SCENARIO

The application contains a hidden blog page which contains a secret password which we can be accessed by using GraphQL queries. We will try to exploit this vulnerability and get the secret password hidden within the hidden page.

**PROCEDURE**

1. Open the web application and study the request made to the server.
2. We see that there are 5 posts in total but only 4 of them are visible to us and those post are equipped with a serialized **postID.**
3. Now we will scan the GraphQL endpoint and get to know that the BlogPost type has a **postPassword** field available which causes the site to make a GraphQL query that fetches the relevant post data via a direct reference to the post's ID.
4. Navigate to any post, send the request to the BurpSuite’s Repeater modify the id variable to 3 and add the postPassword field to the query like the Payload.
5. Now, in the response we will get the hidden blog with the desired secret key.

PAYLOAD

{"variables": {"id": 3}, "query": "\n query getBlogPost($id: Int!) {\n getBlogPost(id: $id) {\n image\n title\n author\r\n\t\t\t\t\t\tpostPassword\n date\n paragraphs\n }\n }", "operationName": "getBlogPost"}

REMEDIATION

1. Access Control: Always ensure that GraphQL endpoints have proper access controls in place. Even if data is not directly exposed to the user interface, ensure that unauthorized users can't query for it.
2. No Leaky Schemas: Keep your GraphQL schema free of information about internal objects or potentially sensitive data if they're not meant for client consumption. A client should never be aware of fields or operations they can't access.
3. Field-level Authorization: Implement authorization checks at the field level in your resolvers to ensure that sensitive fields (like postPassword in this example) are not exposed to users who don't have the necessary permissions.
4. Depth Limiting: Impose a limit on the depth of queries to prevent deep and costly nested queries.
5. Amount Limiting: Limit the amount of data a user can request in a single query. For example, restrict the number of items a user can request from a list or the number of objects they can retrieve.