoi(params["network-id"])
    if err != nil {
        controller.serveError(w, http.StatusBadRequest,
↳ ErrNetworks.New("invalid network id"))
```

```
            return
        }

        tokens, err := controller.networks.SupportedTokens(ctx, uint32
↳ (networkID))
        if err != nil {
            controller.log.Error("could not return supported tokens",
↳ ErrNetworks.Wrap(err))
            controller.serveError(w, http.StatusInternalServerError,
↳ ErrNetworks.Wrap(err))
            return
        }

        if err = json.NewEncoder(w).Encode(tokens); err != nil {
            controller.log.Error("failed to write json error response"
↳ , ErrNetworks.Wrap(err))
        }
}
```

Example requests demonstrating the overflow :

Listing 18: Request 1: Query for supported tokens

```
GET /api/v0/networks/4/supported-tokens HTTP/1.1
Host: 138.68.94.229:8088
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv
↳ :109.0) Gecko/20100101 Firefox/112.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,
↳ image/avif,image/webp,*/*;q=0.8
Accept-Language: en-CA,en-US;q=0.7,en;q=0.3
Accept-Encoding: gzip, deflate
Connection: close
Upgrade-Insecure-Requests: 1

```

Listing 19: Response 1: Response displaying supported tokens for network 4

```
HTTP/1.1 200 OK
...

[{"id":1,"shortName":"TST","longName":"TEST","wraps":[{"networkId"
↳ :4,"smartContractAddress":"3
↳ c0c1847d1c410338ab9b4ee0919c181cf26085997ff9c797e8a1ae5b02ddf23"},
↳ {"networkId":5,"smartContractAddress":"0
↳ e26df2baafbc976a104ee3cccf1b467ff1b7a68"},{"networkId":7,"
↳ smartContractAddress":"df7021bfb23da6654f09d2779f930c6c8528c281"},
↳ {"networkId":9,"smartContractAddress":"87
↳ fa9fabb5e32a3ec720cd74f7f1f64b8d3c74cc"},{"networkId":11,"
↳ smartContractAddress":"df7021bfb23da6654f09d2779f930c6c8528c281"}]
↳ }]
```

Listing 20: Request 2: Specifying a network id that does not exist

```
GET /api/v0/networks/1234567890/supported-tokens HTTP/1.1
Host: 138.68.94.229:8088
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:109.0
↳ ) Gecko/20100101 Firefox/112.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,
↳ image/avif,image/webp,*/*;q=0.8
Accept-Language: en-CA,en-US;q=0.7,en;q=0.3
Accept-Encoding: gzip, deflate
Connection: close
Upgrade-Insecure-Requests: 1
```

Listing 21: Response 2: An error 500 response is returned for networks that are not registered

```
HTTP/1.1 500 Internal Server Error
...

{"error":"networks controller: communication: rpc: rpc error: code
↳  = Internal desc = gateway controller: bridge service: network ,
↳ err: network is not connected"}

```

Listing 22: Request 3: Requesting network=MAX_INT64 + 5 in order to get the same results as network=4

```
GET /api/v0/networks/4294967300/supported-tokens HTTP/1.1
```

Listing 23: Response 3: The same results are displayed for network 4 and network 4294967300. The status code is 200, indicating that the network ID is considered valid

```

HTTP/1.1 200 OK
...

[{"id":1,"shortName":"TST","longName":"TEST","wraps":[{"networkId"
↳ :4,"smartContractAddress":"3
↳ c0c1847d1c410338ab9b4ee0919c181cf26085997ff9c797e8a1ae5b02ddf23"},
↳ {"networkId":5,"smartContractAddress":"0
↳ e26df2baafbc976a104ee3cccf1b467ff1b7a68"},{"networkId":7,"
↳ smartContractAddress":"df7021bfb23da6654f09d2779f930c6c8528c281"},
↳ {"networkId":9,"smartContractAddress":"87
↳ fa9fabb5e32a3ec720cd74f7f1f64b8d3c74cc"},{"networkId":11,"
↳ smartContractAddress":"df7021bfb23da6654f09d2779f930c6c8528c281"}]
↳ }]

```

BVSS:

AO:A/AC:L/AX:M/C:L/I:L/A:L/D:N/Y:N/R:N/S:C (3.1)

Recommendation:

Perform validation on numbers that are parsed from strings to ensure that overflow does not occur.

Halborn did not identify any cases with negative impacts to security. However, as more features are added to the protocol, values retrieved in the way should be validated to avoid problems in the future.

Remediation Plan:

SOLVED: Validations are added to the numbers.

4.17 (HAL-17) INFORMATION DISCLOSURE DUE TO WEB SERVER ERROR HANDLING - INFORMATIONAL (0.0)

BVSS:

AO:A/AC:L/AX:L/C:N/I:N/A:N/D:N/Y:N/R:N/S:C (0.0)

Description:

It is possible to generate error messages using the API that disclose that Go and Postgres are in use. This could provide knowledge that an attacker could use in developing exploits against the web server.

Code Location:

Example 1: Reveals Go

Listing 24: Request sent to the API. Required parameter network-id is missing, which causes an error

```
GET /api/v0/transfers/history/bnb/123?offset=0&limit=0 HTTP/1.1
Host: 138.68.94.229:8088
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv
↳ :109.0) Gecko/20100101 Firefox/112.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,
↳ image/avif,image/webp,*/*;q=0.8
Accept-Language: en-CA,en-US;q=0.7,en;q=0.3
Accept-Encoding: gzip, deflate
Connection: close
Upgrade-Insecure-Requests: 1

```

Listing 25: strconv error message sent to client, indicating that Go is used for the back-end

```
HTTP/1.1 400 Bad Request
Content-Type: application/json
Vary: Origin
Date: Tue, 11 Apr 2023 17:15:00 GMT
Content-Length: 107
Connection: close

{"error":"transfers controller: network-id parameter invalid;
↳ strconv.Atoi: parsing \"\": invalid syntax"}
```

Example 2: Reveals postgres

Listing 26: Sending negative number for limit parameter

```
GET /api/v0/transfers/history/34850b7e36e635783df..?offset=1&limit
↳ =-1&network-id=7 HTTP/1.1
```

Listing 27: Postgres error message appears

```
HTTP/1.1 500 Internal Server Error
Content-Type: application/json
Vary: Origin
Date: Tue, 11 Apr 2023 17:44:02 GMT
Content-Length: 244
Connection: close

{"error":"transfers controller: transfers service: communication:
↳ rpc: rpc error: code = Internal desc = gateway controller: bridge
↳ service: master database: sql: converting argument $3 type: uint64
↳  values with high bit set are not supported"}
```

BVSS:

AO:A/AC:L/AX:L/C:N/I:N/A:N/D:N/Y:N/R:N/S:C (0.0)

Recommendation:

Modify the error handling to return generic messages that do not reveal the software stack in use. Note that this is only an issue for closed-source projects. If the project is released publicly in the future, then there is no benefit to changing the error messages.

Remediation Plan:

SOLVED: Changes introduced in PR 304 prevent information disclosure by returning custom errors.

4.18 (HAL-18) REVERSE.STRINGS DOES NOT FUNCTION CORRECTLY - INFORMATIONAL (0.0)

Description:

The codebase contains a utility function to reverse strings. However, it does not behave correctly for UTF-8 strings.

Code Location:

internal/reverse/reverse.go

Listing 28

```go
// Copyright (C) 2022 Creditor Corp. Group.
// See LICENSE for copying information.

package reverse

// Bytes takes a bytes as argument and return the reverse of bytes
↳ .
func Bytes(value []byte) []byte {
    for i, j := 0, len(value)-1; i < j; i, j = i+1, j-1 {
        value[i], value[j] = value[j], value[i]
    }

    return value
}

// String takes a string as argument and return the reverse of
↳ string.
func String(str string) string {
    var result string
    for _, v := range str {
        result = string(v) + result
    }
    return result
}
```

This vulnerability was discovered using fuzz testing. Further details can be found in the Automated Testing section at the end of this report.

BVSS:

AO:A/AC:L/AX:L/C:N/I:N/A:N/D:N/Y:N/R:N/S:C (0.0)

Recommendation:

Delete this code from the project, or fix the error so that UTF-8 strings are handled properly. It is important to note that this code is not called in the project.

Remediation Plan:

SOLVED: The function is removed in PR 305.

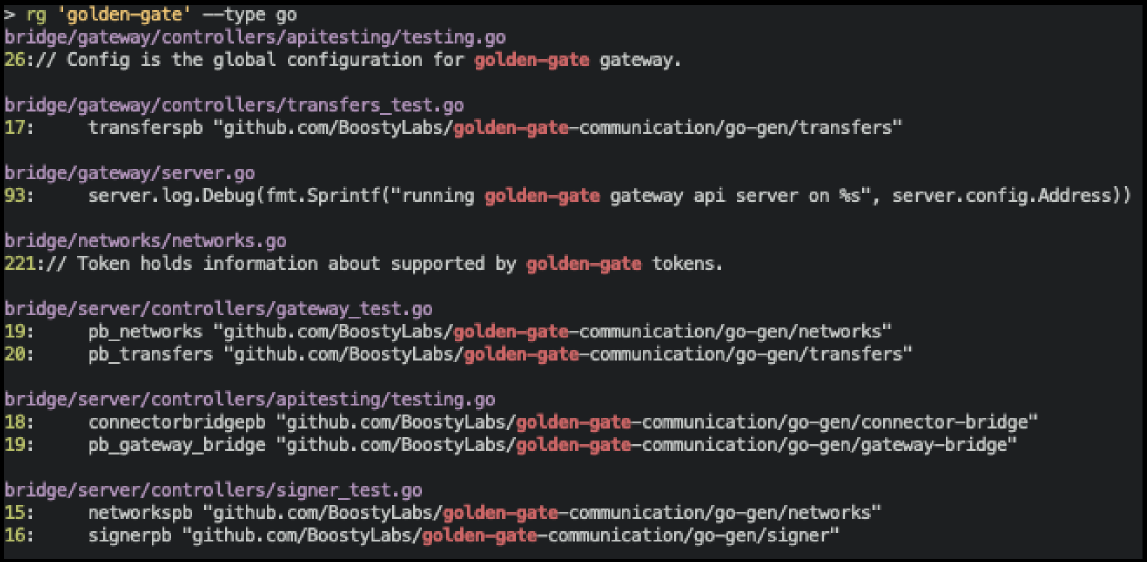
4.19 (HAL-19) OUTDATED REFERENCES TO A REPOSITORY THAT HAS BEEN RENAMED - INFORMATIONAL (0.0)

Description:

The original scope of the assessment on a set of repositories called Golden Gate and Golden Gate Communication, which were renamed during the assessment to Tricorn and Tricorn Communication. However, there are still many references to golden-gate in the Go package imports as well as in code comments.

As a result, the project was unable to build until custom modifications were made to the project.

Code Location:



```
> rg 'golden-gate' --type go
bridge/gateway/controllers/apitesting/testing.go
26:// Config is the global configuration for golden-gate gateway.

bridge/gateway/controllers/transfers_test.go
17:     transferspb "github.com/BoostyLabs/golden-gate-communication/go-gen/transfers"

bridge/gateway/server.go
93:     server.log.Debug(fmt.Sprintf("running golden-gate gateway api server on %s", server.config.Address))

bridge/networks/networks.go
221:// Token holds information about supported by golden-gate tokens.

bridge/server/controllers/gateway_test.go
19:     pb_networks "github.com/BoostyLabs/golden-gate-communication/go-gen/networks"
20:     pb_transfers "github.com/BoostyLabs/golden-gate-communication/go-gen/transfers"

bridge/server/controllers/apitesting/testing.go
18:     connectorbridgepb "github.com/BoostyLabs/golden-gate-communication/go-gen/connector-bridge"
19:     pb_gateway_bridge "github.com/BoostyLabs/golden-gate-communication/go-gen/gateway-bridge"

bridge/server/controllers/signer_test.go
15:     networkspb "github.com/BoostyLabs/golden-gate-communication/go-gen/networks"
16:     signerpb "github.com/BoostyLabs/golden-gate-communication/go-gen/signer"
```

Figure 4: Remaining references to golden-gate

BVSS:

AO:A/AC:L/AX:L/C:N/I:N/A:N/D:N/Y:N/R:N/S:C (0.0)

Recommendation:

It is advised to update all remaining references to golden-gate, especially if the project's source code is published or if it is shared with other developers or auditors. This will improve the build process and help to convey a sense of professionalism.

During the assessment, the naming issues were resolved by adding a replace directive within the project's go.mod file:

Listing 29

```
replace github.com/BoostyLabs/golden-gate-communication/go-gen =>
↳ ../tricorn-communication/go-gen
```

This can be used as a temporary solution if build errors occur. In the future, the imports and comments should be updated to refer to the new repository name.

Remediation Plan:

SOLVED: The reference is removed on the related code base.

4.20 (HAL-20) UNIT TESTS MISSING - INFORMATIONAL (0.0)

Description:

The project makes use of unit tests, but there are some gaps in coverage for certain error conditions or code branches.

Code Location:

Example 1: ToEVMSignature

Listing 30: ToEVMSignature can return errors in some cases.

```go
// ToEVMSignature reforms last two byte of signature from 00, 01
↳ to 1b, 1c.
func ToEVMSignature(signature []byte) ([]byte, error) {
    if len(signature) != SignatureLength {
        return nil, fmt.Errorf("signature length isn't %d bytes",
↳ SignatureLength)
    }

    if signature[SignatureLength-1] != byte(evmsignature.
↳ PrivateKeyVZero) &&
        signature[SignatureLength-1] != byte(evmsignature.
↳ PrivateKeyVOne) {
        return nil, errors.New("signature is wrong")
    }

    signature[SignatureLength-1] += byte(evmsignature.
↳ PrivateKeyVTwentySeven)

    return signature, nil
}
```

The unit tests for ToEVMSignature do not ensure that errors appear when expected, and do not test for signatures ending with 01.

Listing 31: Unit tests does not check error cases or a signature ending in 01

```
    t.Run("ToEVMSignature", func(t *testing.T) {
        signature, err := hex.DecodeString("
↳ aac4e8036a01f1b83ad304f8738e5cf0cf99db\
↳ ne114cb05083ec89ce18c5860844ec4e8036a01f1b83ad304f87/
↳ n38e5cf0cf99dbe114cb05083ec89ce18c58608400")
        require.NoError(t, err)

        ethSignedMessageHash, err := evm.ToEVMSignature(signature)
        require.NoError(t, err)

        expectedHash, err := hex.DecodeString("
↳ aac4e8036a01f1b83ad304f8738e5cf0cf99dbe1/
↳ n14cb05083ec89ce18c5860844ec4e8036a01f1b83ad304f8738/
↳ ne5cf0cf99dbe114cb05083ec89ce18c5860841b")
        require.NoError(t, err)

        require.Equal(t, expectedHash, ethSignedMessageHash)
    })
```

Example 2: TestEventParsing checks only the happy path:

Listing 32

```
func TestEventParsing(t *testing.T) {
    eventData := eventparsing.EventData{
        Bytes: "7c000000013c0c1847d1c410338ab9b4ee0919c181c/
↳ nf26085997ff9c797e8a1ae5b02ddf2306000000474f45524c49280000003330/
↳ n39356639353564613730623936323135636666633962633634616232626/
↳ n5363965623764616203a0860100daa2b596e0a496b04933e241e0/
↳ n567f2bcbecc829aa57d88cab096c28fd07dee2",
    }

    expectedEventTyte := 1
    expectedTokenContractAddress := "3
↳ c0c1847d1c410338ab9b4ee0919c181cf26085997ff9c797e8a1ae5b02ddf23"
    expectedChainName := "GOERLI"
    expectedChainAddress := "3095
↳ f955da700b96215cffc9bc64ab2e69eb7dab"
    expectedAmount := 100000
    expectedUserWalletAddress := "
↳ daa2b596e0a496b04933e241e0567f2bcbecc829aa57d88cab096c28fd07dee2"
```

```go

    t.Run("GetEventType", func(t *testing.T) {
        actualEventTyte, err := eventData.GetEventType()
        require.NoError(t, err)
        require.Equal(t, expectedEventTyte, actualEventTyte)
    })

    t.Run("GetTokenContractAddress", func(t *testing.T) {
        actualTokenContractAddress := eventData.
↳ GetTokenContractAddress()
        require.Equal(t, expectedTokenContractAddress,
↳ actualTokenContractAddress)
    })

    t.Run("GetChainName", func(t *testing.T) {
        actualChainName, err := eventData.GetChainName()
        require.NoError(t, err)
        require.Equal(t, expectedChainName, actualChainName)
    })

    t.Run("GetChainAddress", func(t *testing.T) {
        actualChainAddress, err := eventData.GetChainAddress()
        require.NoError(t, err)
        require.Equal(t, expectedChainAddress, actualChainAddress)
    })

    t.Run("GetAmount", func(t *testing.T) {
        actualAmount, err := eventData.GetAmount()
        require.NoError(t, err)
        require.Equal(t, expectedAmount, actualAmount)
    })

    t.Run("GetUserWalletAddress", func(t *testing.T) {
        actualUserWalletAddress := eventData.GetUserWalletAddress
↳ ()
        require.Equal(t, expectedUserWalletAddress,
↳ actualUserWalletAddress)
    })
}
```

BVSS:

AO:A/AC:L/AX:L/C:N/I:N/A:N/D:N/Y:N/R:N/S:C (0.0)

Recommendation:

Projects that provide bridging functionality are an extremely valuable target for attackers. They are also highly complex. For these reasons, extensive unit tests should be considered a requirement for a protocol that contains bridging functionality.

Remediation Plan:

SOLVED: BoostyLabs added tests to address the gaps in coverage in PR 310.

4.21 (HAL-21) LACK OF FUZZ TESTS - INFORMATIONAL (0.0)

Description:

Fuzz testing is a testing technique to simulate a wide range of potential system states as well as data inputs in order to determine whether critical properties of the system remain true. Properly used, fuzz testing can provide excellent support to unit tests and other integration tests.

BVSS:

AO:A/AC:L/AX:L/C:N/I:N/A:N/D:N/Y:N/R:N/S:C (0.0)

Recommendation:

For a project providing bridging functionality, fuzz testing should be considered a requirement. It is recommended that the Boosty and Casper teams to agree on a set of invariants for the system and then incorporate them into fuzz testing. Example invariants might include:
- After depositing tokens, if a user withdraws tokens, the amount they receive in the withdrawal should not exceed the deposit amount
- Users should not be able to withdraw unless they have first done a deposit

Remediation Plan:

SOLVED: BoostyLabs added fuzz tests with the Pull Request 383.

4.22 (HAL-22) GO VERSION 1.18 IS UNSUPPORTED - INFORMATIONAL (0.0)

Description:

The project uses Go version 1.18. This version has been deprecated. See the Go release notes for their policy on supporting major versions of Go.

Code Location:

go.mod

Listing 33

```
module tricorn

go 1.18
```

BVSS:

AO:A/AC:L/AX:L/C:N/I:N/A:N/D:N/Y:N/R:N/S:C (0.0)

Recommendation:

Update to a supported version of Go in order to receive ongoing security updates.

Remediation Plan:

SOLVED: Dependencies are upgraded in [PR 302] (https://github.com/BoostyLabs/tricorn/p

4.23 (HAL-23) GO VERSIONS PRIOR TO 1.20 ARE MORE VULNERABLE TO SIDE-CHANNEL ATTACKS - INFORMATIONAL (0.0)

Description:

Go version 1.20.2 contains security and performance enhancements. Specifically, this release fixes problems in cryptographic libraries. Older versions of go are more susceptible to cryptography issues and side-channel attacks on cryptographic implementations.

Code Location:

go.mod

Listing 34

```
module tricorn

go 1.18
```

BVSS:

AO:A/AC:L/AX:L/C:N/I:N/A:N/D:N/Y:N/R:N/S:C (0.0)

Recommendation:

Update to at least Go v1.20.2.

Remediation Plan:

SOLVED: Dependencies are upgraded in [PR 302] (https://github.com/BoostyLabs/tricorn/p

4.24 (HAL-24) INCORRECT ERROR MESSAGE FOR HISTORY ENDPOINT IN GATEWAY - INFORMATIONAL (0.0)

Description:

The History endpoint's error message contains text about an unrelated transaction.

Code Location:

bridge/gateway/controllers/transfers.go

Listing 35: History endpoint returns an error message about canceling a transaction

```
    history, err := controller.transfers.History(ctx, uint64(
↳ offset), uint64(limit), signature, pubKey, uint32(networkID))
    if err != nil {
        controller.log.Error("could not cancel pending transfer",
↳ ErrTransfers.Wrap(err))
        controller.serveError(w, http.StatusInternalServerError,
↳ ErrTransfers.Wrap(err))
        return
    }
```

BVSS:

AO:A/AC:L/AX:L/C:N/I:N/A:N/D:N/Y:N/R:N/S:C (0.0)

Recommendation:

Correct the error message to give information about the History endpoint.

Remediation Plan:

SOLVED: The error message has been corrected.

4.25 (HAL-25) TODOS IN CODEBASE - INFORMATIONAL (0.0)

Description:

Numerous code comments in the codebase contain TODO messages or other developer notes indicating malfunctioning or missing functionality.

Code Location:

Listing 36: Instances of TODO in the tricorn repository. Many messages indicate features that are absent

```
chains/solana/connector.go:22:  GasLimit                  uint64
↳        `env:"GAS_LIMIT"` // TODO: count by tx.
chains/communication/controllers/connector_test.go:200:     //
↳ TODO: add remaining methods.
chains/chain.go:43: // TODO: get rid of what is below.
chains/solana/service.go:53:// TODO: place token contract to token
↳ .
chains/solana/service.go:67:    // TODO: read from db mb.
chains/solana/service.go:73:    // TODO implement.
chains/solana/service.go:79:    // TODO implement.
chains/solana/service.go:85:    // TODO implement.
chains/solana/service.go:96:    // TODO: add implementation.
chains/solana/service.go:102:   // TODO: implement.
chains/solana/service.go:156:   // TODO: add implementation.
chains/evm/connector.go:23:// TODO: add gasLimit for In and Out.
chains/evm/connector.go:37: GasLimit                          uint64
↳         `env:"GAS_LIMIT"` // TODO: count by tx.
chains/communication/controllers/connector.go:273:  // TODO:
↳ uncomment after casper fixing.
chains/communication/controllers/connector.go:278:  // // TODO:
↳ give wei from bridge db.
chains/evm/service.go:117:  // TODO: read from db mb.
chains/evm/service.go:143:          // TODO: fix it.
chains/evm/service.go:187:  // TODO: change in dynamic way, as in
↳ BridgeOut method.
chains/casper/connector.go:62:// TODO: add gasLimit for In and Out
↳ .
```

```
currencyrates/service.go:59:        // TODO: get token symbol
↳ dynamically.
currencyrates/service.go:68:            Decimals:   18, // TODO:
↳ get it from db table.
chains/casper/service.go:76:// TODO: place token contract to token
↳ .
chains/casper/service.go:96:    // TODO: read from db mb.
chains/casper/service.go:385:       // TODO: add in event later.
chains/casper/service.go:393:       // TODO: add in event later.
chains/casper/service.go:401:       // TODO: add in event later.
chains/casper/service.go:437:       // TODO: add in event later.
chains/casper/service.go:728:   // TODO: add implementation.
currencyrates/server/controllers/currencyrates_test.go:39:
↳         // TODO: add stream values checks.
chains/communication/controllers/apitesting/testing.go:189:
↳             // TODO: fix it.
web_app/server.go:38:   // TODO: remove after new version of
↳ Casper wallet be released.
bridge/gateway/controllers/transfers_test.go:30:// TODO: test with
↳  err sent from mock service.
bridge/gateway/controllers/transfers.go:44: // TODO: implement
↳ transaction validation.
bridge/gateway/controllers/transfers.go:80: // TODO: implemenmt
↳ amount validation.
bridge/networks/networks.go:30:// TODO: place TokenContract to
↳ Token type.
bridge/networks/networks.go:37: NodeAddress string `json:"
↳ nodeAddress"` // TODO: delete after casper wallet release 2.
bridge/networks/networks.go:42: GasLimit uint64 `json:"gasLimit"`
↳ // TODO: comment why do we need it.
bridge/networks/networks.go:242:// TODO: place to signatures
↳ package after bridge will be rewritten.
bridge/networks/networks.go:267:// TODO: place to signatures
↳ package after bridge will be rewritten.
bridge/server/controllers/gateway_test.go:157:      // TODO:
↳ Extent test with Solana network after txHash type change.
bridge/bridge_test.go:257:              TokenID:          1, //
↳ todo: dynamically change.
bridge/service.go:329:  // TODO: uncomment after fix.
bridge/service.go:363:  // TODO: uncomment after fix.
bridge/service.go:400:      TokenID:            1, // todo:
↳ dynamically change.
bridge/service.go:580:      token, err := service.networkTokens.
↳ Get(ctx, recipientNetworkID, 1) // TODO: add dynamic token id.
```

```
bridge/service.go:608:      TokenID:          1, // todo:
↳ dynamically change.
bridge/service.go:679:      TokenID:          1, // todo:
↳ dynamically change.
cmd/avalanche/main.go:126:          // TODO: fix it.
cmd/avalanche/main.go:165:              // TODO: fix it.
cmd/eth/main.go:127:            // TODO: fix it.
cmd/eth/main.go:166:                // TODO: fix it.
cmd/bnb/main.go:126:            // TODO: fix it.
cmd/bnb/main.go:165:                // TODO: fix it.
cmd/polygon/main.go:126:            // TODO: fix it.
cmd/polygon/main.go:165:                // TODO: fix it.
cmd/bridge/main.go:127: // TODO: replace with migrations.
cmd/signer/main.go:97:  // TODO: remove for production.
```

BVSS:

AO:A/AC:L/AX:L/C:N/I:N/A:N/D:N/Y:N/R:N/S:C (0.0)

Recommendation:

It is recommended to use a separate issue tracker or other task management software to track bugs and features rather than using code comments. Developer notes in comments are very likely to be overlooked and to become out of date relative to the code.

If the source code is shared publicly, such developer notes indicate areas of confusion or complexity which may be leveraged by an attacker reading the code.

Remediation Plan:

ACKNOWLEDGED: BoostyLabs acknowledged the issue.

4.26 (HAL-26) INCONSISTENCY IN API VALUES - INFORMATIONAL (0.0)

Description:

The ordering for certain constants and enums are defined in an inconsistent ordering. This may lead to subtle logic issues if a developer assumes, for example, that values related to the EVM are always 0, Casper is 1, Solana is 2 and so on.

Code Location:

tricorn/networks/networks.go

Listing 37

```go
const (
    // TypeIDEVM describes EVM compatible network id.
    TypeIDEVM TypeID = 0
    // TypeIDCasper describes Casper network id.
    TypeIDCasper TypeID = 1
    // TypeIDSolana describes Solana network id.
    TypeIDSolana TypeID = 2
)

// NetworkIDToNetworkType describes id-to-type ratio for network.
var NetworkIDToNetworkType = map[TypeID]Type{
    TypeIDEVM:    TypeEVM,
    TypeIDCasper: TypeCasper,
    TypeIDSolana: TypeSolana,
}

// NetworkTypeToNetworkID describes type-to-id ratio for network.
var NetworkTypeToNetworkID = map[Type]TypeID{
    TypeEVM:    TypeIDEVM,
    TypeCasper: TypeIDCasper,
    TypeSolana: TypeIDSolana,
}

```

```
// ID defines list of possible blockchain network interoperability
 ↳  ids.
type ID int

const (
    // IDCasper describes Casper network id.
    IDCasper ID = 0
    // IDEth describes Eth network id.
    IDEth ID = 1
    // IDSolana describes Solana network id.
    IDSolana ID = 2
```

BVSS:

AO:A/AC:L/AX:L/C:N/I:N/A:N/D:N/Y:N/R:N/S:C (0.0)

Recommendation:

Consider using a consistent ordering when declaring these values in order to avoid developer confusion.

Remediation Plan:

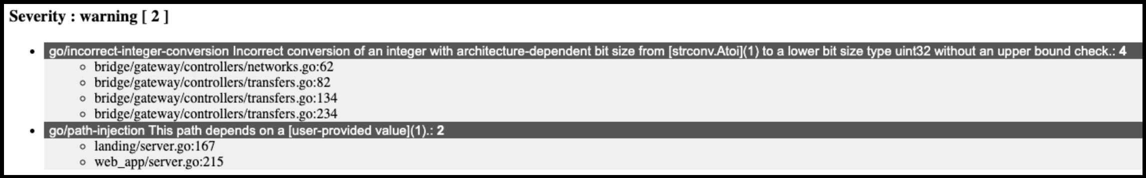
SOLVED: The ordering has been made consistent in PR 311.

AUTOMATED TESTING

5.1 Automated Testing -- Overview

Halborn used automated testing techniques to enhance coverage of certain areas of the scoped component. Among the tools used were codeql, gosec, govulncheck, nancy, and npm audit. After Halborn verified all the modules 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.

5.2 codeql



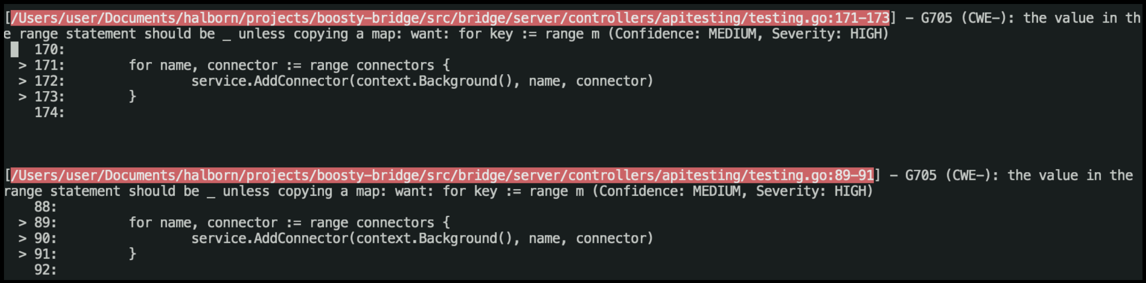
Severity : warning [2]

- go/incorrect-integer-conversion Incorrect conversion of an integer with architecture-dependent bit size from [strconv.Atoi](1) to a lower bit size type uint32 without an upper bound check.: 4
  - bridge/gateway/controllers/networks.go:62
  - bridge/gateway/controllers/transfers.go:82
  - bridge/gateway/controllers/transfers.go:134
  - bridge/gateway/controllers/transfers.go:234
- go/path-injection This path depends on a [user-provided value](1).: 2
  - landing/server.go:167
  - web_app/server.go:215

Figure 5: CodeQL results

5.3 gosec

The following as an excerpt from running the tool gosec:



```
[/Users/user/Documents/halborn/projects/boosty-bridge/src/bridge/server/controllers/apitesting/testing.go:171-173] - G705 (CWE-): the value in th
e range statement should be _ unless copying a map: want: for key := range m (Confidence: MEDIUM, Severity: HIGH)
   170:
 > 171:        for name, connector := range connectors {
 > 172:                service.AddConnector(context.Background(), name, connector)
 > 173:        }
   174:



[/Users/user/Documents/halborn/projects/boosty-bridge/src/bridge/server/controllers/apitesting/testing.go:89-91] - G705 (CWE-): the value in the
range statement should be _ unless copying a map: want: for key := range m (Confidence: MEDIUM, Severity: HIGH)
   88:
 > 89:        for name, connector := range connectors {
 > 90:                service.AddConnector(context.Background(), name, connector)
 > 91:        }
   92:
```

Figure 6: gosec excerpt

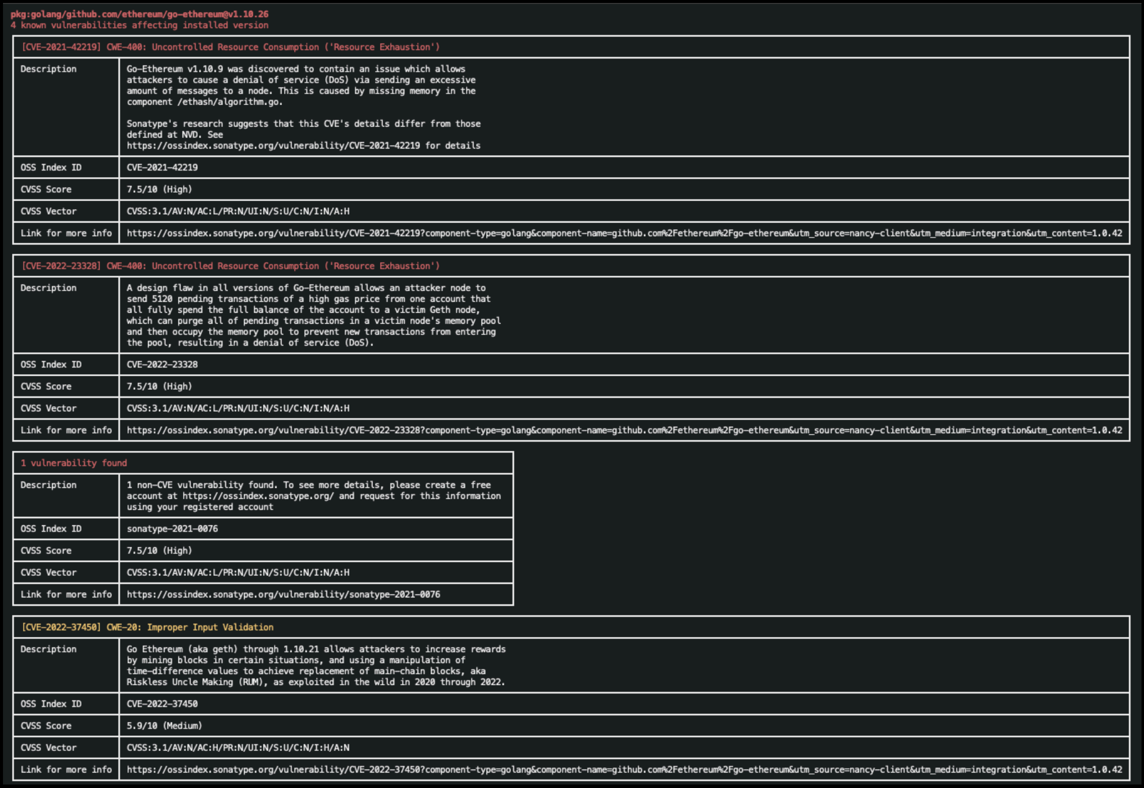
5.4 govulncheck

Listing 38: Excerpt of analysis results for tricorn

```
govulncheck is an experimental tool. Share feedback at https://go.
↳ dev/s/govulncheck-feedback.

Scanning for dependencies with known vulnerabilities...
Found 12 known vulnerabilities.

Vulnerability #1: GO-2022-0524
  Calling Reader.Read on an archive containing a large number of
  concatenated 0-length compressed files can cause a panic due to
  stack exhaustion.

  Call stacks in your code:
      currencyrates/chainlink/chainlinkcurrencyrates.go:64:28:
↳ tricorn/currencyrates/chainlink.Service.GetPrice calls io/ioutil.
↳ ReadAll, which eventually calls compress/gzip.Reader.Read

  Found in: compress/gzip@go1.18.3
  Fixed in: compress/gzip@go1.18.4
  More info: https://pkg.go.dev/vuln/GO-2022-0524

Vulnerability #2: GO-2023-1570
...
```

5.5 nancy

The tool nancy was used to search for known vulnerabilities within project dependencies. Here is an excerpt of the output from this tool:



```
pkg:golang/github.com/ethereum/go-ethereum@v1.10.26
4 known vulnerabilities affecting installed version

[CVE-2021-42219] CWE-400: Uncontrolled Resource Consumption ('Resource Exhaustion')
Description       | Go-Ethereum v1.10.9 was discovered to contain an issue which allows
                  | attackers to cause a denial of service (DoS) via sending an excessive
                  | amount of messages to a node. This is caused by missing memory in the
                  | component /ethash/algorithm.go.
                  |
                  | Sonatype's research suggests that this CVE's details differ from those
                  | defined at NVD. See
                  | https://ossindex.sonatype.org/vulnerability/CVE-2021-42219 for details
OSS Index ID      | CVE-2021-42219
CVSS Score        | 7.5/10 (High)
CVSS Vector       | CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:N/A:H
Link for more info| https://ossindex.sonatype.org/vulnerability/CVE-2021-42219?component-type=golang&component-name=github.com%2Fethereum%2Fgo-ethereum&utm_source=nancy-client&utm_medium=integration&utm_content=1.0.42

[CVE-2022-23328] CWE-400: Uncontrolled Resource Consumption ('Resource Exhaustion')
Description       | A design flaw in all versions of Go-Ethereum allows an attacker node to
                  | send 5120 pending transactions of a high gas price from one account that
                  | all fully spend the full balance of the account to a victim Geth node,
                  | which can purge all of pending transactions in a victim node's memory pool
                  | and then occupy the memory pool to prevent new transactions from entering
                  | the pool, resulting in a denial of service (DoS).
OSS Index ID      | CVE-2022-23328
CVSS Score        | 7.5/10 (High)
CVSS Vector       | CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:N/A:H
Link for more info| https://ossindex.sonatype.org/vulnerability/CVE-2022-23328?component-type=golang&component-name=github.com%2Fethereum%2Fgo-ethereum&utm_source=nancy-client&utm_medium=integration&utm_content=1.0.42

1 vulnerability found
Description       | 1 non-CVE vulnerability found. To see more details, please create a free
                  | account at https://ossindex.sonatype.org/ and request for this information
                  | using your registered account
OSS Index ID      | sonatype-2021-0076
CVSS Score        | 7.5/10 (High)
CVSS Vector       | CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:N/A:H
Link for more info| https://ossindex.sonatype.org/vulnerability/sonatype-2021-0076

[CVE-2022-37450] CWE-20: Improper Input Validation
Description       | Go Ethereum (aka geth) through 1.10.21 allows attackers to increase rewards
                  | by mining blocks in certain situations, and using a manipulation of
                  | time-difference values to achieve replacement of main-chain blocks, aka
                  | Riskless Uncle Making (RUM), as exploited in the wild in 2020 through 2022.
OSS Index ID      | CVE-2022-37450
CVSS Score        | 5.9/10 (Medium)
CVSS Vector       | CVSS:3.1/AV:N/AC:H/PR:N/UI:N/S:U/C:N/I:H/A:N
Link for more info| https://ossindex.sonatype.org/vulnerability/CVE-2022-37450?component-type=golang&component-name=github.com%2Fethereum%2Fgo-ethereum&utm_source=nancy-client&utm_medium=integration&utm_content=1.0.42
```

Figure 7: Nancy excerpt

5.6 npm audit

Listing 39: Analysis results for package.json

```
# npm audit report

json5  2.0.0 - 2.2.1
Severity: high
Prototype Pollution in JSON5 via Parse Method - https://github.com
↳ /advisories/GHSA-9c47-m6qq-7p4h
fix available via `npm audit fix`
node_modules/json5

webpack  5.0.0 - 5.75.0
Severity: high
Cross-realm object access in Webpack 5 - https://github.com/
↳ advisories/GHSA-hc6q-2mpp-qw7j
fix available via `npm audit fix`
node_modules/webpack
```

```

2 high severity vulnerabilities

To address all issues, run:
  npm audit fix
```

5.7 Fuzz Testing

Fuzz testing, also known as fuzzing, is a software testing technique that involves inputting large amounts of random data, or “fuzz,” into a program to see how it reacts and if it can handle unexpected or invalid input. The goal of fuzz testing is to identify and prevent software vulnerabilities, such as buffer overflows and memory leaks, by exposing the program to a wide range of inputs that it may not have been designed to handle.

Halborn performed fuzz testing on the following functionality:
- Event parsing defined in the file internal/eventparsing/eventparsing.go
- Strings reverse logic defined in the file internal/reverse/reverse.go

This function was selected because it contains custom logic to parse events for the bridge, which in turn can trigger transactions. For this reason, event parsing represents a critical function for the bridge.

Event Parsing:

Fuzz Harness The following fuzz harness was added to internal/eventparsing/eventparsing_test.go:

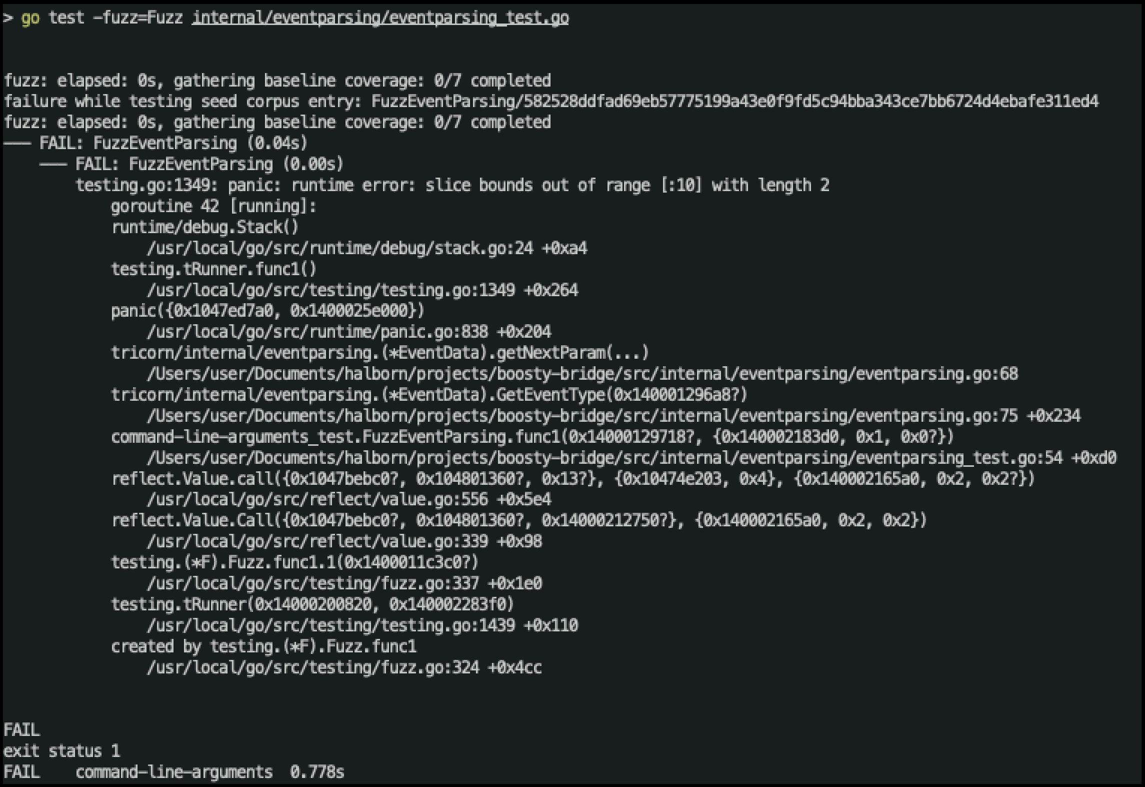
Listing 40: Fuzz harness with test cases and properties defined

```go
func FuzzEventParsing(f *testing.F) {
    // Note: All of these inputs cause a panic due to 'slice
↳ bounds out of range'
    // Create seed corpus
    //testcases := [][]byte{
```

```
    //  []byte(""),          // empty input
    //  []byte("\n"),         // string seperator
    //  []byte("00"),         // TagAccount sequence from
↳ eventparsing.go
    //  []byte("01"),         // TagHash sequence from
↳ eventparsing.go
    //  []byte("0|"),         // Caused a crash during testing
    //  []byte("|"),         // Caused a crash during testing
    //  []byte("0000000000"), // TagAccount sequence from
↳ eventparsing.go
    //}

    //for _, tc := range testcases {
    //  f.Add(tc) // Add seeds from testcases
    //}

    // Use the original event from the unit test as a seed corpus

    event, _ :=hex.DecodeString("string")

    f.Add(event)
    f.Fuzz(func(t *testing.T, input []byte) {

        hexBytes := hex.EncodeToString(input)
        // If the following conditions are added to the test,
↳ crashes do not occur.
        //if len(input) < len(event) {
        //  // Try rejecting short events so that we are in the
↳ range of an appropriate length
        //  return
        //}
        //if hexBytes[2:] != "7c" {
        //  //if len(input) < 1 || hexBytes[2:] != "7c" {
        //  // Try rejecting events that do not begin with 7c in
↳ case this is significant
        //  return
        //}
        eventData := eventparsing.EventData{
            Bytes: hexBytes,
        }
        // Check for panics or crashes
        eventData.GetEventType()
        eventData.GetTokenContractAddress() // function does not
↳ return err
```

```
            eventData.GetChainName()
            eventData.GetChainAddress()
            eventData.GetAmount()
            eventData.GetUserWalletAddress() // function does not
↳ return err
      })
}
```

Results Halborn identified numerous crashes while fuzzing this function. An example is pictured below:



```
> go test -fuzz=Fuzz internal/eventparsing/eventparsing_test.go


fuzz: elapsed: 0s, gathering baseline coverage: 0/7 completed
failure while testing seed corpus entry: FuzzEventParsing/582528ddfad69eb57775199a43e0f9fd5c94bba343ce7bb6724d4ebafe311ed4
fuzz: elapsed: 0s, gathering baseline coverage: 0/7 completed
--- FAIL: FuzzEventParsing (0.04s)
    --- FAIL: FuzzEventParsing (0.00s)
        testing.go:1349: panic: runtime error: slice bounds out of range [:10] with length 2
            goroutine 42 [running]:
            runtime/debug.Stack()
                /usr/local/go/src/runtime/debug/stack.go:24 +0xa4
            testing.tRunner.func1()
                /usr/local/go/src/testing/testing.go:1349 +0x264
            panic({0x1047ed7a0, 0x1400025e000})
                /usr/local/go/src/runtime/panic.go:838 +0x204
            tricorn/internal/eventparsing.(*EventData).getNextParam(...)
                /Users/user/Documents/halborn/projects/boosty-bridge/src/internal/eventparsing/eventparsing.go:68
            tricorn/internal/eventparsing.(*EventData).GetEventType(0x140001296a8?)
                /Users/user/Documents/halborn/projects/boosty-bridge/src/internal/eventparsing/eventparsing.go:75 +0x234
            command-line-arguments_test.FuzzEventParsing.func1(0x140001297l8?, {0x140002183d0, 0x1, 0x0?})
                /Users/user/Documents/halborn/projects/boosty-bridge/src/internal/eventparsing/eventparsing_test.go:54 +0xd0
            reflect.Value.call({0x1047bebc0?, 0x104801360?, 0x13?}, {0x10474e203, 0x4}, {0x140002165a0, 0x2, 0x2?})
                /usr/local/go/src/reflect/value.go:556 +0x5e4
            reflect.Value.Call({0x1047bebc0?, 0x104801360?, 0x140002127507?}, {0x140002165a0, 0x2, 0x2})
                /usr/local/go/src/reflect/value.go:339 +0x98
            testing.(*F).Fuzz.func1.1(0x1400011c3c0?)
                /usr/local/go/src/testing/fuzz.go:337 +0x1e0
            testing.tRunner(0x14000200820, 0x140002283f0)
                /usr/local/go/src/testing/testing.go:1439 +0x110
            created by testing.(*F).Fuzz.func1
                /usr/local/go/src/testing/fuzz.go:324 +0x4cc


FAIL
exit status 1
FAIL    command-line-arguments  0.778s
```

Figure 8: Results of fuzz testing

String Reversal:

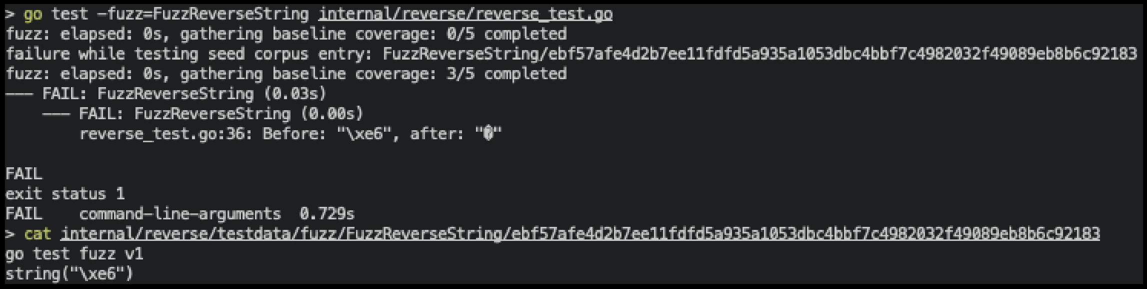
The codebase contains an unused function used to reverse strings. It was determined that this function cannot properly reverse UTF-8 strings.

Fuzz Harness

Listing 41

```go
package reverse_test

import (
    "testing"
    "tricorn/internal/reverse"
    "unicode/utf8"
)

func FuzzReverseString(f *testing.F) {
    // A UTF-8 character with multiple bytes should be used as
↳ input here
    testcases := []string{"Hello, world", " ", "!12345", ""}
    for _, tc := range testcases {
        f.Add(tc) // Use f.Add to provide a seed corpus
    }
    f.Fuzz(func(t *testing.T, orig string) {
        rev := reverse.String(orig)
        doubleRev := reverse.String(rev)
        if orig != doubleRev {
            t.Errorf("Before: %q, after: %q", orig, doubleRev)
        }
        if utf8.ValidString(orig) && !utf8.ValidString(rev) {
            t.Errorf("reverse.String reverse produced invalid UTF
↳ -8 string %q", rev)
        }
    })
}
```

Results



```
> go test -fuzz=FuzzReverseString internal/reverse/reverse_test.go
fuzz: elapsed: 0s, gathering baseline coverage: 0/5 completed
failure while testing seed corpus entry: FuzzReverseString/ebf57afe4d2b7ee11fdfd5a935a1053dbc4bbf7c4982032f49089eb8b6c92183
fuzz: elapsed: 0s, gathering baseline coverage: 3/5 completed
--- FAIL: FuzzReverseString (0.03s)
    --- FAIL: FuzzReverseString (0.00s)
        reverse_test.go:36: Before: "\xe6", after: "�"

FAIL
exit status 1
FAIL    command-line-arguments  0.729s
> cat internal/reverse/testdata/fuzz/FuzzReverseString/ebf57afe4d2b7ee11fdfd5a935a1053dbc4bbf7c4982032f49089eb8b6c92183
go test fuzz v1
string("\xe6")
```

Notes on Fuzz Testing:

There are several limitations to fuzz testing when done with a time constraint, especially regarding security:

- Limited coverage: When done with a time constraint, the amount of fuzzing that can be done is limited, which means that the software may not be fully tested and vulnerabilities may go undetected.
- False negatives: Fuzz testing may not be able to uncover all vulnerabilities in the software, especially when done with a time constraint. This is particularly true for complex software systems, where a significant amount of time is required to uncover all possible vulnerabilities.
- Limited scope: Only a section of the codebase was fuzzed.

THANK YOU FOR CHOOSING

HALBORN