Title

GraphQL Schema Exposure via Error Responses Allowing Field Enumeration and Privilege Mapping

---

Summary

Despite GraphQL introspection being disabled, the API still returns detailed validation error responses that expose schema structure, available fields, mutation names, and authorization flow. This allows an attacker to progressively enumerate the schema and perform targeted mutation testing, increasing the risk of privilege escalation, unauthorized data access, or account takeover.

Endpoint(s) Affected

`https://https://[redacted].bumba.[redacted]/graphql`

Suggested Severity

Low/Medium

(If chained with successful privilege escalation → can become High.)

Impact

An attacker with a low-privilege authenticated account can:

- Enumerate valid query and mutation names
- Identify hidden fields and backend entity structures
- Map authorization boundaries based on error changes
- Identify potential high-value mutations such as:

1. `update_vault`
2. `create_user`
3. `delete_user`

This significantly expands the attack surface and directly assists in crafting future exploitation attempts, including privilege escalation, unauthorized data access, and business logic abuse.

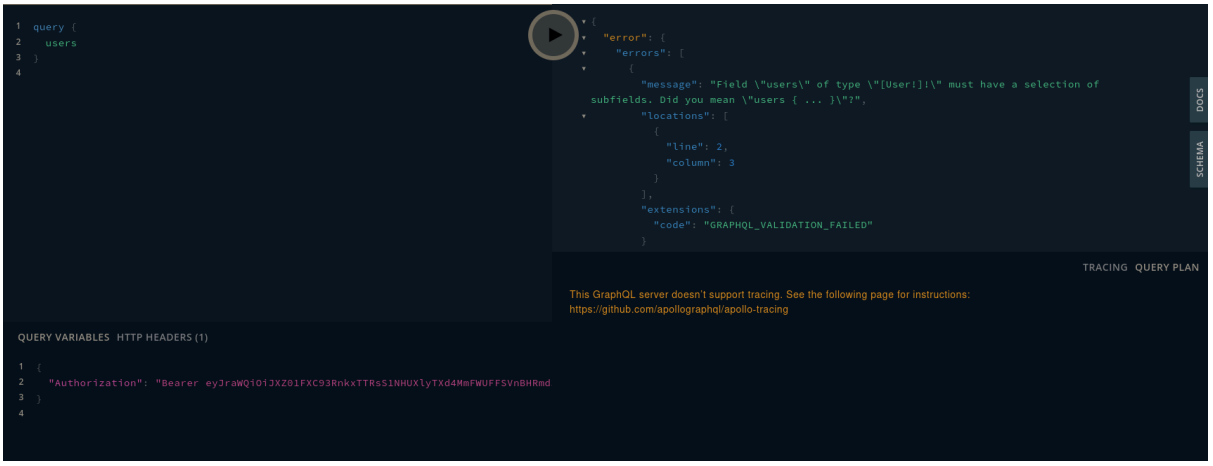
Steps to Reproduce

1. Send a basic request

```
query {
  users
}
```

Response:

```
/"Field \"users\" of type \"[User!]!\" must have a selection of subfields. Did you mean \"users { ... }\"?"
```



⟶ This confirms the field exists.

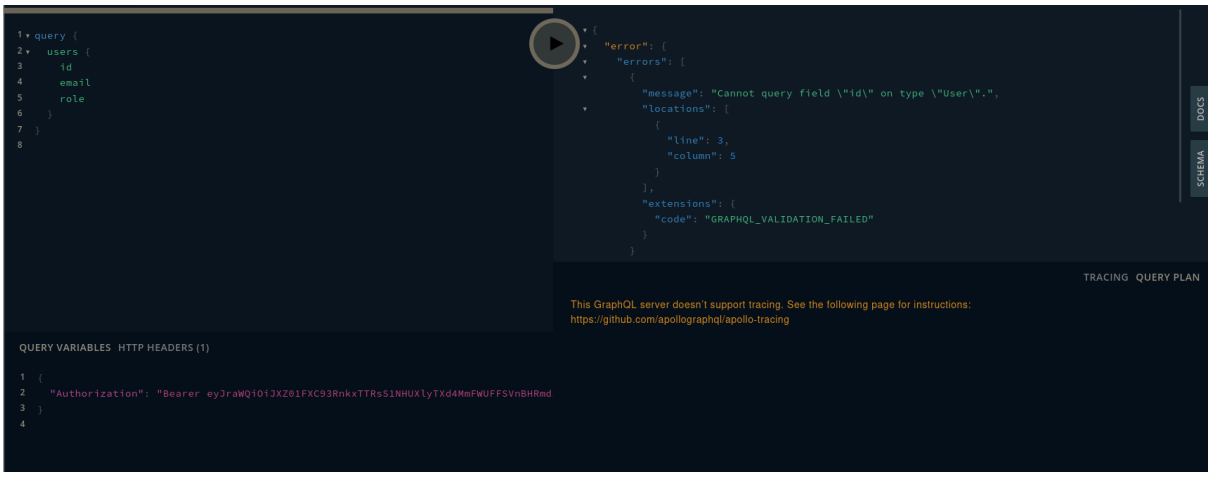
---

2. Attempt field enumeration

```
query {
  users {
    id
    email
    role
  }
}
```

Response:

```
"Cannot query field \"id\" on type \"User\"."
```



⟶ This leaks details schema even without introspection.

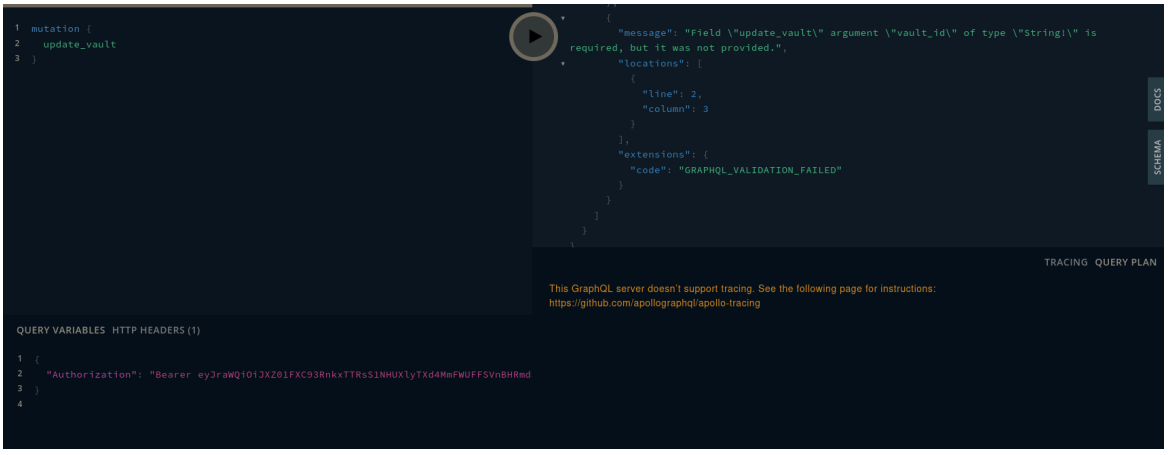
---

3. Enumerate mutations

```
mutation {
  update_vault
}
```

Response:

```
"Field \"update_vault\" argument \"vault_id\" of type \"String!\" is required"
```



⟶ Exposes mutation name + required arguments.

Proof of Auth Bypass Difference

Sending requests with and without JWT produces different error messages:

Request	Result
No JWT	`AUTH_GUARD_NO_JWT_FOUND_IN_HEADERS`
Invalid JWT	`AUTH_GUARD_JWT_TOKEN_NOT_AUTHORIZED`
Valid JWT	`Cannot query field ...` → schema leakage

This allows attackers to map permission levels and role-based access control behavior.

Expected Behavior

Error responses should be generic and must not reveal internal schema or mutation names.

Example expected response:

```
"Invalid request"
```

Recommendations

- Enable GraphQL production security mode
- Disable detailed validation error messages
- Return standardized generic errors (`Apollo + Helmet recommended`)
- Enforce RBAC on field level access
- Consider implementing allow list schema exposure

Optional Hardening

✔️ Enable rate limiting
✔️ Disable field suggestions (`Did you mean?`)
✔️ Enforce query depth and cost limits

References

- OWASP API Security: API3 – Excessive Data Exposure
- CWE-209 – Information Exposure Through Error Messages
- Apollo Best Practices Security Guide

Reporter Environment

- Login required: **yes**
- Method: manual enumeration + logic probing
- No exploitation or harm performed

End of report.