



# Avalanche Warden

## Security Audit

Prepared by: Halborn

Date of Engagement: May 26th , 2021 - June 16th, 2021

Visit: [Halborn.com](https://Halborn.com)

DOCUMENT REVISION HISTORY	5
CONTACTS	6
1 EXECUTIVE OVERVIEW	7
1.1 INTRODUCTION	8
1.2 AUDIT SUMMARY	8
1.3 TEST APPROACH & METHODOLOGY	9
RISK METHODOLOGY	9
1.4 SCOPE	11
2 ASSESSMENT SUMMARY & FINDINGS OVERVIEW	12
3 FINDINGS & TECH DETAILS	13
4 AVA LABS WARDEN FINDINGS & TECH DETAILS	15
4.1 (HAL-01) HARDCODED CREDENTIALS IN GIT COMMITS - HIGH	16
Description	16
Risk Level	18
Code Location	18
Recommendation	19
Remediation Plan	19
4.2 (HAL-02) UNPRIVILEGED SECRET SHARE MODIFICATION - MEDIUM	20
Description	20
Risk Level	20
Code Location	21
Recommendation	23
Remediation Plan	23
4.3 (HAL-03) LACK OF WARDEN SECURITY HARDENING RECOMMENDATIONS - MEDIUM	24

Description	24
Risk Level	24
Recommendation	24
Remediation Plan	25
4.4 (HAL-04) DEFAULT PERMISSIONS TOO LAX - LOW	26
Description	26
Risk Level	26
Code Location	26
Recommendation	27
Recommendation Plan	27
4.5 (HAL-05) UNHANDLED ERRORS - LOW	28
Description	28
Risk Level	28
Code Location	28
Recommendation	29
Remediation Plan	29
4.6 (HAL-06) SENSITIVE CREDENTIALS DISCOVERED IN MEMORY - LOW	30
Description	30
MEMORY ANALYSIS	30
Risk Level	30
Remediation Plan	31
4.7 (HAL-07) LACK OF INPUT VALIDATION - INFORMATIONAL	32
Description	32
Risk Level	32
Code Location	33

Recommendation	34
Remediation Plan	35
4.8 (HAL-08) MISSING HTTP SECURITY HEADERS ON WARDEN REQUESTS - INFORMATIONAL	36
Description	36
Code Location	37
Recommendation	37
Remediation Plan	38
4.9 (HAL-09) VERBOSE ERROR MESSAGES - INFORMATIONAL	39
Description	39
Risk Level	39
Code Location	39
Recommendation	40
Remediation Plan	40
4.10 (HAL-10) SHADOWED VARIABLES - INFORMATIONAL	41
Description	41
Risk Level	41
Code Location	42
Recommendation	42
Remediation Plan	42
4.11 (HAL-11) MISSING GO COMPILER BUILD DIRECTIVES - INFORMATIONAL	43
Description	43
Risk Level	43
Repository Makefile Flags	43
Example Flags	44

	Recommendation	44
	Remediation Plan	44
4.12	(HAL-12) INEFFECTUAL VARIABLE ASSIGNMENT - INFORMATIONAL	45
	Description	45
	Risk Level	45
	Code Location	46
	Example Code	46
	Recommendation	47
	Remediation Plan	47
5	STATIC ANALYSIS REPORT	48
5.1	STATIC ANALYSIS	49
	Test Coverage	49
	Gosec - Security Analysis Output Sample	50
	Staticcheck - Security Analysis Output Sample	52
	Ineffassign - Security Analysis Output Sample	52
	Unconvert - Security Analysis Output Sample	53
6	FUZZING REPORT	54
6.1	FUZZING REPORT	55
6.2	PROPERTY BASED FUZZING	55
	Example Test Case	55
	Fuzzing Progress	56

## DOCUMENT REVISION HISTORY

VERSION	MODIFICATION	DATE	AUTHOR
0.1	Document Creation	05/24/2021	Gabi Urrutia
0.2	Document Edits	06/08/2021	Ataberk Yavuzer
0.5	Document Edits	06/11/2021	Gokberk Gulgun
0.9	Document Edits	06/11/2021	Piotr Cielas
1.0	Final Draft	06/18/2021	Ataberk Yavuzer
1.1	Remediation Plan	07/16/2021	Ataberk Yavuzer
1.1	Remediation Plan Review	07/21/2021	Gabi Urrutia

# CONTACTS

CONTACT	COMPANY	EMAIL
Rob Behnke	Halborn	<a href="mailto:Rob.Behnke@halborn.com">Rob.Behnke@halborn.com</a>
Steven Walbroehl	Halborn	<a href="mailto:Steven.Walbroehl@halborn.com">Steven.Walbroehl@halborn.com</a>
Gabi Urrutia	Halborn	<a href="mailto:Gabi.Urrutia@halborn.com">Gabi.Urrutia@halborn.com</a>
Ataberk Yavuzer	Halborn	<a href="mailto:Ataberk.Yavuzer@halborn.com">Ataberk.Yavuzer@halborn.com</a>
Gokberk Gulgun	Halborn	<a href="mailto:Gokberk.Gulgun@halborn.com">Gokberk.Gulgun@halborn.com</a>
Piotr Cielas	Halborn	<a href="mailto:Piotr.Cielas@halborn.com">Piotr.Cielas@halborn.com</a>



# EXECUTIVE OVERVIEW





## 1.1 INTRODUCTION

AVA Labs Bridge & Warden components are designed to integrate a diverse set of blockchains specialised for transferring assets between Avalanche and Ethereum protocols. AVA Labs Warden controls these transfer operations and indexes them on a database.

AVA Labs engaged Halborn to conduct a security assessment on their multiple structures beginning on May 26th and ending on June 16th, 2021. The security assessment was scoped to the Warden repository. An audit of the security risk and implications regarding the changes introduced by the development team at AVA Labs prior to its production release shortly following the assessments deadline.

Though this security audit's outcome is satisfactory, the most essential aspects were tested and verified to achieve objectives and deliverables set in the scope of the audit. Several libraries and components that are imported for the Warden, such as the Intel SGX secure Enclave, are descope from this audit. It is essential to note the use of the best practices for secure Warden development.

## 1.2 AUDIT SUMMARY

The team at Halborn was provided three weeks for the engagement and assigned three full time security engineers to audit the security of the smart contract. The security engineers are blockchain and smart-contract security experts with advanced penetration testing, smart-contract hacking, and deep knowledge of multiple blockchain protocols.

The purpose of this audit to achieve the following:

- Ensure that Warden functions are intended.
- Identify potential security issues with the Warden structure.

In summary, Halborn identified few security risks, and recommends performing further testing to validate extended safety and correctness in context to the whole structure.

## 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 structures. While manual testing is recommended to uncover flaws in logic, process, and implementation; automated testing techniques help enhance coverage of structures and can quickly identify items that do not follow security best practices. The following phases and associated tools were used throughout the term of the audit:

- Research into architecture and purpose.
- Static Analysis of security for scoped structures, and imported functions. (`gosec`, `shadow`, `staticcheck`, `errcheck`)
- Manual Assessment for discovering security vulnerabilities on code-bases.
- Dynamic Analysis on Warden structure.
- Memory Analysis of security for Warden. (`gcore`)
- Test deployment of scoped structures.

### RISK METHODOLOGY:

Vulnerabilities or issues observed by Halborn are ranked based on the risk assessment methodology by measuring the **LIKELIHOOD** of a security incident, and the **IMPACT** should an incident occur. This framework works for communicating the characteristics and impacts of technology vulnerabilities. It's quantitative model ensures repeatable and accurate measurement

while enabling users to see the underlying vulnerability characteristics that was used to generate the Risk scores. For every vulnerability, a risk level will be calculated on a scale of 5 to 1 with 5 being the highest likelihood or impact.

#### RISK SCALE - LIKELIHOOD

- 5 - Almost certain an incident will occur.
- 4 - High probability of an incident occurring.
- 3 - Potential of a security incident in the long term.
- 2 - Low probability of an incident occurring.
- 1 - Very unlikely issue will cause an incident.

#### RISK SCALE - IMPACT

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

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

CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL
----------	------	--------	-----	---------------

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

## 1.4 SCOPE

IN-SCOPE:

The security assessment was scoped to `bridge-warden` repository:

### Bridge-Warden Commit IDs

First tested: `5f0549ca7a60b9d99bac6ffdb436b21728839e26`

Last tested: `29f09851570e0c8700e6af380a4e93ec3eea1507`

## 2. ASSESSMENT SUMMARY & FINDINGS OVERVIEW

CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL
0	1	2	3	6

### LIKELIHOOD

IMPACT

	(HAL-03)		(HAL-01)	
(HAL-04) (HAL-05)		(HAL-02)		
(HAL-08)				
(HAL-09) (HAL-10) (HAL-11) (HAL-12)	(HAL-07)	(HAL-06)		

SECURITY ANALYSIS	RISK LEVEL	REMEDIATION DATE
HAL01 - HARDCODED CREDENTIALS IN GIT HISTORY	High	SOLVED: 06/10/2021
HAL02 - UNPRIVILEGED SECRET SHARE MODIFICATION	Medium	SOLVED: 06/17/2021
HAL03 - LACK OF WARDEN SECURITY HARDENING RECOMMENDATIONS	Medium	SOLVED: 07/16/2021
HAL04 - DEFAULT PERMISSIONS TOO LAX	Low	NOT APPLICABLE: 07/16/2021
HAL05 - UNHANDLED ERRORS	Low	RISK ACCEPTED: 07/16/2021
HAL06 - SENSITIVE CREDENTIALS DISCOVERED IN MEMORY	Low	ACKNOWLEDGED: 07/16/2021
HAL07 - LACK OF INPUT VALIDATION	Informational	RISK ACCEPTED: 07/16/2021
HAL08 - MISSING HTTP SECURITY HEADERS ON WARDEN REQUESTS	Informational	NOT APPLICABLE: 07/16/2021
HAL09 - VERBOSE ERROR MESSAGES	Informational	SOLVED: 07/16/2021
HAL10 - SHADOWED VARIABLES	Informational	ACKNOWLEDGED: 07/16/2021
HAL11 - MISSING GO COMPILER BUILD DIRECTIVES	Informational	NOT APPLICABLE: 07/16/2021
HAL12 - INEFFECTUAL VARIABLE ASSIGNMENT	Informational	ACKNOWLEDGED: 07/16/2021



# FINDINGS & TECH DETAILS





# AVA LABS WARDEN FINDINGS & TECH DETAILS





## 4.1 (HAL-01) HARDCODED CREDENTIALS IN GIT COMMITS - HIGH

### Description:

In the original commit being audited by Halborn, it was discovered that there are multiple critical hardcoded credentials in `/bridge-warden/domain/configuration.go`. Using these credentials, it would be possible to take advantage of AWS S3 and Azure Storages. Also, Warden creates a new file on `db_setup.sh` script when user provides MySQL Username and password to store these variables on a file (`/etc/warden/warden.conf`). If somehow an attacker breaches to the system, the attacker can claim MySQL username and password from that file.

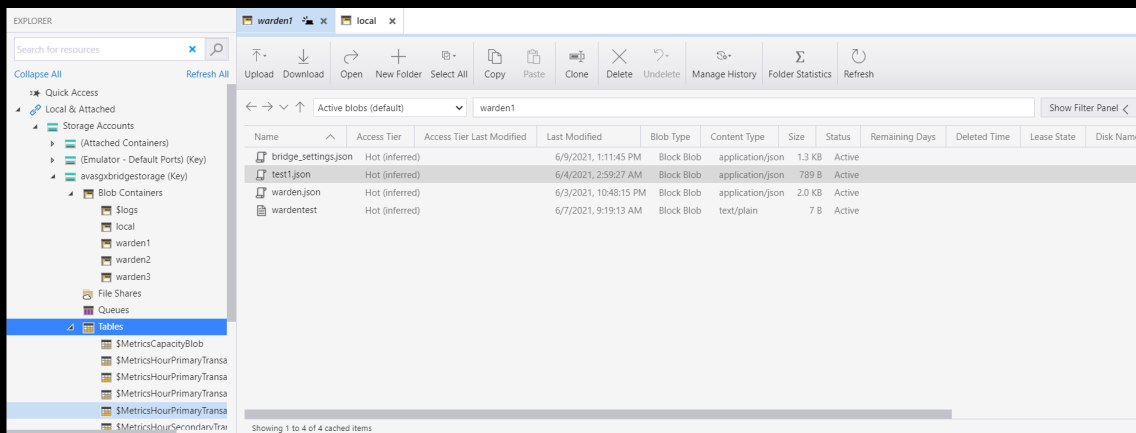
Since the original finding, the credentials have been removed from the master branch, and the developers intend to use AWS Secrets Manager to retrieve and store credentials securely. However, these credentials can still be seen in the Git History.

```
λ aws configure
AWS Access Key ID [None]: AK
AWS Secret Access Key [None]: o
Default region name [None]: us-east-1
Default output format [None]:

λ aws s3 ls
2020-11-24 09:13:23
2021-05-21 21:52:09 bridge-wardens-dev
2021-01-10 22:37:50
2020-04-12 03:21:57
2017-11-19 20:43:13
2019-09-27 03:26:51
2019-05-24 00:37:21
2017-10-18 06:23:50
2020-04-08 20:21:20
2019-06-06 22:50:12
2019-07-02 19:41:46
2019-09-27 03:26:39

λ aws s3 ls bridge-wardens-dev
PRE failure_case/
PRE success_case/
2021-06-04 05:28:50 789 test.json
```

Figure 1: Hardcoded Credentials - 01



Name	Access Tier	Access Tier Last Modified	Last Modified	Blob Type	Content Type	Size	Status	Remaining Days	Deleted Time	Lease State	Disk Name
bridge_settings.json	Hot (inferred)		6/9/2021, 1:11:45 PM	Block Blob	application/json	1.3 KB	Active				
test1.json	Hot (inferred)		6/4/2021, 2:59:27 AM	Block Blob	application/json	789 B	Active				
warden.json	Hot (inferred)		6/3/2021, 10:48:15 PM	Block Blob	application/json	2.0 KB	Active				
warden.test	Hot (inferred)		6/7/2021, 9:19:13 AM	Block Blob	text/plain	7 B	Active				

Figure 2: Hardcoded Credentials - 02

```
→ SmartContracts gIt:(main) X cat /etc/warden/warden.conf
```

```
{
  "apiPort": "6123",
  "apiHttps": false,
  "apiIpFilter": true,
  "bridgeIp": "127.0.0.1",
  "sslCertFile": "/etc/ssl/warden/cert.pem",
  "sslKeyFile": "/etc/ssl/warden/key.pem",
  "networks": {
    "ethereum": {
      "endpoint": "http://127.0.0.1:5050",
      "minConfirmations": 30,
      "startBlock": 10,
      "blockPollIntervalSeconds": 15
    },
    "avalanche": {
      "endpoint": "http://127.0.0.1:9650/ext/bc/C/rpc",
      "minConfirmations": 1,
      "startBlock": 0,
      "blockPollIntervalSeconds": 3
    }
  },
  "useAwsSecretsManager": false,
  "awsSecretName": "warden/database/mysql",
  "awsRegion": "us-east-1",
  "allowUnencryptedSecrets": true,
  "useChainlinkPriceFeeds": false,
  "chainlinkEthUsdFeedAddress": "0x",
  "chainlinkEthGasPriceFeedAddress": "0x"
}
```

Figure 3: Hardcoded Credentials - 03

Risk Level:

Likelihood - 4

Impact - 4

Code Location:

Listing	1:	domain/configuration/configuration.go	(Lines
			102,103,106,107)
81	wc := WardenConfiguration{		
82	APIPort:	"4325",	
83	APIHTTPS:	false,	
84	APIIPFilter:	true,	
85	SSLCertFile:	"/etc/ssl/warden/cert.pem",	
		,	
86	SSLKeyFile:	"/etc/ssl/warden/key.pem"	
		,	
87	BridgeIP:	"127.0.0.1",	

```

88     BlobStorageCORSOrigins:      []string{"REDACTED"},
89     Networks:                    defaultNetworks,
90     UseChainlinkPriceFeeds:      false,
91     ChainlinkETHUSDFeedAddress:  "REDACTED",
92     ChainlinkETHGasPriceFeedAddress: "REDACTED",
93     MySQLUser:                   "REDACTED",
94     MySQLPassword:               "REDACTED",
95     MySQLIP:                     "127.0.0.1",
96     UseAWSSecretsManager:        false,
97     AWSSecretName:               "REDACTED",
98     AWSRegion:                   "us-east-1",
99     AllowUnencryptedSecrets:     false,
100    PricesRefreshIntervalSeconds: 15,
101    BlobStorageLocation:          BlobStorageAzure,
102    S3ID:                         "REDACTED",
103    S3Secret:                      "REDACTED",
104    S3BucketName:                 "REDACTED",
105    AzureStorageAccountName:      "REDACTED",
106    AzureContainerName:           "REDACTED",
107    AzureStorageKey:              "REDACTED",
108    }

```

#### Recommendation:

Use secure vault to store credentials instead of using them hardcoded. Also, It is recommended to delete hardcoded credentials from the Git history, and change any previously used passwords to prevent future incidents if these credentials have been leaked, or reused elsewhere.

#### Remediation Plan:

**SOLVED:** AVA Labs Team solved this issue by removing these credentials from the repository. In addition, all data related to storage and IAM permissions was removed.

## 4.2 (HAL-02) UNPRIVILEGED SECRET SHARE MODIFICATION - MEDIUM

### Description:

The warden stores a secret share on MySQL database to interact with Bridge. The bridge derives different kind of keys from the secret share value. These keys are AES SALT, AES KEY, ETHEREUM PRIVATE KEY and AVALANCHE PRIVATE KEY. The main problem is any anonymous user can change this secret share variable by sending POST request to /shares endpoint of Warden instance.

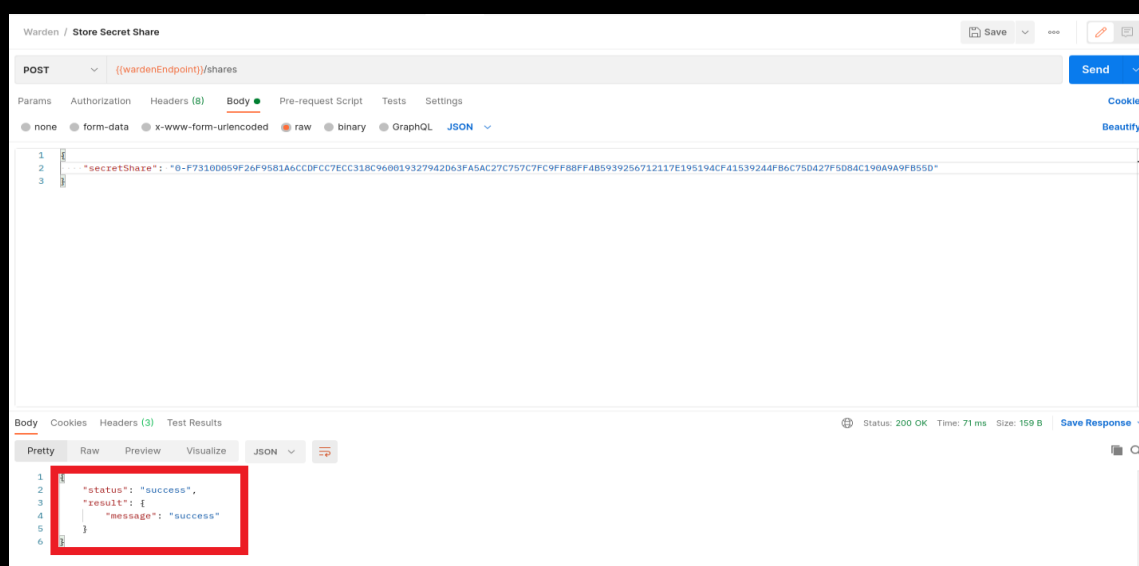


Figure 4: Unprivileged Secret Share Modification

### Risk Level:

Likelihood - 3

Impact - 3

## Code Location:

Listing 2: warden.go (Lines 80,81)

```
78 apiRouter := mux.NewRouter()
79     apiRouter.HandleFunc("/settings", handler.SetBridgeSettings).
        Methods("POST")
80     apiRouter.HandleFunc("/shares", handler.SetSecretShare).
        Methods("POST")
81     apiRouter.HandleFunc("/shares", handler.GetSecretShare).
        Methods("GET")
82     apiRouter.HandleFunc("/nonces", handler.GetNetworkNonce).
        Methods("GET")
83     apiRouter.HandleFunc("/prices/gas", handler.GetGasPrice).
        Methods("GET")
84     apiRouter.HandleFunc("/prices", handler.GetCurrentPrices).
        Methods("GET")
85     apiRouter.HandleFunc("/pairings", handler.GetAllAssetPairs).
        Methods("GET")
86     apiRouter.HandleFunc("/assets", handler.CreateAssetPair).
        Methods("POST")
87     apiRouter.HandleFunc("/transfers", handler.
        GetUnprocessedTransfers).Methods("GET")
88     apiRouter.HandleFunc("/unwraps", handler.GetUnprocessedUnwraps
        ).Methods("GET")
89     apiRouter.HandleFunc("/prepares", handler.GetPreparedRequests)
        .Methods("GET")
90     apiRouter.HandleFunc("/prepares", handler.PrepareRequest).
        Methods("POST")
91     apiRouter.HandleFunc("/completes", handler.CompleteRequest).
        Methods("POST")
92     apiRouter.HandleFunc("/requests", handler.GetRequest).Methods(
        "GET")
93     apiRouter.HandleFunc("/fees", handler.
        GetCurrentOperatorFeeAmounts).Methods("GET")
94     apiRouter.HandleFunc("/fees/prepares", handler.
        GetPreparedFeeTransactions).Methods("GET")
95     apiRouter.HandleFunc("/fees/prepares", handler.
        PrepareFeeTransaction).Methods("POST")
96     apiRouter.HandleFunc("/fees/completes", handler.
        CompleteFeeTransaction).Methods("POST")
97     apiRouter.HandleFunc("/status", handler.GetStatus(
        gitCommitHash)).Methods("GET")
```

Listing 3: handlers/handler.go (Lines 99)

```

66 func (h *Handler) SetSecretShare(w http.ResponseWriter, request *
    http.Request) {
67     log.Println("Serving SetSecretShare request.")
68     decoder := json.NewDecoder(request.Body)
69     defer request.Body.Close()
70
71     var sss contracts.SetSecretShareRequest
72     var err error
73     if configuration.GlobalConfiguration.AllowUnencryptedSecrets {
74         err = decoder.Decode(&sss)
75     } else {
76         var er contracts.EncryptedRequest
77         err := decoder.Decode(&er)
78         if err != nil || len(er.Body) == 0 {
79             log.Println("Malformed SetSecretShare encrypted
                request.")
80             sendResponse(w, false, contracts.ErrorResponse{
                ErrorMessage: "Malformed request"})
81             return
82         }
83         decryptedBody, err := secrets.DecryptRequest(er)
84         if err != nil || len(decryptedBody) == 0 {
85             log.Println("Error decrypting secret share request
                body.")
86             sendResponse(w, false, contracts.ErrorResponse{
                ErrorMessage: "Error decrypting body."})
87             return
88         }
89         err = json.Unmarshal(decryptedBody, &sss)
90     }
91
92     if err != nil || len(sss.SecretShare) == 0 {
93         log.Println("Error unmarshaling secret share.")
94         sendResponse(w, false, contracts.ErrorResponse{
            ErrorMessage: "Error unmarshaling secret share."})
95         return
96     }
97
98     // Store the new secret share in the database. Old secret
        share values are still kept but no longer used.
99     err = h.db.SetSecretShare(sss.SecretShare)

```

#### Recommendation:

An authentication mechanism should be implemented to protect important endpoints such as “/shares”.

#### Remediation Plan:

**SOLVED:** AVA Labs Team solved this issue by using Remote Attestation.



## 4.3 (HAL-03) LACK OF WARDEN SECURITY HARDENING RECOMMENDATIONS – MEDIUM

### Description:

The wardens are intended to be a set of “nodes” that facilitate the consensus and validation of transactions for the Avalanche Network. These wardens will be built, hosted, configured, and maintained by participants of the network. There will most likely be many different types of platforms and networks that will host these wardens, such as Azure, Digital Ocean, Self-Hosted Data centers etc.

Because there are disparate systems, and the security of the network relies on these wardens, and the participants to keep them secure and active, hardening guidelines must also be provided to recommend configurations and best practices for security.

### Risk Level:

**Likelihood - 2**

**Impact - 4**

### Recommendation:

It is the responsibility of the warden owners to make sure security is properly addressed; but Ava Labs should recommend several specifics that are applicable to the Warden infrastructure, such as Firewall or blocking the ports that may be publicly exposed, such as MySQL (TCP 1433) or to use strong passwords that cannot be guessed or brute forced. In addition to that, implementing an authorization mechanism to critical endpoints is suggested.

**Reference:** [Server Hardening Best Practices](#)

## Remediation Plan:

**SOLVED:** AVA Labs Team, supported by Halborn, built a reliable structure to solve this issue.

## 4.4 (HAL-04) DEFAULT PERMISSIONS TOO LAX - LOW

### Description:

The Warden sets too lax permissions (0644) on the configuration file it creates. This file contains sensitive data which can be read by all local users.

```
-rw-r--r-- 1 root root 1051 Jun  9 08:53 /etc/warden/warden.conf
```

Figure 5: Default Permissions Too Lax

### Risk Level:

**Likelihood - 1**

**Impact - 3**

### Code Location:

#### Listing 4: domain/settings/settings\_manager.go (Lines 232)

```
226 switch configuration.GlobalConfiguration.BlobStorageLocation {
227     case configuration.BlobStorageS3:
228         _, err = blob_storage.UploadToS3(combinedSettingsBytes,
229             BridgeSettingsFile)
229     case configuration.BlobStorageAzure:
230         _, err = blob_storage.UploadToAzure(combinedSettingsBytes,
231             BridgeSettingsFile)
231     case configuration.BlobStorageLocal:
232         err = ioutil.WriteFile(BridgeSettingsFile,
233             combinedSettingsBytes, 0644)
233     default:
234         panic("Invalid blob storage location when saving settings
235             file.")
235 }
```

#### Recommendation:

Change the default permissions to more strict in order to restrict the access to the owner only (e.g. `0600`). This can be done with setup commands, or configuration steps using \*NIX `chown` `chmod` utilities.

#### Recommendation Plan:

**NOT APPLICABLE:** AVA Labs Team considers keeping the mask value as `0644` because `0644` permissions are required for warden to interact with the file.

## 4.5 (HAL-05) UNHANDLED ERRORS - LOW

### Description:

The Warden does not properly anticipate or handle exceptional conditions that could occur during normal operation of the software. For example, if the program tries to work with a `nil` value, then the application will give an error. Furthermore, in a possible panic situation, the program will terminate itself. This situation can lead to Denial of Service due to correctly unhandled errors.

### Risk Level:

**Likelihood - 1**

**Impact - 3**

### Code Location:

#### Listing 5: domain/secrets/secrets.go

```
58 mac.Write(req.Request)
```

#### Listing 6: domain/database/database.go

```
204 func (db *WardenDatabase) CloseDatabase() {
205     db.getCurrentSecretShareStmt.Close()
206     db.insertSecretShareStmt.Close()
207     db.insertTxStmt.Close()
208     db.prepareRequestStmt.Close()
209     db.getRequestStmt.Close()
210     db.completeRequestStmt.Close()
211     db.getPreparedRequestBlobsStmt.Close()
212     db.getUnprocessedTransfersStmt.Close()
213     db.getUnprocessedUnwrapsStmt.Close()
214     db.getMaxIndexedBlockStmt.Close()
215     db.getAllAssetPairsStmt.Close()
216     db.getAssetPairByNativeStmt.Close()
```

```
217     db.getAssetPairByWrappedStmt.Close()
218     db.insertAssetPairStmt.Close()
219     db.getOperatorFeeAmountsStmt.Close()
220     db.getPreparedFeeTransactionBlobsStmt.Close()
221     db.prepareFeeTransactionStmt.Close()
222     db.completeFeeTransactionStmt.Close()
223     db.insertBridgeSettingsStmt.Close()
224     db.getLatestBridgeSettingsStmt.Close()
225     db.db.Close()
226 }
```

#### Recommendation:

Implement error handling for the function calls above to simplify the debugging process in case they fail.

Example:

#### Listing 7: domain/secrets/secrets.go

```
58 mac.Write(req.Request)
59
60 if err != nil {
61     // handle the error
62 }
```

Reference: [Error handling in GO](#)

#### Remediation Plan:

**RISK ACCEPTED:** AVA Labs Team assumes that the application will not fail in the above codes.

## 4.6 (HAL-06) SENSITIVE CREDENTIALS DISCOVERED IN MEMORY - LOW

### Description:

The Warden executable uses private keys and clear-text credentials in the memory. In some cases, it is possible for an attacker to acquire the necessary privileges to dump the memory contents of an arbitrary user process by exploiting a vulnerable service. In summary, these credentials should not be reachable by dumping the memory. With this knowledge, Halborn team tried to fetch important information from the memory after running the applications.

### MEMORY ANALYSIS:

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

```
→ database_scripts glt:(main) X strings warden_dump.16763 | grep 'password'
```




Figure 6: Warden Memory Analysis - MySQL Password Dump

Due to the nature of MySQL driver on Go language, username and password variables needs to be used over and over again. It is not possible to fully encrypt these variables in the memory. However, Halborn team strongly recommends that using the [memguard software](#) can help to store secure credentials in the memory.

### Risk Level:

Likelihood - 3

Impact - 1

### Remediation Plan:

**ACKNOWLEDGED:** The AVA Labs Team explains that the attack will only succeed if a threshold of wardens are compromised. To further limit the risk, each warden operator is required by a contractual agreement to follow best known security practices. Specifically, a warden operator must ensure that only a small number of trusted system administrators have access to their warden deployment. There are also incident response procedures in place for handling situations when a compromised warden is detected. The AVA Labs Team will consider other solutions that address this risk in future versions of the warden.



## 4.7 (HAL-07) LACK OF INPUT VALIDATION - INFORMATIONAL

### Description:

Due to lack of input validation on storing secret shares, the application runs an extra step on the database. According to source code, the application tries to control whether the secret share input is hexadecimal or not when storing that secret share in the memory. However, that control does not exist while the application stores any secret share on the database. As a result even if the value is not hexadecimal, the application tries to store invalid value on the database.

```
2021/06/10 14:31:44 Error saving secret share in memory. Error: encoding/hex: invalid byte: U+005A 'Z'
2021/06/10 14:31:52 Serving GetGasPrice request.
2021/06/10 14:31:52 Serving GetGasPrice request.
```

Figure 7: Lack of Input Validation - Memory

shareid	timestamp	share
1	2021-06-10 12:55:07	0-073100059F26F9581A6CCDFCC7ECC318C960019327942D63FA5AC27C757C7FC9FF8BFF4B5939256712117E195194CF41539244FB6C750427F5D84C198A9A9FB55D
2	2021-06-10 14:29:35	0-073100059F26F9581A6CCDFCC7ECC318C960019327942D63FA5AC27C757C7FC9FF8BFF4B5939256712117E195194CF41539244FB6C750427F5D84C198A9A9FB55D
3	2021-06-10 14:31:44	0-073100059F26F9581A6CCDFCC7ECC318C960019327942D63FA5AC27C757C7FC9FF8BFF4B5939256712117E195194CF41539244FB6C750427F5D84C198A9A9FB55D
4	2021-06-10 14:32:17	0-F73100059F26F9581A6CCDFCC7ECC318C960019327942D63FA5AC27C757C7FC9FF8BFF4B5939256712117E195194CF41539244FB6C750427F5D84C198A9A9FB55D

Figure 8: Lack of Input Validation - DB

Additionally, the `NewDatabase` function in `domain/database/database.go` expects three user-supplied parameters: `username`, `password` and DB's IP address in order to create the connection string. The function however neither encodes nor sanitizes those parameters so e.g. in case username or password contain non-alphanumeric characters the connection string will be malformed.

### Risk Level:

Likelihood - 2

Impact - 1

Code Location:

Listing 8: handlers/handler.go (Lines 77,99,107)

```

66 func (h *Handler) SetSecretShare(w http.ResponseWriter, request *
    http.Request) {
67     log.Println("Serving SetSecretShare request.")
68     decoder := json.NewDecoder(request.Body)
69     defer request.Body.Close()
70
71     var sss contracts.SetSecretShareRequest
72     var err error
73     if configuration.GlobalConfiguration.AllowUnencryptedSecrets {
74         err = decoder.Decode(&sss)
75     } else {
76         var er contracts.EncryptedRequest
77         err := decoder.Decode(&er)
78         if err != nil || len(er.Body) == 0 {
79             log.Println("Malformed SetSecretShare encrypted
                request.")
80             sendResponse(w, false, contracts.ErrorResponse{
                ErrorMessage: "Malformed request"})
81             return
82         }
83         decryptedBody, err := secrets.DecryptRequest(er)
84         if err != nil || len(decryptedBody) == 0 {
85             log.Println("Error decrypting secret share request
                body.")
86             sendResponse(w, false, contracts.ErrorResponse{
                ErrorMessage: "Error decrypting body."})
87             return
88         }
89         err = json.Unmarshal(decryptedBody, &sss)
90     }
91
92     if err != nil || len(sss.SecretShare) == 0 {
93         log.Println("Error unmarshaling secret share.")
94         sendResponse(w, false, contracts.ErrorResponse{
            ErrorMessage: "Error unmarshaling secret share."})
95         return
96     }
97
98     // Store the new secret share in the database. Old secret
        share values are still kept but no longer used.
99     err = h.db.SetSecretShare(sss.SecretShare)

```

```

100
101     if err != nil {
102         log.Println("Error saving secret share in database. Error: ", err.Error())
103         sendResponse(w, false, contracts.ErrorResponse{
104             ErrorMessage: err.Error()})
105         return
106     }
107     err = secrets.SetWardenSecretShare(sss.SecretShare)
108
109     if err != nil {
110         log.Println("Error saving secret share in memory. Error:", err.Error())
111         sendResponse(w, false, contracts.ErrorResponse{
112             ErrorMessage: err.Error()})
113         return
114     }
115     sendResponse(w, true, contracts.WardenMessage{Message: "success"})
116 }

```

Listing 9: domain/database/database.go (Lines 42)

```

40 func NewDatabase(mysqlUser, mysqlPassword, mysqlIP string) (*
    WardenDatabase, error) {
41     // Open the database
42     var dbConnectionString = mysqlUser + ":" + mysqlPassword + "
        @tcp(" + mysqlIP + ":3306)/Warden?parseTime=true"
43     db, err := sql.Open("mysql", dbConnectionString)
44     if err != nil {
45         return nil, err
46     }

```

#### Recommendation:

Always validate user-supplied parameters to match the expected format and use relevant encoding when data is transferred.

Apply input validation to `h.db.SetSecretShare(sss.SecretShare)` before

sending invalid secret share value to the database.

Reference: [Connection Strings in Golang](#)

Remediation Plan:

**RISK ACCEPTED:** AVA Labs Team accepts the risk.

## 4.8 (HAL-08) MISSING HTTP SECURITY HEADERS ON WARDEN REQUESTS - INFORMATIONAL

### Description:

Halborn noticed HTTP security response headers are missing on the Warden HTTP Responses.

- `X-Content-Type-Options`, which indicates that the MIME types advertised in the Content-Type headers should not be changed and be followed.
- `X-Frame-Options`, which indicates whether or not a browser should be allowed to render a page in a `<frame>`, `<iframe>`, `<embed>` or `<object>`.
- `Content-Security-Policy`, which allows web site administrators to control resources the user agent is allowed to load for a given page.
- `Strict-Transport-Security (HSTS)` - allows to force the user's client to make all requests and connections take place only through the encrypted HTTPS communication channel.
- `Referrer-Policy` - specifies what information, or parts of it, should be sent in the Referer header with the request that prompted the redirection.
- `Cache-Control` - instructs the browser: if, how and which items should be stored in the temporary memory.
- `Pragma` - using the "no-cache" directive forces the browser to query the server before downloading a cached copy of the page, resulting in the download of the most recent version.
- `Expires` - includes a date, period, or value indicating when the server's response is no longer correct.

## Code Location:

Listing 10: handlers/handler.go (Lines 33)

```
32 func sendResponse(w http.ResponseWriter, success bool, v interface
    {})) {
33     w.Header().Set("Content-Type", "application/json")
34     result, err := json.Marshal(v)
35
36     if err != nil {
37         panic(err)
38     }
39
40     status := "success"
41     if !success {
42         status = "error"
43     }
44
45     response := contracts.WardenResponse{
46         Status: status,
47         Result: result,
48     }
49
50     respBytes, err := json.Marshal(response)
51
52     if err != nil {
53         panic(err)
54     }
55
56     w.Write(respBytes)
57
58     return
59 }
```

## Recommendation:

It is recommended to implement the HTTP headers.

#### Remediation Plan:

**NOT APPLICABLE:** The AVA Labs Team essentially uses HTTP as an RPC protocol between Warden and Bridge. Browser-specific HTTP headers are not required, as there will be no interaction with the browser while both of the applications are running.

## 4.9 (HAL-09) VERBOSE ERROR MESSAGES – INFORMATIONAL

### Description:

Improper handling of errors can introduce a variety of security problems for software. The most common problem is when detailed internal error messages such as stack traces, database dumps, and error codes are displayed to the user (hacker). These messages reveal implementation details that should never be revealed. Such details can provide hackers important clues on potential flaws in the site and such messages are also disturbing to normal users.

### Risk Level:

**Likelihood - 1**

**Impact - 1**

### Code Location:

Listing 11: domain/secrets/secrets.go (Lines 66,80,88)

```
64 func DecryptRequest(req contracts.EncryptedRequest) ([]byte, error) {
65     if len(bridgeRASharedSecret) == 0 {
66         return nil, errors.New("No shared secret from RA. Can't
           perform decryption.")
67     }
68
69     ciphertext, err := hex.DecodeString(req.Body)
70     if err != nil {
71         return nil, err
72     }
73
74     block, err := aes.NewCipher(bridgeRASharedSecret)
75     if err != nil {
76         return nil, err
77     }
```



```
78
79     if len(ciphertext) < aes.BlockSize {
80         return nil, errors.New("Ciphertext too short")
81     }
82
83     iv := ciphertext[:aes.BlockSize]
84     ciphertext = ciphertext[aes.BlockSize:]
85
86     // CBC mode always works in whole blocks.
87     if len(ciphertext)%aes.BlockSize != 0 {
88         return nil, errors.New("Ciphertext is not a multiple of
            the block size")
89     }
90
```

#### Recommendation:

Make all cryptography-related error messages more generic. Another way to fix this issue could be implementing the **error code - error message** mapping.

#### Remediation Plan:

**SOLVED:** The **AVA Labs Team** has addressed this issue by reducing the amount of information in the error messages.

## 4.10 (HAL-10) SHADOWED VARIABLES - INFORMATIONAL

### Description:

In computer programming, variable shadowing occurs when a variable declared within a certain scope has the same name as a variable declared in an outer scope. Shadowing can lead to unintended consequences, such as accessing one variable when the intention was to have accessed another. For example, when a variable “x” in the inner scope shadows another variable “y” in the outer scope you might be thinking that you are accessing and modifying the value referenced by “y”, while you are actually accessing and only modifying the value reference by “x”. This can lead to confusion, as it may be unclear which variable subsequent uses of the shadowed variable name refer to, which depends on the name resolution rules of the language.

```
# command-line-arguments
shadow: domain/utils/utils_test.go:9:6: undeclared name: CheckFileExists
# command-line-arguments
shadow: domain/indexing/index_manager.go:9:23: undeclared name: TransactionIndexer
# command-line-arguments
shadow: domain/configuration/secrets_manager.go:29:70: undeclared name: GlobalConfiguration
# command-line-arguments
shadow: domain/networks/networks.go:50:18: undeclared name: FormatEvmGetNonceRequest
# command-line-arguments
domain/settings/settings_manager.go:186:13: declaration of "err" shadows declaration at line 173
# command-line-arguments
domain/database/database.go:306:3: declaration of "err" shadows declaration at line 289
domain/database/database.go:419:3: declaration of "err" shadows declaration at line 398
domain/database/database.go:484:3: declaration of "err" shadows declaration at line 463
domain/database/database.go:539:3: declaration of "err" shadows declaration at line 528
domain/database/database.go:631:3: declaration of "err" shadows declaration at line 618
domain/database/database.go:664:3: declaration of "err" shadows declaration at line 653
# command-line-arguments
handlers/handler.go:77:3: declaration of "err" shadows declaration at line 72
```

Figure 9: Shadowed Variables

### Risk Level:

Likelihood - 1

Impact - 1

#### Code Location:

domain/settings/settings\_manager.go - Lines #173,186  
domain/database/database.go - Lines #289,306  
domain/database/database.go - Lines #398,416  
domain/database/database.go - Lines #463,484  
domain/database/database.go - Lines #528,539  
domain/database/database.go - Lines #618,631  
domain/database/database.go - Lines #653,664  
handlers/handler.go - Lines #72,77

#### Recommendation:

The variable shadowing could be avoided by simply renaming variables with unambiguous names. Re-defining or re-initializing variables also can solve the problem. As the long term plan, run Go's "shadow" tool over the source code.

#### Remediation Plan:

**ACKNOWLEDGED:** This informational risk was acknowledged and AVA Labs Team has decided to not take action on this issue.

## 4.11 (HAL-11) MISSING GO COMPILER BUILD DIRECTIVES - INFORMATIONAL

### Description:

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

### Risk Level:

**Likelihood - 1**

**Impact - 1**

### Repository Makefile Flags:

#### Listing 12: Makefile (Lines )

```
1      @go build -o warden -ldflags "-X main.gitCommitHash=$(  
      GIT_COMMIT)" .  
2
```

### Example Flags:

Listing 13: Example Compiler Build Flags (Lines )

```
1 -a
2     force rebuilding of packages that are already up-to-date.
3 -ldflags "-s -w"
4     The -w turns off DWARF debugging information
5     The -s turns off generation of the Go symbol table
6 -trimpath
7     The -trimpath Remove all file system paths from the resulting
      executable.
8 -gcflags
9     arguments to pass on each go tool compile invocation.
```

### Recommendation:

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

### Remediation Plan:

**NOT APPLICABLE:** AVA Labs Team considers that this issue do not possess any security risk.

## 4.12 (HAL-12) INEFFECTUAL VARIABLE ASSIGNMENT - INFORMATIONAL

### Description:

During the tests, it has been observed that the `err` variable assignment is ineffectual when the multiple errors exist.

### Risk Level:

Likelihood - 1

Impact - 1

## Code Location:

`/domain/networks/networks.go` - Lines #256,294`/handlers/handler.go` - Lines #85,135

## Example Code:

Listing 14: `/domain/networks/networks.go`

```
85 .....
86         err = json.Unmarshal(decryptedBody, &sss)
87     }
88
89     if err != nil || len(sss.SecretShare) == 0 {
90         log.Println("Error unmarshaling secret share.")
91         sendResponse(w, false, contracts.ErrorResponse{
92             ErrorMessage: "Error unmarshaling secret share."})
93         return
94     }
95
96     // Store the new secret share in the database. Old secret
97     // share values are still kept but no longer used.
98     err = h.db.SetSecretShare(sss.SecretShare)
99
100    if err != nil {
101        log.Println("Error saving secret share in database. Error: ", err.Error())
102        sendResponse(w, false, contracts.ErrorResponse{
103            ErrorMessage: err.Error()})
104        return
105    }
106
107    err = secrets.SetWardenSecretShare(sss.SecretShare)
108
109    if err != nil {
110        log.Println("Error saving secret share in memory. Error:", err.Error())
111        sendResponse(w, false, contracts.ErrorResponse{
112            ErrorMessage: err.Error()})
113        return
114    }
115    }
```

**Recommendation:**

The application error handling mechanisms should be designed according to error code types. The error codes have an implied value in the way that they both clarify the situation.

**Remediation Plan:**

**ACKNOWLEDGED:** This informational risk was acknowledged and AVA Labs Team has decided to not take action on this issue.





# STATIC ANALYSIS REPORT



## 5.1 STATIC ANALYSIS

Halborn used automated testing techniques to enhance coverage of certain areas of the scoped test contracts such as `TestERC20`, `WrappedERC` and etc. Among the tools used were `staticcheck`, `gosec` `ineffassign` and others. After Halborn verified all the contracts and scoped structures in the repository and was able to compile them correctly, these tools were leveraged on scoped structures. With these tools, Halborn can statically verify security related issues across the entire codebase.

### Test Coverage:

```
? github.com/ava-labs/bridge-warden [no test files]
? github.com/ava-labs/bridge-warden/contracts [no test files]
? github.com/ava-labs/bridge-warden/domain/clients [no test files]
? github.com/ava-labs/bridge-warden/domain/configuration [no test files]
? github.com/ava-labs/bridge-warden/domain/database [no test files]
? github.com/ava-labs/bridge-warden/domain/indexing [no test files]
? github.com/ava-labs/bridge-warden/domain/interfaces [no test files]
? github.com/ava-labs/bridge-warden/domain/networks [no test files]
? github.com/ava-labs/bridge-warden/domain/secrets [no test files]
ok github.com/ava-labs/bridge-warden/domain/utls 0.004s coverage: 60.0% of statements
? github.com/ava-labs/bridge-warden/handlers [no test files]
```

```
github.com/ava-labs/bridge-warden/domain/utls/utls.go:16: CheckFileExists 100.0%
github.com/ava-labs/bridge-warden/domain/utls/utls.go:24: HexToDecimal 80.0%
github.com/ava-labs/bridge-warden/domain/utls/utls.go:33: HexToBigDecimal 0.0%
github.com/ava-labs/bridge-warden/domain/utls/utls.go:46: DecimalToHex 100.0%
github.com/ava-labs/bridge-warden/domain/utls/utls.go:50: HexStringToDecimalString 77.8%
github.com/ava-labs/bridge-warden/domain/utls/utls.go:64: CalculateEvmTransactionId 88.9%
github.com/ava-labs/bridge-warden/domain/utls/utls.go:80: IsHexString 0.0%
total: (statements) 60.0%
```

## Gosec - Security Analysis Output Sample:

```
Results:

[/home/ziion/projects/avalanche/AVA_LABS/bridge-warden/domain/settings/settings_manager.go:232] - G306 (CWE-276): Expect WriteFile permissions to be 0600 or less (Confidence: HIGH, Severity: MEDIUM)Local:
233:         default:

[/home/ziion/projects/avalanche/AVA_LABS/bridge-warden/handlers/handler.go:60] - G104 (CWE-703): Errors unchecked. (Confidence: HIGH, Severity: LOW)
59:
> 60:         w.Write(respBytes)
61:

[/home/ziion/projects/avalanche/AVA_LABS/bridge-warden/domain/utills/utills.go:74] - G104 (CWE-703): Errors unchecked. (Confidence: HIGH, Severity: LOW)
73:         keccak := sha3.NewLegacyKeccak256()
> 74:         keccak.Write(transactionBytes)
75:         resultBytes := keccak.Sum(nil)

[/home/ziion/projects/avalanche/AVA_LABS/bridge-warden/domain/settings/settings_manager.go:97] - G104 (CWE-703): Errors unchecked. (Confidence: HIGH, Severity: LOW)
97:         if settingsExist {
> 97:             res.getAndStoreCombinedSettings(settingsBytes)
98:             go res.updateRoutine()

[/home/ziion/projects/avalanche/AVA_LABS/bridge-warden/domain/secrets/secrets.go:58] - G104 (CWE-703): Errors unchecked. (Confidence: HIGH, Severity: LOW)
57:         mac := hmac.New(sha256.New, bridgeSecretShareHMAC)
> 58:         mac.Write(req.Request)
59:         expectedMAC := mac.Sum(nil)

[/home/ziion/projects/avalanche/AVA_LABS/bridge-warden/domain/database/database.go:226] - G104 (CWE-703): Errors unchecked. (Confidence: HIGH, Severity: LOW)
225:         db.getLatestBridgeSettingsStmt.Close()
> 226:         db.db.Close()
227:     }

[/home/ziion/projects/avalanche/AVA_LABS/bridge-warden/domain/database/database.go:225] - G104 (CWE-703): Errors unchecked. (Confidence: HIGH, Severity: LOW)
224:         db.insertBridgeSettingsStmt.Close()
> 225:         db.getLatestBridgeSettingsStmt.Close()
226:         db.db.Close()

[/home/ziion/projects/avalanche/AVA_LABS/bridge-warden/domain/database/database.go:224] - G104 (CWE-703): Errors unchecked. (Confidence: HIGH, Severity: LOW)
223:         db.completeFeeTransactionStmt.Close()
> 224:         db.insertBridgeSettingsStmt.Close()
225:         db.getLatestBridgeSettingsStmt.Close()

[/home/ziion/projects/avalanche/AVA_LABS/bridge-warden/domain/database/database.go:223] - G104 (CWE-703): Errors unchecked. (Confidence: HIGH, Severity: LOW)
222:         db.prepareFeeTransactionStmt.Close()
> 223:         db.completeFeeTransactionStmt.Close()
224:         db.insertBridgeSettingsStmt.Close()

[/home/ziion/projects/avalanche/AVA_LABS/bridge-warden/domain/database/database.go:222] - G104 (CWE-703): Errors unchecked. (Confidence: HIGH, Severity: LOW)
221:         db.getPreparedFeeTransactionBlobsStmt.Close()
> 222:         db.prepareFeeTransactionStmt.Close()
223:         db.completeFeeTransactionStmt.Close()

[/home/ziion/projects/avalanche/AVA_LABS/bridge-warden/domain/database/database.go:221] - G104 (CWE-703): Errors unchecked. (Confidence: HIGH, Severity: LOW)
220:         db.getOperatorFeeAmountsStmt.Close()
> 221:         db.getPreparedFeeTransactionBlobsStmt.Close()
222:         db.prepareFeeTransactionStmt.Close()
```

```
[/home/ziion/projects/avalanche/AVA_LABS/bridge-warden/domain/database/database.go:220] - G104 (CWE-703): Errors unhandled. (Confidence: HIGH, Severity:
LOW)
219:         db.insertAssetPairStmt.Close()
> 220:         db.getOperatorFeeAmountsStmt.Close()
221:         db.getPreparedFeeTransactionBlobsStmt.Close()

[/home/ziion/projects/avalanche/AVA_LABS/bridge-warden/domain/database/database.go:219] - G104 (CWE-703): Errors unhandled. (Confidence: HIGH, Severity:
LOW)
218:         db.getAssetPairByWrappedStmt.Close()
> 219:         db.insertAssetPairStmt.Close()
220:         db.getOperatorFeeAmountsStmt.Close()

[/home/ziion/projects/avalanche/AVA_LABS/bridge-warden/domain/database/database.go:218] - G104 (CWE-703): Errors unhandled. (Confidence: HIGH, Severity:
LOW)
217:         db.getAssetPairByNativeStmt.Close()
> 218:         db.getAssetPairByWrappedStmt.Close()
219:         db.insertAssetPairStmt.Close()

[/home/ziion/projects/avalanche/AVA_LABS/bridge-warden/domain/database/database.go:217] - G104 (CWE-703): Errors unhandled. (Confidence: HIGH, Severity:
LOW)
216:         db.getAllAssetPairsStmt.Close()
> 217:         db.getAssetPairByNativeStmt.Close()
218:         db.getAssetPairByWrappedStmt.Close()

[/home/ziion/projects/avalanche/AVA_LABS/bridge-warden/domain/database/database.go:216] - G104 (CWE-703): Errors unhandled. (Confidence: HIGH, Severity:
LOW)
215:         db.getMaxIndexedBlockStmt.Close()
> 216:         db.getAllAssetPairsStmt.Close()
217:         db.getAssetPairByNativeStmt.Close()

[/home/ziion/projects/avalanche/AVA_LABS/bridge-warden/domain/database/database.go:216] - G104 (CWE-703): Errors unhandled. (Confidence: HIGH, Severity:
LOW)15:         db.getMaxIndexedBlockStmt.Close()
LOW)
214:         db.getUnprocessedUnwrapsStmt.Close()
> 215:         db.getMaxIndexedBlockStmt.Close()
216:         db.getAllAssetPairsStmt.Close()

[/home/ziion/projects/avalanche/AVA_LABS/bridge-warden/domain/database/database.go:214] - G104 (CWE-703): Errors unhandled. (Confidence: HIGH, Severity:
LOW)
213:         db.getUnprocessedTransfersStmt.Close()
> 214:         db.getUnprocessedUnwrapsStmt.Close()
215:         db.getMaxIndexedBlockStmt.Close()

[/home/ziion/projects/avalanche/AVA_LABS/bridge-warden/domain/database/database.go:213] - G104 (CWE-703): Errors unhandled. (Confidence: HIGH, Severity:
LOW)
212:         db.getPreparedRequestBlobsStmt.Close()
> 213:         db.getUnprocessedTransfersStmt.Close()
214:         db.getUnprocessedUnwrapsStmt.Close()

[/home/ziion/projects/avalanche/AVA_LABS/bridge-warden/domain/database/database.go:212] - G104 (CWE-703): Errors unhandled. (Confidence: HIGH, Severity:
LOW)
211:         db.completeRequestStmt.Close()
> 212:         db.getPreparedRequestBlobsStmt.Close()
213:         db.getUnprocessedTransfersStmt.Close()

[/home/ziion/projects/avalanche/AVA_LABS/bridge-warden/domain/database/database.go:211] - G104 (CWE-703): Errors unhandled. (Confidence: HIGH, Severity:
LOW)
210:         db.getRequestStmt.Close()
> 211:         db.completeRequestStmt.Close()
212:         db.getPreparedRequestBlobsStmt.Close()

[/home/ziion/projects/avalanche/AVA_LABS/bridge-warden/domain/database/database.go:211] - G104 (CWE-703): Errors unhandled. (Confidence: HIGH, Severity:
LOW)10:         db.getRequestStmt.Close()
> 211:         db.completeRequestStmt.Close()
LOW)
209:         db.prepareRequestStmt.Close()
> 210:         db.getRequestStmt.Close()
211:         db.completeRequestStmt.Close()

[/home/ziion/projects/avalanche/AVA_LABS/bridge-warden/domain/database/database.go:209] - G104 (CWE-703): Errors unhandled. (Confidence: HIGH, Severity:
LOW)
208:         db.insertTxStmt.Close()
> 209:         db.prepareRequestStmt.Close()
210:         db.getRequestStmt.Close()

[/home/ziion/projects/avalanche/AVA_LABS/bridge-warden/domain/database/database.go:208] - G104 (CWE-703): Errors unhandled. (Confidence: HIGH, Severity:
LOW)
207:         db.insertSecretShareStmt.Close()
> 208:         db.insertTxStmt.Close()
209:         db.prepareRequestStmt.Close()

[/home/ziion/projects/avalanche/AVA_LABS/bridge-warden/domain/database/database.go:207] - G104 (CWE-703): Errors unhandled. (Confidence: HIGH, Severity:
LOW)
206:         db.getCurrentSecretShareStmt.Close()
> 207:         db.insertSecretShareStmt.Close()
208:         db.insertTxStmt.Close()

[/home/ziion/projects/avalanche/AVA_LABS/bridge-warden/domain/database/database.go:206] - G104 (CWE-703): Errors unhandled. (Confidence: HIGH, Severity:
LOW)
205: func (db *WardenDatabase) CloseDatabase() {
> 206:     db.getCurrentSecretShareStmt.Close()
207:     db.insertSecretShareStmt.Close()
}
```



## Unconvert - Security Analysis Output Sample:

```
→ bridge-warden-main unconvert -v ./...
/home/zilon/Desktop/AVA_LABS2/AVA_LABS/bridge-warden-main/domain/secrets/secrets.go:125:21: unnecessary conversion
    plaintext := string(go_plaintext)
                        ^
→ bridge-warden-main █
```

According to the test results, some of the findings found by these tools were considered as false positives while some of these findings were real security concerns. All relevant findings were reviewed by the auditors and relevant findings addressed on the report as security concerns.



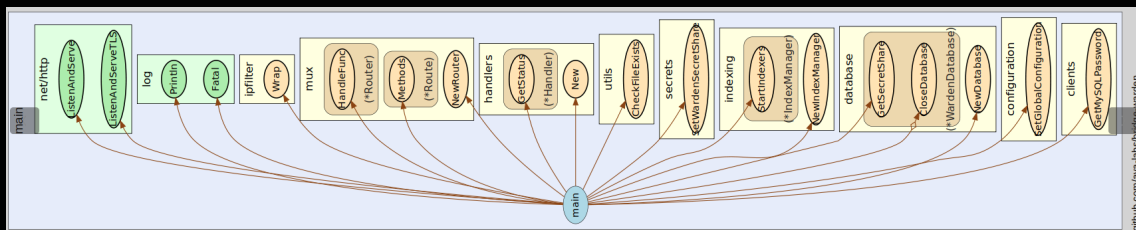
# FUZZING REPORT



## 6.1 FUZZING REPORT

Fuzzing is a type of automated testing which continuously manipulates inputs to a program to find issues such as panics or bugs. These semi-random data mutations can discover new code coverage that existing unit tests may miss, and uncover edge case bugs which would otherwise go unnoticed. Since fuzzing can reach these edge cases, fuzz testing is particularly valuable for finding security exploits and vulnerabilities. The Halborn team worked in many fuzzing test cases.

## Warden Components Covered by Fuzzing



## 6.2 PROPERTY BASED FUZZING

During the fuzzing tests, the Halborn team used the property based fuzzing. Basically, all important functions were tested against crashes. To achieve this, multiple specific test cases were written for multiple `.go` files. Nearly all functions have passed the fuzzing tests. However, one important function crashed during the fuzzing; `CheckFileExists`.

### Example Test Case:

### Listing 15



```

6     return 1
7 }

```

### Fuzzing Progress:

```

2021/06/17 07:47:12 workers: 2, corpus: 10 (8m5s ago), crashers: 1, restarts: 1/5, execs: 358795 (448/sec), cover: 49, uptime: 13m21s
2021/06/17 07:47:15 workers: 2, corpus: 10 (8m8s ago), crashers: 1, restarts: 1/5, execs: 360098 (448/sec), cover: 49, uptime: 13m24s
2021/06/17 07:47:18 workers: 2, corpus: 10 (8m11s ago), crashers: 1, restarts: 1/5, execs: 361557 (448/sec), cover: 49, uptime: 13m27s
2021/06/17 07:47:21 workers: 2, corpus: 10 (8m14s ago), crashers: 1, restarts: 1/5, execs: 363094 (448/sec), cover: 49, uptime: 13m30s
2021/06/17 07:47:24 workers: 2, corpus: 10 (8m17s ago), crashers: 1, restarts: 1/5, execs: 364667 (449/sec), cover: 49, uptime: 13m33s
2021/06/17 07:47:27 workers: 2, corpus: 10 (8m20s ago), crashers: 1, restarts: 1/5, execs: 366057 (449/sec), cover: 49, uptime: 13m36s
2021/06/17 07:47:30 workers: 2, corpus: 10 (8m23s ago), crashers: 1, restarts: 1/5, execs: 367057 (448/sec), cover: 49, uptime: 13m39s
2021/06/17 07:47:33 workers: 2, corpus: 10 (8m26s ago), crashers: 1, restarts: 1/5, execs: 368558 (448/sec), cover: 49, uptime: 13m42s
2021/06/17 07:47:36 workers: 2, corpus: 10 (8m29s ago), crashers: 1, restarts: 1/5, execs: 370127 (449/sec), cover: 49, uptime: 13m45s
2021/06/17 07:47:39 workers: 2, corpus: 10 (8m32s ago), crashers: 1, restarts: 1/5, execs: 371075 (448/sec), cover: 49, uptime: 13m48s
2021/06/17 07:47:42 workers: 2, corpus: 10 (8m35s ago), crashers: 1, restarts: 1/5, execs: 371765 (447/sec), cover: 49, uptime: 13m51s
2021/06/17 07:47:45 workers: 2, corpus: 10 (8m38s ago), crashers: 1, restarts: 1/5, execs: 373173 (447/sec), cover: 49, uptime: 13m54s
2021/06/17 07:47:48 workers: 2, corpus: 10 (8m41s ago), crashers: 1, restarts: 1/5, execs: 374850 (448/sec), cover: 49, uptime: 13m57s
2021/06/17 07:47:51 workers: 2, corpus: 10 (8m44s ago), crashers: 1, restarts: 1/5, execs: 376200 (448/sec), cover: 49, uptime: 14m0s
2021/06/17 07:47:54 workers: 2, corpus: 10 (8m47s ago), crashers: 1, restarts: 1/5, execs: 377652 (448/sec), cover: 49, uptime: 14m3s
2021/06/17 07:47:57 workers: 2, corpus: 10 (8m50s ago), crashers: 1, restarts: 1/5, execs: 379146 (448/sec), cover: 49, uptime: 14m6s
2021/06/17 07:48:00 workers: 2, corpus: 10 (8m53s ago), crashers: 1, restarts: 1/5, execs: 380723 (448/sec), cover: 49, uptime: 14m9s
2021/06/17 07:48:03 workers: 2, corpus: 10 (8m56s ago), crashers: 1, restarts: 1/5, execs: 382232 (449/sec), cover: 49, uptime: 14m12s
2021/06/17 07:48:06 workers: 2, corpus: 10 (8m59s ago), crashers: 1, restarts: 1/5, execs: 383608 (449/sec), cover: 49, uptime: 14m15s
2021/06/17 07:48:09 workers: 2, corpus: 10 (9m2s ago), crashers: 1, restarts: 1/5, execs: 384985 (449/sec), cover: 49, uptime: 14m18s
2021/06/17 07:48:12 workers: 2, corpus: 10 (9m5s ago), crashers: 1, restarts: 1/5, execs: 386381 (449/sec), cover: 49, uptime: 14m21s
2021/06/17 07:48:15 workers: 2, corpus: 10 (9m8s ago), crashers: 1, restarts: 1/5, execs: 387782 (449/sec), cover: 49, uptime: 14m24s
2021/06/17 07:48:18 workers: 2, corpus: 10 (9m11s ago), crashers: 1, restarts: 1/5, execs: 389208 (449/sec), cover: 49, uptime: 14m27s
2021/06/17 07:48:21 workers: 2, corpus: 10 (9m14s ago), crashers: 1, restarts: 1/5, execs: 390650 (449/sec), cover: 49, uptime: 14m30s
2021/06/17 07:48:24 workers: 2, corpus: 10 (9m17s ago), crashers: 1, restarts: 1/5, execs: 392108 (449/sec), cover: 49, uptime: 14m33s
2021/06/17 07:48:27 workers: 2, corpus: 10 (9m20s ago), crashers: 1, restarts: 1/5, execs: 393578 (449/sec), cover: 49, uptime: 14m36s
2021/06/17 07:48:30 workers: 2, corpus: 10 (9m23s ago), crashers: 1, restarts: 1/5, execs: 394903 (449/sec), cover: 49, uptime: 14m39s
2021/06/17 07:48:33 workers: 2, corpus: 10 (9m26s ago), crashers: 1, restarts: 1/5, execs: 396349 (449/sec), cover: 49, uptime: 14m42s
2021/06/17 07:48:36 workers: 2, corpus: 10 (9m29s ago), crashers: 1, restarts: 1/5, execs: 397779 (449/sec), cover: 49, uptime: 14m45s
2021/06/17 07:48:39 workers: 2, corpus: 10 (9m32s ago), crashers: 1, restarts: 1/5, execs: 399200 (450/sec), cover: 49, uptime: 14m48s
2021/06/17 07:48:42 workers: 2, corpus: 10 (9m35s ago), crashers: 1, restarts: 1/5, execs: 400232 (449/sec), cover: 49, uptime: 14m51s
2021/06/17 07:48:45 workers: 2, corpus: 10 (9m38s ago), crashers: 1, restarts: 1/5, execs: 401271 (449/sec), cover: 49, uptime: 14m54s
2021/06/17 07:48:48 workers: 2, corpus: 10 (9m41s ago), crashers: 1, restarts: 1/5, execs: 401900 (448/sec), cover: 49, uptime: 14m57s
2021/06/17 07:48:51 workers: 2, corpus: 10 (9m44s ago), crashers: 1, restarts: 1/5, execs: 402347 (447/sec), cover: 49, uptime: 15m0s
2021/06/17 07:48:54 workers: 2, corpus: 10 (9m47s ago), crashers: 1, restarts: 1/5, execs: 402548 (446/sec), cover: 49, uptime: 15m3s
→ crashers cat 5ba93c9db0cff93f52b521d7420e43f6eda2784f.quoted
"\x00"
→ crashers cat 5ba93c9db0cff93f52b521d7420e43f6eda2784f.output
panic: runtime error: invalid memory address or nil pointer dereference
[signal SIGSEGV: segmentation violation code=0x1 addr=0x18 pc=0x4a2690]

goroutine 1 [running]:
github.com/ava-labs/bridge-warden/domain/utls.CheckFileExists(0x54bda0, 0x1, 0x1)
/home/ziion/Desktop/AVA_LABS2/AVA_LABS/bridge-warden-main/domain/utls/utls.go:21 +0x110
github.com/ava-labs/bridge-warden/domain/utls.Fuzz(0x7f8cab2c8000, 0x1, 0x1, 0x4)
/home/ziion/Desktop/AVA_LABS2/AVA_LABS/bridge-warden-main/domain/utls/utls.go:25 +0x72
go-fuzz-dep.Main(0xc0000a9f70, 0x1, 0x1)
go-fuzz-dep/main.go:36 +0x1b8
main.main()
github.com/ava-labs/bridge-warden/domain/utls/go.fuzz.main/main.go:15 +0x52
exit status 2

```

Also, further research shows that it is possible to crash the Warden executable by providing broken Symlink as `/etc/warden/warden.conf` file. Based on these test cases, the Halborn team strongly recommends applying “Null byte” input checking and “Symlink” checking to the `CheckFileExists` function to avoid security vulnerabilities that may be introduced with this crash condition.



THANK YOU FOR CHOOSING

// HALBORN

