CS 6035

FLAG 6: Restful Data (15 pts)

Make sure you have gone through the Setup and Intro sections.

If you haven't already, run the start script in the home directory of log4j user, start the container with the start script:

```
./StartContainer.sh
```

For this flag we will use the /products/ resource. There are four endpoints for this resource:

GET All - Fetch all product records

```
curl 'http://localhost:8080/rest/products/productlist' -H 'GATECH_ID:123456789'
```

GET By ID - Fetch a single product record by ID

```
curl 'http://localhost:8080/rest/products/product/<id>' -H 'GATECH_ID:123456789'
```

GET By Email - Fetch all records associated with an email (This will require you to create a new product with an email field. The initial set of products do not have an email persisted, they return with the default email)

```
curl 'http://localhost:8080/rest/products/product?email=example@example.com' -H 'GATECH_ID:123456
```

POST Create or Update a new record - Post a new record or update an existing record by providing the id in the request

```
curl -X POST 'http://localhost:8080/rest/products/product' \
-H 'GATECH_ID:123456789' \
-H 'Content-Type: application/json' \
-d '{
"make": "Ford",
"model": "Mustang",
```

```
"trim": "GT",
"price": 45000.00,
"email": "example@example.com"
}'
```

We've explored introducing malicious strings to be triggered by the application as it accepts and processes an HTTP request. For this flag we're going to explore an often overlooked attack vector for exploits like Log4J: Data at Rest.

Data at Rest is data that has already been persisted to some data store and is sitting idle. In the case of log4shell, this could be data that is structured in such a way that when the application retrieves and attempts to use or log it, it triggers the LDAP call.

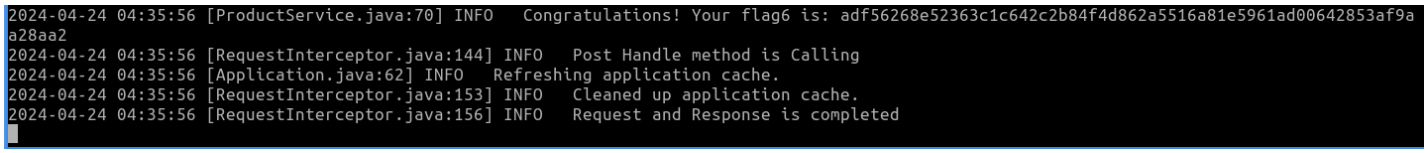
Use the product POST endpoint to persist a record to the database that, when retrieved later, will trigger the LDAP call. You will have to inspect the logs of each of the endpoints to come up with a successful attack strategy.

You will use the log4j exploit to update the "config.properties" file saved in the root directory of the application similar to what you did in Flag 3 and Flag 5. You will write a new property product.id that should have the value set to the id of the malicious product record that you have created/updated.

When the application fetches the record upon calling the right GET endpoint, it will trigger the exploit and, if successful, will generate the Flag 6 message in the logs.

Note: You will have to trigger the LDAP call with the malicious record in order to generate the Flag.

Upon success, you should see your output similar to that below:



***** **IF THIS FLAG COMES OUT BLANK, Restart container by running the stopContainer and startContainer scripts in the home directory of the log4j user. *******