Non-Fungible Token Auction Demo: Cryptography and Implementation Review

Hedera Hashgraph

August 6, 2021 – Version 1.0

Prepared for

Greg Scullard

Alex Popowycz

Prepared by

Parnian Alimi

Eric Schorn

Executive Summary

Synopsis

During the summer of 2021, Hedera engaged NCC Group's Cryptography Services team to conduct an implementation and cryptography review of the NFT Auction Demo Application. This application provides two sets of REST APIs: the first allows any user to monitor the auctions underway, and the second allows administrators to manage these auctions. Functionality is included for auction and bid processing, registering new auctions, checking bid validity, and issuing refunds. Full source code for the Java backend deployed on Docker with Postgres was provided. Two consultants delivered this engagement with 15-person days of effort and with developer support over Slack. A retest was then performed to confirm fixed findings.

Scope

The primary scope of NCC Group's evaluation included commit `87873d1` and later `6c9cd8d` of https://github.com/hashgraph/hedera-nft-auction-demo. This code implements target functionality involving:

- REST API handling client UI's GET request and administrator POST requests: com.hedera.demo.auction.app.api.
- Functionality handling business and user scenarios: com.hedera.demo.auction.app.
- Docker and Postgres containers.

Limitations

While the target functionality is part of a much larger system context, good coverage was achieved over all in-scope material. Note that though the JavaScript UI was out of scope, one finding involving dependencies was noted as it complemented a similar in-scope (Java) finding.

Key Findings

NCC Group found the documentation and setup instructions to be thorough and helpful, while the demo application code is well structured, utilizes modern web frameworks and uses secure techniques such as prepared statements for SQL database queries. However, the engagement did uncover a number of issues, some rated high and medium severity, that will need to be addressed prior to production deployment, with the most significant including:

- Insecure network transport lacking support for confidentiality, integrity and authentication between the user and the API, as well as between the internal components of the application (e.g., database).
- State-changing operations are possible without a robust authentication check.
- Missing input validation at each API endpoint to prevent malicious input from causing undesirable downstream behavior.
- Hardcoded and default credentials stored in the version control system.
- Including mock object creation for testing as well as having a flag to enable/disable parts of the logic for testing, increases application's attack surface and can potentially lead to unexpected behavior as the application grows in complexity.
- Insufficient cleanup on failure can leave the auctions' database in corrupted state.

Many of the findings reported during the project mid-point status update were subsequently addressed in commit `e8f11a8`.

Strategic Recommendations

NCC Group recommends addressing the findings from this engagement and prioritizing several aspects of future development as follows:

- Secure coding practices as outlined in the OWASP Project's Top 10 Web Application Security Risks.
- Review the potential for secret and credential management through environment variables.
- Refactor the test code out of the main package and set the visibility of the sample/easySetup scripts to local.
- Ensure all dependencies are fully up to date with versions recommended for production deployment.

Additional implementation-related observations are offered in Appendix A on page 34.

Retest Result

A retest was performed in August 2021 over several new commits. All findings were fixed, with the exception of the one out-of-scope finding involving JavaScript dependencies which has been reported to the owning team. Brief retest comments are appended to each individual finding and indicate which specific commit was reviewed. The overall status is summarized in Table of Findings on page 4.

As discussed during the final read-out call, once development progresses to the next milestone, the project will benefit from a fresh review as the large number of higher-severity findings and various commits retested have the potential to obscure lower-severity flaws.

Dashboard

Target Metadata

Name	Non-Fungible Token Auction Demo
Type	Web/Blockchain Application
Platforms	Java, Postgres and Docker
Environment	Testing

Engagement Data

Type	Implementation and Cryptography Review
Method	Manual Source Code Analysis
Dates	2021-07-05 to 2021-07-16
Consultants	2
Level of Effort	15 person-days

Targets

GitHub Code Repository Commit `6c9cd`	https://github.com/hashgraph/hedera-nft-auction-demo/tree/6c9cd8d9af1252548abab5634749b19102b0a14b

Finding Breakdown

Critical issues	0	
High issues	6	🟥🟥🟥🟥🟥🟥
Medium issues	4	🟧🟧🟧🟧
Low issues	2	🟨🟨
Informational issues	4	⬜⬜⬜⬜
Total issues	**16**	

Category Breakdown

Auditing and Logging	2	🟧🟨
Authentication	2	🟥🟥
Configuration	2	🟥🟨
Cryptography	1	🟥
Data Exposure	2	🟥⬜
Data Validation	3	🟥🟧🟧
Error Reporting	1	⬜
Patching	2	⬜⬜
Session Management	1	🟧

Component Breakdown

Deployment Environment	2	🟥🟨
hedera-nft-auction-demo-java-node	13	🟥🟥🟥🟥🟥🟧🟧🟧🟧🟨⬜⬜⬜
hedera-nft-auction-demo-javascript-client	1	⬜

Key

■ Critical 🟥 High 🟧 Medium 🟨 Low ⬜ Informational

Table of Findings

For each finding, NCC Group uses a composite risk score that takes into account the severity of the risk, application's exposure and user population, technical difficulty of exploitation, and other factors. For an explanation of NCC Group's risk rating and finding categorization, see Appendix B on page 35.

Title	Status	ID	Risk
Insufficient Input Validation	Fixed	004	High
Missing Authentication	Fixed	005	High
Missing HTTPS Transport Security	Fixed	006	High
Default/Hardcoded Postgres Credentials	Fixed	008	High
Directory Traversal	Fixed	009	High
Incomplete Validity Check in AuctionKeyList API	Fixed	015	High
Missing Unicode Normalization Step	Fixed	003	Medium
Log Injection	Fixed	011	Medium
Missing Cleanup on Failure	Fixed	016	Medium
Missing State Validation before Transition	Fixed	017	Medium
Docker Containerized (Guest) Applications Execute As Root	Fixed	001	Low
Logging of Private Key	Fixed	010	Low
Outdated Java Dependencies	Fixed	002	Informational
Missing Null Check in `PostAuctionHandler`	Fixed	007	Informational
Mock Object Creation Should Not Have Public Visibility	Fixed	013	Informational
Using Vulnerable Node Dependencies	Reported	014	Informational

Finding	**Insufficient Input Validation**
Risk	**High** Impact: High, Exploitability: Medium
Identifier	NCC-E001942-004
Status	Fixed
Category	Data Validation
Component	hedera-nft-auction-demo-java-node

Location Systemic, including:

- hedera-nft-auction-demo/hedera-nft-auction-demo-java-node/src/main/java/com/hedera/demo/auction/app/api/GetAuctionHandler.java
- hedera-nft-auction-demo/hedera-nft-auction-demo-java-node/src/main/java/com/hedera/demo/auction/app/api/GetLastBidderBidHandler.java
- hedera-nft-auction-demo/hedera-nft-auction-demo-java-node/src/main/java/com/hedera/demo/auction/app/api/GetBidsHandler.java
- hedera-nft-auction-demo/hedera-nft-auction-demo-java-node/src/main/java/com/hedera/demo/auction/app/api/PostEasySetupHandler.java
- hedera-nft-auction-demo/hedera-nft-auction-demo-java-node/src/main/java/com/hedera/demo/auction/app/api/PostCreateToken.java
- hedera-nft-auction-demo/hedera-nft-auction-demo-java-node/src/main/java/com/hedera/demo/auction/app/api/PostAuctionAccountHandler.java
- hedera-nft-auction-demo/hedera-nft-auction-demo-java-node/src/main/java/com/hedera/demo/auction/app/api/PostAuctionHandler.java
- hedera-nft-auction-demo/hedera-nft-auction-demo-java-node/src/main/java/com/hedera/demo/auction/app/api/PostTransferHandler.java
- hedera-nft-auction-demo/hedera-nft-auction-demo-java-node/src/main/java/com/hedera/demo/auction/app/api/PostValidators.java

Impact Insufficient input validation may expose undesired or undetermined behavior of downstream logic leading to vulnerabilities such as log injection, directory traversal, and denial of service.

Description External application input may originate from normal (and imperfect) users, maliciously modified "normal user clients", or constructed from scratch by malicious attackers and scripts, and has inherently crossed trust boundaries. Thus, all API endpoint input should be considered untrusted and aggressively validated before being stored or used. Additional client-side validation may improve user experience but final responsibility rests exclusively with the server.

Currently, the application performs minimal input validation. Examples from each of the locations noted above are excerpted below in the same order. Note that in some cases: integers are not validated against expected sign or magnitude, Strings are extracted verbatim without length or null checks, account formats are not inspected, and JSON is mapped directly into data objects. Any errors encountered are typically handled by a surrounding try/catch block for generic exceptions. The handling of duplicate or extraneous JSON fields is not obvious.

```java
long id = Long.parseLong(routingContext.pathParam("id"));
String bidderAccountId = routingContext.pathParam("bidderaccountid");
int auctionId = Integer.parseInt(routingContext.pathParam("auctionid"));
data = body.mapTo(RequestEasySetup.class);
TokenId tokenId = createToken.create(body.encode());
```

```
AccountId auctionAccount =
 ➜  createAuctionAccount.create(body.getLong("initialBalance"), keys.toString());
var data = body.mapTo(RequestCreateAuction.class);
var data = body.mapTo(RequestTokenTransfer.class);
args[3] = "--url=".concat(validatorJson.getString("url", ""));
```

Listing 1: Selected lines from each of the locations noted above

Best practices require that all input be validated as early as possible and as strictly as possible. The Vert.x framework provides a validation pattern and library[1] specifically for this purpose. Failure to sufficiently validate API input is the root cause of finding NCC-E001942-009 on page 13, finding NCC-E001942-011 on page 19 and finding NCC-E001942-003 on page 17.

Recommendation Develop an expected schema for all API endpoint input, and ensure all received parameters are validated as early as possible and as strictly as possible. For example:

- Duplicate or extraneous JSON fields should result in full rejection.
- Integers should have a minimum and maximum magnitude.
- Strings should have a minimum and maximum length. In many cases, the character set can be constrained to prevent vulnerabilities such as directory traversal. In many cases, Unicode normalization should be applied.
- Enumerated types should be explicitly checked for exactly one match.
- Objects having a specialized format, such as account IDs and URLs, should have their format validated.

Retest Results NCC Group reviewed commit `e8f11a8`, inspected each Location noted above, and observed the following:

- hedera-nft-auction-demo/hedera-nft-auction-demo-java-node/src/main/java/com/hedera/demo/auction/app/api/GetAuctionHandler.java – no relevant changes to this source file, but auction `id` is validated in ../ApiVerticle.java. Fixed.
- hedera-nft-auction-demo/hedera-nft-auction-demo-java-node/src/main/java/com/hedera/demo/auction/app/api/GetLastBidderBidHandler.java – no relevant changes to this source file, but `auctionid` and `bidderaccountid` are validated in ../ApiVerticle.java. Fixed.
- hedera-nft-auction-demo/hedera-nft-auction-demo-java-node/src/main/java/com/hedera/demo/auction/app/api/GetBidsHandler.java – no relevant changes to this source file, but `auctionid` is validated in ../ApiVerticle.java. Fixed.
- hedera-nft-auction-demo/hedera-nft-auction-demo-java-node/src/main/java/com/hedera/demo/auction/app/api/PostEasySetupHandler.java – now uses `io.vertx.json.schema.SchemaParser` and `validateSync()` on POST body JSON parameters. Fixed.
- hedera-nft-auction-demo/hedera-nft-auction-demo-java-node/src/main/java/com/hedera/demo/auction/app/api/PostCreateToken.java – now uses `io.vertx.json.schema.SchemaParser` and `validateSync()` on POST body JSON parameters. Fixed.
- hedera-nft-auction-demo/hedera-nft-auction-demo-java-node/src/main/java/com/hedera/demo/auction/app/api/PostAuctionAccountHandler.java – now uses `io.vertx.json.schema.SchemaParser` and `validateSync()` on POST body JSON parameters. Fixed.
- hedera-nft-auction-demo/hedera-nft-auction-demo-java-node/src/main/java/com/hedera/demo/auction/app/api/PostAuctionHandler.java – now uses `io.vertx.json.schema.SchemaParser` and `validateSync()` on POST body JSON parameters. Fixed.
- hedera-nft-auction-demo/hedera-nft-auction-demo-java-node/src/main/java/com/hedera/demo/auction/app/api/PostTransferHandler.java – now uses `io.vertx.json.schema.Sch`

[1] https://vertx.io/docs/vertx-web-validation/java/

`emaParser` and `validateSync()` on POST body JSON parameters. Fixed.

- hedera-nft-auction-demo/hedera-nft-auction-demo-java-node/src/main/java/com/hedera/demo/auction/app/api/PostValidators.java – now uses `io.vertx.json.schema.SchemaParser` and `validateSync()` on POST body JSON parameters. Fixed.

All of the noted POST and GET request parameters are now validated against expectations. As such, this finding has been marked ‘Fixed’.

Finding	**Missing Authentication**
Risk	**High** Impact: High, Exploitability: Low
Identifier	NCC-E001942-005
Status	Fixed
Category	Authentication
Component	hedera-nft-auction-demo-java-node
Location	Documentation at hedera-nft-auction-demo/docs/SolutionSpecification.md#admin-api
Impact	An attacker able to reach the admin API will be able to maliciously operate the application unimpeded.

Description

The application does not currently implement any form of authentication on the administrative interface. The documentation states:

> Note: The admin API isn't meant to be exposed to the public, there is no authorization or challenge on the API. It is meant to be invoked from the command line with a curl type command against a production system.

Note that the application utilizes the sensitive and private `OPERATOR_KEY`, `MASTER_KEY` and `NFT_STORAGE_API_KEY` information to operate. This may be suitable for a demonstration system, but is not sufficient for production deployment.

Recommendation

Implement authentication on the admin API. As the code base matures, this may involve several evolving levels of complexity (for example):

- The client supplies a secret `API_KEY` value with each request.
- The application server could validate[2] the client's TLS certificate (which `curl` can support[3]).
- Traditional accounts and passwords.[4]

Retest Results

NCC Group reviewed commit `e8f11a8` and observed the addition of the AuthenticationHandler.java class source file. This functionality extracts and checks the `x-api-key` request header, and has been registered within all 7 POST handlers of AdminApiVerticle.java. As such, this finding has been marked as 'Fixed'.

[2]https://vertx.io/docs/apidocs/io/vertx/core/http/HttpServerOptions.html#setClientAuth-io.vertx.core.http.ClientAuth-
[3]See the `--cert` and `--key` options in https://man7.org/linux/man-pages/man1/curl.1.html
[4]https://vertx.io/docs/vertx-auth-common/java/

Finding	**Missing HTTPS Transport Security**
Risk	**High** Impact: High, Exploitability: High
Identifier	NCC-E001942-006
Status	Fixed
Category	Cryptography
Component	hedera-nft-auction-demo-java-node
Location	• Absent in all Docker instances/configuration/setup. • hedera-nft-auction-demo/hedera-nft-auction-demo-java-node/src/main/java/com/hedera/demo/auction/app/api/AdminApiVerticle.java • hedera-nft-auction-demo/hedera-nft-auction-demo-java-node/src/main/java/com/hedera/demo/auction/app/api/ApiVerticle.java

Impact The unencrypted, unauthenticated and non integrity-checked HTTP protocol may allow attackers on the network to intercept, modify and replay traffic flowing between various components of the application. This could include the compromise of auctions and tokens, substitution or extraction of keys, or other malicious metadata modifications.

Description The HTTP protocol lacks support for the encryption, authentication and integrity provided by its close sibling HTTPS, and thus allows an attacker on the network to easily observe, modify and replay network traffic contents. Given the distributed and sensitive nature of the system, insecure transport must be avoided.

The `start()` method for the `AdminApiVerticle` class is shown below. Line 40 starts[5] a HTTP server with the default configuration.[6]

```
29  public void start(Promise<Void> startPromise) {
30
31      Dotenv env = Dotenv
32              .configure()
33              .filename(config().getString("envFile"))
34              .directory(config().getString("envPath"))
35              .ignoreIfMissing()
36              .load();
37
38      int httpPort = Integer.parseInt(Optional.ofNullable(config().getString(
     ➜  "ADMIN_API_PORT")).orElse(Optional.ofNullable(env.get("ADMIN_API_PORT")).
     ➜  orElse("9006")));
39
40      var server = vertx.createHttpServer();
```

The (elsewhere) specification of acceptable port lists, routing paths, firewall rules, network segmentation and the like, provide excellent complementary defense-in-depth measures but do not obviate the need for secure transport. These measures do not typically provide the required support for confidentiality, authenticity and integrity.

Recommendation Utilize HTTPS transport security[7,8,9] Develop and implement an explicit policy/strategy to-

[5] https://vertx.io/docs/apidocs/io/vertx/core/Vertx.html#createHttpServer--
[6] HttpServerOptions inherits from TCPSSLOptions https://vertx.io/docs/apidocs/io/vertx/core/net/TCPSSLOptions.html#DEFAULT_SSL
[7] https://aws.amazon.com/certificate-manager/
[8] https://letsencrypt.org/
[9] https://aws.amazon.com/blogs/aws/new-tls-termination-for-network-load-balancers/

wards checking certificate revocation. Given the transient nature of the application, business requirements may obviate the need for the revocation-checking aspect of HTTPS. Note that additional measures related to replay-resistance may be required at the application level.

Retest Results NCC Group reviewed commit `e8f11a8` and observed a large amount of changes pertinent to this finding, including functionality for certificate creation, application/build configuration, and HTTPS deployment. For example, AdminApiVerticle.java now starts an HTTPS server based on options enabling SSL. As such, this finding has been marked 'Fixed'.

Finding	**Default/Hardcoded Postgres Credentials**
Risk	**High** Impact: High, Exploitability: Medium
Identifier	NCC-E001942-008
Status	Fixed
Category	Configuration
Component	Deployment Environment
Location	• hedera-nft-auction-demo/hedera-nft-auction-demo-java-node/src/main/java/com/hedera/demo/auction/app/App.java • hedera-nft-auction-demo/docker-postgres/initdb.sh • hedera-nft-auction-demo/docker-compose.yaml • hedera-nft-auction-demo/docker-files/.env.sample • hedera-nft-auction-demo/docker-postgres/Dockerfile
Impact	An attacker may easily guess the default Postgres credentials gaining unrestricted access to the application's database for full system compromise.

Description

System breaches stemming from hardcoded credentials[10] commonly have extremely high-impact results.[11]

The initialization code in `App.java` attempts to read configuration data from the `.env` environment file. If the Postgres credentials are missing (or the file itself is missing), the code supplies a default username and password of 'postgres' and 'password'. The supplied `.env.sample` example file does not currently specify these values.

In the Postgres Docker container, the `initdb.sh` script captures the username/password from environment variables. The `docker-compose.yaml` file specifies these values as 'postgres' and 'password'. The adapted transcript below shows the state of the running Postgres container with these same values.

```
eschorn@ataraxy$ sudo docker exec -it 8042c293399f /bin/sh
/ # env
HOSTNAME=8042c293399f
...
POSTGRES_PASSWORD=password
...
POSTGRES_USER=postgres
...
```

As a result, both the demo application and Postgres database are configured to use the default hardcoded credentials of 'postgres' and 'password'. Note that the Dockerfile specifies permissive connectivity.

Recommendation

Do not use easily guessable passwords. Sensitive data should not be stored directly in source, but should instead be referenced from configuration files or environment variables that exist outside of the application's root. When checking code into a repository, use example files or

[10] https://owasp.org/www-project-top-ten/2017/A2_2017-Broken_Authentication
[11] Selected CVEs corresponding to CWE-798

stubs which are overwritten when the application is deployed[12],[13]

All secrets that have been checked into the source code repositories should be considered compromised, and changed.

Retest Results NCC Group reviewed commit `db2675a` and observed the following:

- Lines 58 and 60 of hedera-nft-auction-demo-java-node/src/main/java/com/hedera/demo/auction/app/App.java no longer have default values.
- The hedera-nft-auction-demo/docker-postgres/initdb.sh file required no change.
- The hedera-nft-auction-demo/docker-compose.yaml file no longer has hardcoded Postgres credentials.
- The docker-files/.env.sample file is no longer configured to accept hardcoded Postgres credentials.
- The hedera-nft-auction-demo/docker-postgres/Dockerfile required no change.

As such, this finding has been marked 'Fixed'.

[12] Hardcoded Credentials Example: http://nakedsecurity.sophos.com/2014/04/03/is-amazon-hacking-our-apps-or-doing-us-all-a-security-favour/

[13] SANS Top 25 Software Flaws: Number 11, Hardcoded Credentials: http://software-security.sans.org/blog/2010/03/10/top-25-series-rank-11-hardcoded-credentials/

Finding	**Directory Traversal**
Risk	**High** Impact: High, Exploitability: Low
Identifier	NCC-E001942-009
Status	Fixed
Category	Data Exposure
Component	hedera-nft-auction-demo-java-node
Location	• Line 42 of hedera-nft-auction-demo/hedera-nft-auction-demo-java-node/src/main/java/com/hedera/demo/auction/app/CreateAuction.java • Line 64 of hedera-nft-auction-demo/hedera-nft-auction-demo-java-node/src/main/java/com/hedera/demo/auction/app/Utils.java
Impact	An attacker may adjust the `auctionFile` attribute supplied to the `admin/auction` endpoint to expose the contents of arbitrary files within the Docker machine or exhaust memory with large files leading to a denial of service.

Description The `handle()` method for the `PostAuctionHandler` class retrieves the POST request body and maps it to a `RequestCreateAuction` object containing an `auctionFile` string among other items. This auction file name is subsequently passed into the `create()` method implemented in `CreateAuction.java` and partially excerpted below.

```
40  // submit message with auction file contents
41  log.info("Loading {} file", auctionFile);
42  String auctionInitData = Files.readString(Path.of(auctionFile),
     ➜ StandardCharsets.US_ASCII);
43
44  log.info("Submitting {} file contents to HCS on topic {}", auctionFile,
     ➜ localTopicId);
45
46  try {
47      TopicMessageSubmitTransaction topicMessageSubmitTransaction = new
         ➜ TopicMessageSubmitTransaction()
48              .setTopicId(TopicId.fromString(localTopicId))
49              .setTransactionMemo("CreateAuction")
50              .setMessage(auctionInitData);
51
52      TransactionResponse response =
         ➜ topicMessageSubmitTransaction.execute(hederaClient.client());
53  ...
```

In the above code, line 42 directly loads the contents of the specified file name, line 44 logs the file name (see also finding NCC-E001942-011 on page 19), and line 50 inserts the file contents into an outbound message which is then submitted. There are no constraints or validation applied to the file name. As a result, a POST request similar to that shown below will expose the contents of any readable files,[14] including log files (see finding NCC-E001942-010 on page 27 and finding NCC-E001942-011 on page 19). Since the application is running as root (see finding NCC-E001942-001 on page 25), these files may contain sensitive information.

```
curl -H "Content-Type: application/json" -X POST -d ' {
  "auctionFile": "/etc/adduser.conf",
```

[14] https://owasp.org/www-community/attacks/Path_Traversal

```
  "topicId": "asdf",
  "tokenid": "1.2.3",
  "auctionaccountid": "4.5.6",
  "reserve": "",
  "minimumbid": "1000000",
  "endtimestamp": "",
  "winnercanbid": true,
  "title": "Auction title",
  "description": "Auction description"
}' http://localhost:8082/v1/admin/auction
```

In addition, code in `Utils.java` (and called by `CreateToken.java`) performs similar functionality as shown below. Note that in this instance the modifier `UTF_8` is used in place of `US_ASCII` above. Nonetheless, no constraints or validation are applied to the file name.

```
content = new String ( Files.readAllBytes( Paths.get(filePath)), UTF_8);
```

In addition to exposing sensitive files, reading large files (for example `/dev/mem/`[15]) may exhaust memory and present a denial of service attack vector.

Recommendation Implement strong input validation around provided file names and paths to prevent the use of character sequences that facilitate path traversal. Constrain files to a single known subdirectory.

Retest Results NCC Group reviewed commit `e8f11a8` and observed the following:

- The functionality within `CreateAuction.java` source file has been reworked to avoid interacting with the file system.
- The `readFileIntoString()` function is no longer used (dead code)
- A new `fileIsAFile()` function has been added to the `Utils.java` source file that uses Java's Path class to differentiate a base/raw file name from one including path information. This is used in `RequestCreateToken.java` when working with the image description.

As such, this finding has been marked 'Fixed'.

[15] https://man7.org/linux/man-pages/man4/mem.4.html

Finding	**Incomplete Validity Check in AuctionKeyList API**
Risk	**High** Impact: High, Exploitability: High
Identifier	NCC-E001942-015
Status	Fixed
Category	Authentication
Component	hedera-nft-auction-demo-java-node
Location	hedera-nft-auction-demo/hedera-nft-auction-demo-java-node/src/main/java/com/hedera/demo/auction/app/AuctionKey.java
Impact	An adversary could repeat its public key, `threshold` times, during auction account creation and control the auction.

Description

In order to decentralize auction ownership, the auction account is designed to be controlled by multiple accounts. This means that a threshold number of accounts must sign the transaction that finalizes the auction, i.e. transfers the token to the winner and sends the final bid amount to the auction creator. It is important that the threshold is larger than one, since otherwise any single one of the auction account participants can stop the auction (if they do not like the bided price) or claim the winning bid and do not transfer the token to the winner.

The `AuctionKey` class uses a Java *ArrayList* to store the keys.[16] In Java, ArrayList is implemented as a variable sized array, and it does not guarantee that the array elements are unique. As a result it is possible to create an `auctionKeyList` with repeated public keys.[17] Note that the `isValid()` API does not check if the list of public keys are distinct.

```java
public class AuctionKey {
    @JsonProperty("key")
    public String key = "";
    @JsonProperty("keyList")
    public AuctionKeyList auctionKeyList = new AuctionKeyList();

    public Key toKeyList() {
        if (StringUtils.isEmpty(key)) {
            return auctionKeyList.toKeyList();
        } else {
            return PublicKey.fromString(key);
        }
    }

    public boolean isValid() {
        return auctionKeyList != null || !StringUtils.isEmpty(key);
    }
}
```

Reproduction Steps

As a proof of concept this unit test passes:

```java
@Test
    public void testAuctionKeyListSimple() {
```

[16] https://docs.oracle.com/javase/8/docs/api/java/util/ArrayList.html
[17] Similar logic exists in Hedera's Java SDK, see https://github.com/hashgraph/hedera-sdk-java/blob/bad0511b93fe2dece9a447f82568749ce68aee8a/sdk/src/main/java/com/hedera/hashgraph/sdk/KeyList.java.

```
        JsonArray keys = new JsonArray();
        JsonObject key1 = new JsonObject().put("key", publicKey1);
        JsonObject key2 = new JsonObject().put("key", publicKey1);
        keys.add(key1).add(key2);

        JsonObject keyList = new JsonObject();
        keyList.put("keys", keys);
        keyList.put("threshold", threshold1);

        JsonObject key = new JsonObject();
        key.put("keyList", keyList);

        AuctionKey auctionKey = key.mapTo(AuctionKey.class);

        assertEquals(keys.size(), auctionKey.auctionKeyList.auctionKeys.size());
        assertEquals(threshold1, auctionKey.auctionKeyList.threshold);

    }
```

NCC Group could also verify that it is possible to make a multiparty auction account that is practically using a single public key:

```
curl -H "Content-Type: application/json" -X POST -d '
{
  "keyList": {
    "keys": [
      {
        "key": "302a300506032b657003210090ec5045925d37b358ee0c60f858dc79c3b4370cb
 ➜  f7e0c5dad882f1171265cb3"
      },
      {
        "key": "302a300506032b657003210090ec5045925d37b358ee0c60f858dc79c3b4370cb
 ➜  f7e0c5dad882f1171265cb3"
      }
    ],
    "threshold": 2
  },
  "initialBalance": 100
}' http://localhost:8082/v1/admin/auctionaccount
```

This command successfully executes and creates an account on Hedera network.

Recommendation The `isValid()` API must check that the set of public keys are distinct.

Retest Results NCC Group reviewed commit `db2675a` and observed that the multiparty key handling logic has moved to the `PostAuctionAccountHandler` class which makes sure all public keys are valid, and there are no duplicated public keys (here). As such, this finding has been marked as 'Fixed'.

Finding **Missing Unicode Normalization Step**

Risk **Medium** Impact: Medium, Exploitability: Medium

Identifier NCC-E001942-003

Status Fixed

Category Data Validation

Component hedera-nft-auction-demo-java-node

Location hedera-nft-auction-demo/hedera-nft-auction-demo-java-node/src/main/java/com/hedera/demo/auction/app/api/PostEasySetupHandler.java

Impact An overlooked Unicode normalization step may result in user confusion or even a divergence between application instances that breaks interoperability.

Description When a user enters the same string identifier, differently encoded Unicode strings may arise from malicious intent or simply the broad range of participating devices, operating systems, locales, languages and applications involved.

Divergent string encoding typically involves characters with accents or other modifiers that have multiple correct Unicode encodings. For example, the Á (a-acute) glyph can be encoded as a single character U+00C1 (the "composed" form) or as two separate characters U+0041 then U+0301 (the "decomposed" form). In some cases, the order of a glyph's combining elements is significant and in other cases different orders must be considered equivalent.[18] At the extreme, the character U+FDFA can be correctly encoded as a single code point or a sequence of up to 18 code points.[19] A string identifier may appear identical but in fact be distinct, such as "Bank of Álpha" and "Bank of Álpha". Normalization[20,21,22] is the process of standardizing string representation such that if two strings are canonically equivalent and are normalized to the same normal form, their byte representations will be the same. Only then can string comparison, ordering and cryptographic operations be relied upon. There are four standard normalization forms,[23] of which NFKC[24] is the most popular and secure, though NFKD is also acceptable.

The Web API allows for a number of string parameters supplied as user input. As an example, the `handler()` for `PostEasySetupHandler` is shown below. The body is pulled from the request as a JSON object on line 24, mapped to an instance of `RequestEasySetup` on line 28, and individual fields extracted on lines 34 and 35. Unicode normalization is not performed prior to subsequent processing.

```
23 public void handle(RoutingContext routingContext) {
24     var body = routingContext.getBodyAsJson();
25     @Var RequestEasySetup data = new RequestEasySetup();
26 
27     if (body != null) {
28         data = body.mapTo(RequestEasySetup.class);
29     }
30 
```

[18] http://unicode.org/reports/tr15/tr15-22.html
[19] https://www.compart.com/en/unicode/U+FDFA
[20] https://docs.oracle.com/javase/tutorial/i18n/text/normalizerapi.html
[21] https://blog.golang.org/normalization
[22] https://docs.rs/unicode-normalization/0.1.13/unicode_normalization/
[23] http://unicode.org/reports/tr15/#Norm_Forms
[24] See question 2 of https://unicode.org/faq/normalization.html

```
    try {
        String[] args = new String[3];

        args[0] = "--symbol=".concat(data.symbol);
        args[1] = "--name=".concat(data.name);
...
```

As a result, input may be supplied with unusual encodings that result in a visually-identical but different string identifier. This could result in user confusion from typo squatting, or break application interoperability. This issue is applicable to all strings supplied as API input, with those operated upon as identifiers being the most sensitive, e.g., src/main/java/com/hedera/demo/auction/app/api/PostCreateToken.java#L33.

Recommendation Perform NKFC Unicode normalization on input strings immediately upon receipt and prior to subsequent processing.

Retest Results NCC Group reviewed commit `e8f11a8` and observed the addition of a `normalize()` function to the `Utils.java` source file to perform `Form.NFKC` string normalization. This function is used for the title, description, name, symbol, menu and nameToUpdate strings across 5 of the api source files. As such, this finding has been marked as 'Fixed'.

Finding **Log Injection**

Risk **Medium** Impact: Medium, Exploitability: Medium

Identifier NCC-E001942-011

Status Fixed

Category Auditing and Logging

Component hedera-nft-auction-demo-java-node

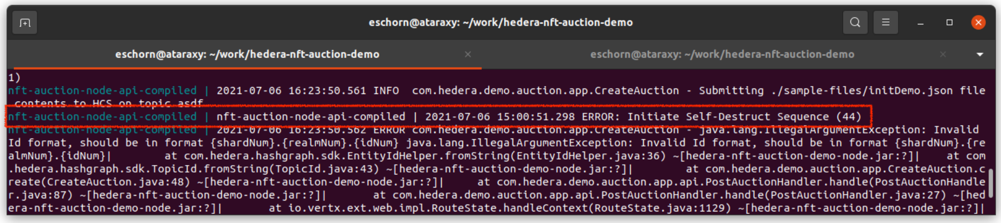
Location Line 44 of hedera-nft-auction-demo/hedera-nft-auction-demo-java-node/src/main/java/com/hedera/demo/auction/app/CreateAuction.java

Impact An attacker may be able to inject forged or malicious log entries in order to induce confusion related to repudiation, trigger inappropriate downstream activity and/or impede other auditing efforts.

Description Activity logs provide the primary source data for auditing and debugging purposes. In some blockchain applications, logging functions can also trigger external activity. When a user is able to inject log entries verbatim, they may be able to inject forged or malicious entries that induce confusion related to repudiation, trigger downstream activity and/or impact the auditing function. This can include format corruption, misdirection, duplication, deletion or other exploits.

The following curl command injects an entry into the application log as shown in the subsequent terminal screenshot.

```
curl -H "Content-Type: application/json" -X POST -d '{
  "topicId": "asdf\nnft-auction-node-api-compiled | 2021-07-06 15:00:51.298 ERROR
→   : Initiate Self-Destruct Sequence",
  "tokenid": "123",
  "auctionaccountid": "456",
  "reserve": "",
  "minimumbid": "1000000",
  "endtimestamp": "",
  "winnercanbid": true,
  "title": "Auction title",
  "description": "Auction description"
}' http://localhost:8082/v1/admin/auction
```


eschorn@ataraxy: ~/work/hedera-nft-auction-demo
eschorn@ataraxy: ~/work/hedera-nft-auction-demo
eschorn@ataraxy: ~/work/hedera-nft-auction-demo

```
1)
nft-auction-node-api-compiled | 2021-07-06 16:23:50.561 INFO  com.hedera.demo.auction.app.CreateAuction - Submitting ./sample-files/initDemo.json file
 contents to HCS on topic asdf
nft-auction-node-api-compiled | nft-auction-node-api-compiled | 2021-07-06 15:00:51.298 ERROR: Initiate Self-Destruct Sequence (44)
nft-auction-node-api-compiled | 2021-07-06 16:23:50.562 ERROR com.hedera.demo.auction.app.CreateAuction - java.lang.IllegalArgumentException: Invalid
Id format, should be in format {shardNum}.{realmNum}.{idNum} java.lang.IllegalArgumentException: Invalid Id format, should be in format {shardNum}.{re
almNum}.{idNum}|        at com.hedera.hashgraph.sdk.EntityIdHelper.fromString(EntityIdHelper.java:36) ~[hedera-nft-auction-demo-node.jar:?]|    at com
.hedera.hashgraph.sdk.TopicId.fromString(TopicId.java:43) ~[hedera-nft-auction-demo-node.jar:?]|        at com.hedera.demo.auction.app.CreateAuction.c
reate(CreateAuction.java:48) ~[hedera-nft-auction-demo-node.jar:?]|     at com.hedera.demo.auction.app.api.PostAuctionHandler.handle(PostAuctionHandle
r.java:87) ~[hedera-nft-auction-demo-node.jar:?]|       at com.hedera.demo.auction.app.api.PostAuctionHandler.handle(PostAuctionHandler.java:27) ~[hed
era-nft-auction-demo-node.jar:?]|       at io.vertx.ext.web.impl.RouteState.handleContext(RouteState.java:1129) ~[hedera-nft-auction-demo-node.jar:?]|
```

Figure 1: Injected log entry

As with insufficient input validation, this finding is systemic across multiple functions.

Recommendation Ideally, data originating from external input should not be logged. Where logging is necessary, the function must comprehensively validate input prior to its use. Stricter escaping, such as removal of non-alphanumeric ASCII characters, should be applied along with length restrictions. Sensitive data should not be logged.

Retest Results NCC Group reviewed commit `e8f11a8` and observed the following:

- The noted source file `CreateAuction.java` has been adapted to remove the logging statement causing this finding.
- An overall review of logging instances did not suggest the presence of similar logging issues elsewhere.

As such, this finding has been marked 'Fixed'.

Finding	**Missing Cleanup on Failure**
Risk	**Medium** Impact: Medium, Exploitability: Medium
Identifier	NCC-E001942-016
Status	Fixed
Category	Session Management
Component	hedera-nft-auction-demo-java-node
Location	1. hedera-nft-auction-demo-java-node/src/main/java/com/hedera/demo/auction/app/api/PostValidators.jav 2. hedera-nft-auction-demo-java-node/src/main/java/com/hedera/demo/auction/app/api/RequestCreateTol
Impact	Failure to clean up the validators' list after failure can lead to corrupted auction state.
Description	

1. Posting Validators' List

The `PostValidators` class prepares a POST call to set a list of validators list for an auction. It parses validators' metadata one by one (line 36) and sends a request transaction to the topic. If the request is successful, the validator will be added to the list, and it will proceed to add the next validator on the list. However if it fails, it will log the error, and return (Lines 58-60).

```
33  if (validators != null) {
34      for (Object validatorObject : validators) {
35          JsonObject validatorJson = JsonObject.mapFrom(validatorObject);
36          String[] args = new String[5];
37
38          args[0] = "--name=".concat(validatorJson.getString("name", ""));
39          args[1] =
              ➜ "--nameToUpdate=".concat(validatorJson.getString("nameToUpdate",
              ➜ ""));
40          args[2] = "--operation=".concat(validatorJson.getString("operation",
              ➜ ""));
41          args[3] = "--url=".concat(validatorJson.getString("url", ""));
42          args[4] = "--publicKey=".concat(validatorJson.getString("publicKey",
              ➜ ""));
43
44          try {
45              ManageValidator manageValidator = new ManageValidator();
46              manageValidator.manage(args);
47
48              JsonObject response = new JsonObject();
49              log.info("validator request submission successful");
50              response.put("status", "success");
51
52              routingContext.response()
53                      .putHeader("content-type", "application/json")
54                      .end(Json.encodeToBuffer(response));
55          } catch (Exception e) {
56              log.error(e, e);
57              routingContext.fail(500, e);
58              return;
59          }
60      }
61  } else {
    log.error("message body does not contain validators array");
    routingContext.fail(500);
    return;
}
```

As a result if the failure happens after one or more validators have successfully been added, it will result in a corrupted validators list state for the auction.

2. Uploading Token Metadata to IPFS

The `RequestCreateToken` class prepares a metadata payload and uploads it to IPFS, if successful it saves the returned CID to https://cloudflare-ipfs.com. The `saveImagesToIPFS()` API first uploads the image and then the certificate:

```
public void saveImagesToIPFS(String nftStorageKey) throws Exception {
    // does the metadata contain an image
    saveImageToIPFS(nftStorageKey, this.image);
    saveImageToIPFS(nftStorageKey, this.certificate);
}
```

If uploading the image succeeds, but uploading the certificate fails, it will leave the token's metadata storage in a corrupted state.

Recommendation

1. On failure, the handler should send `delete` requests for the successfully added validators.
2. One solution is to bundle the token metadata into a CAR file and upload it at once. Another solution is to DELETE the image from IPFS if uploading the certificate fails.

Retest Results

NCC Group reviewed commit `db2675a` and observed the following:

1. The list of all validators is bundled as a single transaction and is posted to the topic. Therefore in case of failure, none of the validators will be set for the auction. (see here.)
2. In case uploading the certificate to IPFS fails, the uploaded image for the token is deleted from IPFS.

Finding **Missing State Validation before Transition**

Risk **Medium** Impact: Medium, Exploitability: Medium

Identifier NCC-E001942-017

Status Fixed

Category Data Validation

Component hedera-nft-auction-demo-java-node

Location
- AuctionsRepository.setActive()
- setStatus()
- setStarttimestamp()

Impact An attacker can replay the initial token transfer to re-activate closed/ended auctions. An auction that requires a token transfer to the winner does not transition from Closed to Ended.

Description An auction's lifecycle is managed as a state machine: `Pending` -> `Active` -> `Closed`/`Ended` . The `AuctionsRepository` class provides an API for an Admin to set auction state as it progresses via `setActive()`, `setEnded()`, and `setClosed()` but fails to check if the auction was in the expected state before transitioning it.

At the application level, this API is used by the validators that monitor an auction's transfers:

Pending to Active Transition: `AuctionReadinessWatcher` frequently fetches transfers and if it finds the token transfer from the token owner to the auction account, it sets the auction status to Active (AuctionReadinessWatcher.java). Since neither `AuctionReadinessWatcher` nor `AuctionsRepository` verify that the auction was in Pending state, this opens up the possibility to replay a mirror's transfers and get the `AuctionReadinessWatcher` to re-Activate an Ended auction.

Closed to Ended Transition: `AuctionsClosureWatcher` monitors the mirror for auctions that are past their end timestamp. It will transition them to Closed if `transferOnWin` flag is set, and to Ended otherwise (AuctionsClosureWatcher.java). In parallel, `AuctionEndTransfer` monitors closed auctions and transfers the token to the winner; on success it must set the auction's state to Ended, but it does not (AuctionEndTransfer).

Reproduction Steps

```
@Test
public void setAuctionActiveAfterEndedTest() throws Exception {
    auctionsRepository.setEnded(auction.getId());
    auctionsRepository.setActive(auction, auction.getTokenowneraccount(),
     ➜  auction.getStarttimestamp());
    Auction getAuction = auctionsRepository.getAuction(auction.getId());

    assertEquals(Auction.ACTIVE, getAuction.getStatus());
    assertEquals(auction.getStarttimestamp(), getAuction.getStarttimestamp());
}
```

Recommendation The `AuctionRepository` API must check auction's state before transitioning it:

1. Check that `auction.isPending()` returns true before setting the status to `ACTIVE`.
2. Check that `auction.isActive()` returns true before setting the status to `CLOSED` or `END`

`ED`.

Retest Results NCC Group reviewed commit `a86f15c` and observed that auction's state is only transitioned if its current state is as expected. Unit tests have also been added to cover failure scenarios. As such, this finding has been marked as 'Fixed'.

Finding	**Docker Containerized (Guest) Applications Execute As Root**
Risk	**Low** Impact: Medium, Exploitability: Low
Identifier	NCC-E001942-001
Status	Fixed
Category	Configuration
Component	Deployment Environment
Location	hedera-nft-auction-demo/hedera-nft-auction-demo-java-node/Dockerfile
Impact	An attacker able to escape the application would have elevated root privileges inside the container. Elevated privileges inside the container increase the opportunity to escape the container. An attacker able to escape the container would have elevated root privileges on the host.

Description Docker container processes typically execute as root on the host OS, which is a known (but required and generally accepted) risk because the container must interact with the host kernel as root. However, if a container breakout were identified, the attacker would then have root privileges on the host.[25]

As an attack surface reduction technique and best practice, the containerized (guest) application should be run within a restricted user role or employ user namespaces. An attacker able to escape the application would then operate in a low-privilege context rather than as root. This would present obstacles to attacks that require user-to-kernel interactions, which require a privileged account. For example, if a device driver were mapped into the container and it required root privileges to access its IOCTLs that expose kernel objects, then the restricted user would be unable to communicate with it, thus preventing exposure of a potential flaw.

The shell session shown below, after bringing up a default configuration per the project `README.md`, demonstrates two of the four containers with guest applications running as root.

```
$ sudo docker exec -it cf33b497ca74 ps auwx      # (demo_rest-compiled)
USER         PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root           1  0.0  0.0   2616   596 ?        Ss   13:42   0:00 /bin/sh -c
 ➜  "java" "-Dlog4j.configurationFile=/demo/log4j2.xml" "-jar" /srv/hedera-
root           7  4.1  1.6 4641932 135244 ?      Sl   13:42   0:02 java
 ➜  -Dlog4j.configurationFile=/demo/log4j2.xml -jar /srv/hedera-
root          32  0.0  0.0   8900  3168 pts/0    Rs+  13:43   0:00 ps auwx

$ sudo docker exec -it 9a2c5784d3b4 ps auwx      # (demo_node-compiled)
USER         PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root           1  0.0  0.0   2616   600 ?        Ss   13:42   0:00 /bin/sh -c
 ➜  "java" "-Dlog4j.configurationFile=/demo/log4j2.xml"
root           6  4.1  2.9 4746416 237784 ?      Sl   13:42   0:05 java
 ➜  -Dlog4j.configurationFile=/demo/log4j2.xml -jar /srv/hedera-
root          59  0.0  0.0   8900  3168 pts/0    Rs+  13:44   0:00 ps auwx
```

The other two containers have guest processes running in lower privilege service accounts (nginx and postgres). These standard Dockerfiles explicitly add a low-privilege service account[26] for running the guest application.

[25] https://i.blackhat.com/USA-19/Thursday/us-19-Edwards-Compendium-Of-Container-Escapes-up.pdf
[26] https://github.com/nginxinc/docker-nginx/blob/f3fe494531f9b157d9c09ba509e412dace54cd4f/mainline/debian/Dockerfile#L17

Recommendation Enforce a restricted user context by crafting a hardened Dockerfile using the `RUN` and `USER` directives, which will run the guest application in a service account, similar to the nginx and postgres Dockerfiles and/or the below fragment:

```
FROM <base image>
RUN groupadd -g 999 appuser && useradd -r -u 999 -g appuser appuser
USER appuser
... <rest of Dockerfile> ...
```

Consider additional hardening strategies found in the CIS Benchmarks[27] for Docker.

Retest Results NCC Group reviewed commit `739ec11` and observed the addition of `USER appuser` related commands in the hedera-nft-auction-demo-java-node/Dockerfile file. As such, this finding has been marked 'Fixed'.

[27] https://www.cisecurity.org/benchmark/docker/

Finding	**Logging of Private Key**
Risk	**Low** Impact: High, Exploitability: Undetermined
Identifier	NCC-E001942-010
Status	Fixed
Category	Auditing and Logging
Component	hedera-nft-auction-demo-java-node
Location	Line 27 of hedera-nft-auction-demo/hedera-nft-auction-demo-java-node/src/main/java/com/hedera/demo/auction/app/GenerateKey.java
Impact	A private key is logged in cleartext and thus vulnerable to disclosure over the log lifecycle.

Description Developers must be very careful when writing to logs as they can be a shared resource with many opportunities for exposure during their lifecycle. The application logs a private key in plaintext as shown by the code excerpt below.

```java
24 public static void main(String[] args) {
25     GenerateKey generateKey = new GenerateKey();
26     PrivateKey privateKey = generateKey.generate();
27     log.info("Private Key: {}", privateKey.toString());
28     log.info("Public Key: {}", privateKey.getPublicKey().toString());
29 }
```

As this function appears to be only used during build rather than deployment, the risk rating has been reduced to 'Low'.

Recommendation Remove all code that logs private keys, passwords and any other sensitive information. Alternatively, mask sensitive fields (e.g. with '***') before writing output. Do not encrypt sensitive data in the logs as it then necessitates additional and complex key management functionality.

Retest Results NCC Group reviewed commit `db2675a` and observed that the hedera-nft-auction-demo-java-node/src/main/java/com/hedera/demo/auction/app/GenerateKey.java file now utilizes `System.out.println` rather than logging to output the private key. As such, this finding has been marked 'Fixed'.

Finding	**Outdated Java Dependencies**
Risk	**Informational** Impact: Undetermined, Exploitability: Undetermined
Identifier	NCC-E001942-002
Status	Fixed
Category	Patching
Component	hedera-nft-auction-demo-java-node
Location	hedera-nft-auction-demo/hedera-nft-auction-demo-java-node/build.gradle
Impact	An attacker may attempt to identify and utilize vulnerabilities in outdated dependencies to exploit the application.

Description Using outdated dependencies with discovered vulnerabilities is one of the most common and serious routes of application exploitation. Many of the most severe breaches have relied upon exploiting known vulnerabilities in dependencies.[28]

Running the `./gradlew dependencyUpdates` tool returns an overview of the project's dependency status. Most key dependencies are up to date and the relative overall status is good. However, a minority of dependencies are outdated and should be reviewed.

```
Outdated major versions
   - com.google.guava:guava [29.0-jre -> 30.1.1-jre]
   - io.github.jklingsporn:vertx-jooq-classic-reactive [5.2.0 -> 6.3.0]
   - org.glassfish.jaxb:jaxb-core [2.3.0.1 -> 3.0.2-b01]
   - org.glassfish.jaxb:jaxb-runtime [2.3.3 -> 3.0.2-b01]
   - org.slf4j:slf4j-log4j12 [1.7.28 -> 2.0.0-alpha2]

Outdated minor versions
   - com.google.errorprone:error_prone_annotations [2.4.0 -> 2.6.0]
   - io.grpc:grpc-netty-shaded [1.35.0 -> 1.39.0]
   - io.vertx:vertx-core [4.0.3 -> 4.1.1]
   - io.vertx:vertx-junit5 [4.0.3 -> 4.1.1]
   - io.vertx:vertx-pg-client [4.0.3 -> 4.1.1]
   - io.vertx:vertx-unit [4.0.3 -> 4.1.1]
   - io.vertx:vertx-web [4.0.3 -> 4.1.1]
   - io.vertx:vertx-web-client [4.0.3 -> 4.1.1]
   - javax.xml.bind:jaxb-api [2.3.1 -> 2.4.0-b180830.0359]
   - org.awaitility:awaitility [4.0.3 -> 4.1.0]
   - org.flywaydb:flyway-core [7.7.0 -> 7.11.0]
   - org.junit.jupiter:junit-jupiter-api [5.4.2 -> 5.8.0-M1]
   - org.junit.jupiter:junit-jupiter-engine [5.4.2 -> 5.8.0-M1]
```

Note that development dependencies are of a lower priority since the application does not expose them at runtime. Moreover, some transitive dependencies are outside of the control of the project.

Recommendation Update all dependencies and tools to the latest versions recommended for production deployment. Add a gating milestone to the development process that involves reviewing all dependencies for outdated or vulnerable versions.

Retest Results NCC Group reviewed commit `739ec11` and re-ran the `./gradlew dependencyUpdates` tool.

[28] https://arstechnica.com/information-technology/2017/09/massive-equifax-breach-caused-by-failure-to-patch-two-month-old-bug/

As the original status was noted to be generally good, a majority of the most central and most dated dependencies were updated, and this finding is rated ‘Informational‘, the finding has been marked ‘Fixed’.

Finding **Missing Null Check in `PostAuctionHandler`**

Risk **Informational** Impact: None, Exploitability: None

Identifier NCC-E001942-007

Status Fixed

Category Error Reporting

Component hedera-nft-auction-demo-java-node

Location Line 83 of hedera-nft-auction-demo/hedera-nft-auction-demo-java-node/src/main/java/com/hedera/demo/auction/app/api/PostAuctionHandler.java

Impact An unhandled null pointer exception may indicate insufficient data validation and may leave the application in an undesired intermediate state.

Description The `PostAuctionHandler` class handles the creation of auctions with the logic relating to the 'topic ID' shown below.

```
81 CreateAuction createAuction = new CreateAuction();
82 createAuction.setEnv(env);
83 @Var String localTopicId = env.get("TOPIC_ID");
84 if (! StringUtils.isEmpty(data.topicId)) {
85     localTopicId = data.topicId;
86 }
87 createAuction.create(fileName, localTopicId);
88 ...
```

The statements on lines 84-86 are intended to (re)set the `localTopicId` variable only if `data.topicId` is not blank or null. This prevents potentially passing a null value into the second argument of `createAction.create()`. The logic preceding the above snippet involves the writing of files in local storage that could then be left in an intermediate state in the case of an exception occurring.

However, the `env.get()` function call on line 83 may return a null value. If this happens and line 85 is not executed, a null value is passed into the second argument of `createAction.create()` which will attempt to dereference it and throw an exception.

This can be achieved by starting a fresh docker application instance and executing the following curl command.

```
curl -H "Content-Type: application/json" -X POST -d '
{
  "tokenid": "123",
  "auctionaccountid": "456",
  "reserve": "",
  "minimumbid": "1000000",
  "endtimestamp": "",
  "winnercanbid": true,
  "title": "Auction title",
  "description": "Auction description"
}' http://localhost:8082/v1/admin/auction
```

Recommendation Check the validity of all input data prior to processing it.

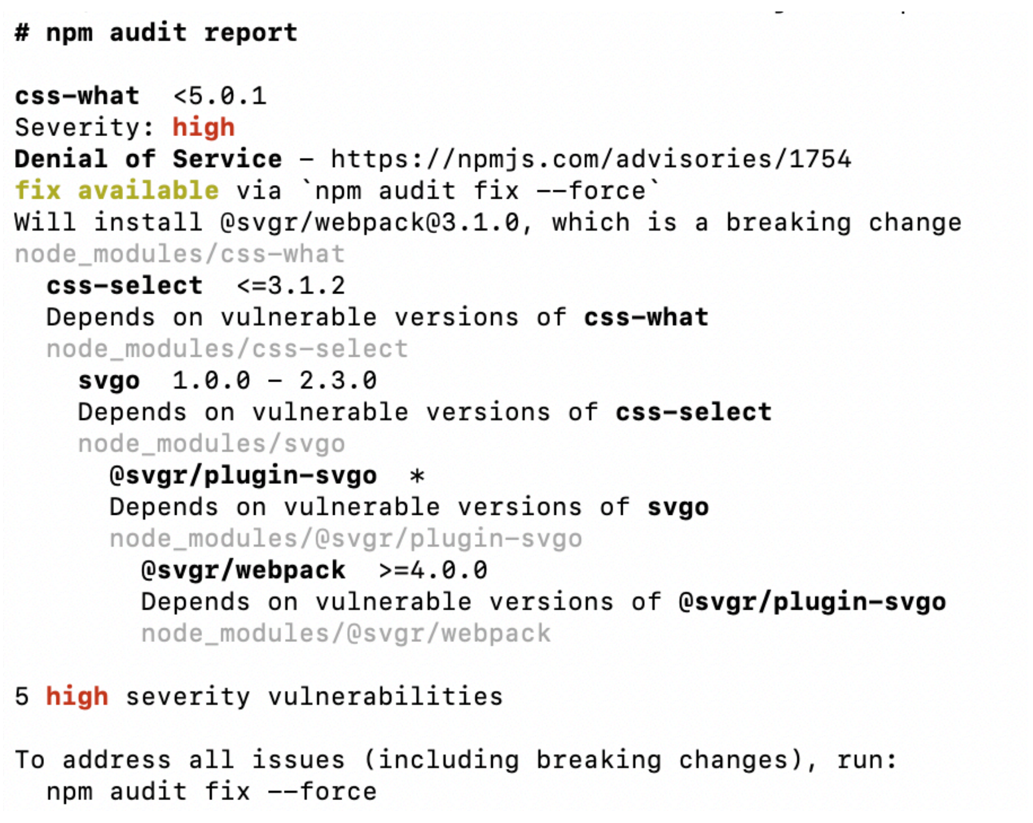
Retest Results NCC Group reviewed commit `db2675a` and observed a new null check on line 62 to prevent the code from subsequently (unexpectedly) dereferencing a null value. As such, this finding has been marked ‘Fixed’.

Finding	**Mock Object Creation Should Not Have Public Visibility**
Risk	**Informational** Impact: Low, Exploitability: Medium
Identifier	NCC-E001942-013
Status	Fixed
Category	Data Exposure
Component	hedera-nft-auction-demo-java-node
Location	hedera-nft-auction-demo-java-node/src/main/java/com/hedera/demo/auction/app/HederaClient.java hedera-nft-auction-demo-java-node/src/main/java/com/hedera/demo/auction/BidsWatcher.java
Impact	Including mock object creation and test flags with public visibility in the main package can lead to unexpected behavior in production and slow down debugging as the application becomes more complex.
Description	One instance of mock object creation for testing is the `emptyTestClient()` API which creates a mock client, that only interacts with the *testnet*. Although exposing this API does not harm the *mainnet*, it can lead to unexpected behavior on the *testnet*. Another example of public API that is meant for testing is `TopicSubscriber`, `AuctionReadinessWatcher`, `Refunder`, `BidsWatcher`, and `AuctionEndTransfer`'s `setTesting()` API that sets the `testing` flag. Conditional on this flag being set, parts of the implementation are turned off. This can lead to shipping untested code and is dangerous in production. As the development team grows, unfamiliar members might use test cases as a template for development and assume that all the public APIs will always be available. This can lead to confusion and unexpected behavior.
Recommendation	It is better design to move mock object creation to the test module. In addition, to test parts of the logic, it is best to break it down into smaller functions and test them individually.
Retest Results	NCC Group reviewed commit `db2675a` and observed that all mock object creations have been moved out of the `src/` folder and the `testing` flag is no longer used to disable parts of the logic. As such, this finding has been marked as 'Fixed'.

Finding	**Using Vulnerable Node Dependencies**
Risk	**Informational** Impact: Low, Exploitability: Low
Identifier	NCC-E001942-014
Status	Reported
Category	Patching
Component	hedera-nft-auction-demo-javascript-client
Location	https://github.com/hashgraph/hedera-nft-auction-demo/tree/6c9cd8d9af1252548abab5634749b19102b0a14b/hedera-nft-auction-demo-javascript-client
Impact	Using the `svgro` package versions lower than `5.0.1` can lead to Denial of Service Attacks.

Description According to this CVE (https://nvd.nist.gov/vuln/detail/CVE-2021-33587), (now) deprecated versions of Node.js failed to ensure input parsing has linear time complexity. This could lead to DoS attacks on the client UI application. While reviewing the UI application was not in scope, NCC Group highly advises to patch the UI application and use `svgr/webpack` version `5.0.1` or higher.

This vulnerability was found through `npm audit`:



```
# npm audit report

css-what  <5.0.1
Severity: high
Denial of Service - https://npmjs.com/advisories/1754
fix available via `npm audit fix --force`
Will install @svgr/webpack@3.1.0, which is a breaking change
node_modules/css-what
  css-select  <=3.1.2
  Depends on vulnerable versions of css-what
  node_modules/css-select
    svgo  1.0.0 - 2.3.0
    Depends on vulnerable versions of css-select
    node_modules/svgo
      @svgr/plugin-svgo  *
      Depends on vulnerable versions of svgo
      node_modules/@svgr/plugin-svgo
        @svgr/webpack  >=4.0.0
        Depends on vulnerable versions of @svgr/plugin-svgo
        node_modules/@svgr/webpack

5 high severity vulnerabilities

To address all issues (including breaking changes), run:
  npm audit fix --force
```

Figure 2: npm audit

Recommendation Run `npm audit fix --force` to upgrade the vulnerable packages.

Appendix A: Implementation Review Notes

This section highlights a few minor observations that do not pose any security risks.

- The TopicSubscriber should set the query parameters such that the response is sorted in ascending order based on message timestamps; otherwise, the last transfer might not have the latest processed timestamp, and `nextTimestamp` will be set incorrectly. Note that in live networks multiple messages might have the same timestamp, and querying the Mirror for messages with timestamp greater than the last processed timestamp can potentially lead to unprocessed messages. It might be more robust to query the Mirror based on an index such as transaction ID, if the support exists in the Mirror API.
- Filtering `MirrorTransactions` for a given token ID is common between the transaction watchers (AuctionEndTransfer and AuctionReadinessWatcher). This can be refactored into a function that takes in the Mirror transactions and the `tokenId`, and returns a list of matching transactions:

```
TransferResult transferOccurredAlready(MirrorTransactions mirrorTransactions, String tokenId) {
    // Empty results list.
    for (MirrorTransaction transaction : mirrorTransactions.transactions) {
        if (transaction.isSuccessful()) {
            for (MirrorTokenTransfer tokenTransfer : transaction.tokenTransfers) {
                if (tokenTransfer.tokenId.equals(‹tokenId›)) {
                    // Add the transfer to the list.
                }
            }
        }
    }
    // Return the list.
}
```

- Local handler functions' visibility should be set to `private`, e.g. handleTransaction(), to limit exposure.
- If the new bid is *not* the winner, AuctionsRepository.commitBidAndAuction() will unnecessarily update the winning bid to what it was. This `update` operation should be executed only if the new bid is the new winner.
- AuctionReadinessWatcher class should save the `nextTimestamp` in persistent memory, e.g. auctions repository, otherwise if the thread is restarted it will re-activate the auction and set a new bid watcher for it. This mechanism is already supported in BidWatcher.
- In this sentence: The highest bigger on the HNFT will receive a signed print of the painting by the artist, `bigger` should be `bidder`.
- This if (env != null) check is performed after the `env` variable is dereferenced a few times; if it is in fact `null` the function will throw an exception earlier.

Appendix B: Finding Field Definitions

The following sections describe the risk rating and category assigned to issues NCC Group identified.

Risk Scale

NCC Group uses a composite risk score that takes into account the severity of the risk, application's exposure and user population, technical difficulty of exploitation, and other factors. The risk rating is NCC Group's recommended prioritization for addressing findings. Every organization has a different risk sensitivity, so to some extent these recommendations are more relative than absolute guidelines.

Overall Risk

Overall risk reflects NCC Group's estimation of the risk that a finding poses to the target system or systems. It takes into account the impact of the finding, the difficulty of exploitation, and any other relevant factors.

Critical	Implies an immediate, easily accessible threat of total compromise.
High	Implies an immediate threat of system compromise, or an easily accessible threat of large-scale breach.
Medium	A difficult to exploit threat of large-scale breach, or easy compromise of a small portion of the application.
Low	Implies a relatively minor threat to the application.
Informational	No immediate threat to the application. May provide suggestions for application improvement, functional issues with the application, or conditions that could later lead to an exploitable finding.

Impact

Impact reflects the effects that successful exploitation has upon the target system or systems. It takes into account potential losses of confidentiality, integrity and availability, as well as potential reputational losses.

High	Attackers can read or modify all data in a system, execute arbitrary code on the system, or escalate their privileges to superuser level.
Medium	Attackers can read or modify some unauthorized data on a system, deny access to that system, or gain significant internal technical information.
Low	Attackers can gain small amounts of unauthorized information or slightly degrade system performance. May have a negative public perception of security.

Exploitability

Exploitability reflects the ease with which attackers may exploit a finding. It takes into account the level of access required, availability of exploitation information, requirements relating to social engineering, race conditions, brute forcing, etc, and other impediments to exploitation.

High	Attackers can unilaterally exploit the finding without special permissions or significant roadblocks.
Medium	Attackers would need to leverage a third party, gain non-public information, exploit a race condition, already have privileged access, or otherwise overcome moderate hurdles in order to exploit the finding.
Low	Exploitation requires implausible social engineering, a difficult race condition, guessing difficult-to-guess data, or is otherwise unlikely.

Category

NCC Group categorizes findings based on the security area to which those findings belong. This can help organizations identify gaps in secure development, deployment, patching, etc.

Access Controls	Related to authorization of users, and assessment of rights.
Auditing and Logging	Related to auditing of actions, or logging of problems.
Authentication	Related to the identification of users.
Configuration	Related to security configurations of servers, devices, or software.
Cryptography	Related to mathematical protections for data.
Data Exposure	Related to unintended exposure of sensitive information.
Data Validation	Related to improper reliance on the structure or values of data.
Denial of Service	Related to causing system failure.
Error Reporting	Related to the reporting of error conditions in a secure fashion.
Patching	Related to keeping software up to date.
Session Management	Related to the identification of authenticated users.
Timing	Related to race conditions, locking, or order of operations.

Appendix C: Project Contacts

The team from NCC Group has the following primary members:

- Parnian Alimi — Consultant
  parnian.alimi@nccgroup.com
- Eric Schorn — Consultant
  eric.schorn@nccgroup.com
- Javed Samuel — Vice President, Practice Director Cryptography Services
  javed.samuel@nccgroup.com

The team from Hedera Hashgraph has the following primary members:

- Greg Scullard — Hedera
  gregscullard@hedera.com
- Alex Popowycz — Hedera
  a@hedera.com