APOLLO

GRAPHQL CONF 2026

DoS Wars: Revenge of the Fragments

Sachin Shinde Staff Software Engineer, Apollo GraphQL

Modularize data requirements with named fragments

BEFORE without_fragments.graphql

```
query {
  me {
    id
    name
    email
    friends {
      id
      name
      email
    }
  }
}
```

AFTER with_fragments.graphql

```
query {
  me {
    ...UserInfo
    friends {
      ...UserInfo
    }
  }
}

fragment UserInfo on Identity {
  id
  name
  email
}
```

COMMON PITFALL #1

Named fragments can form cycles

fragment_cycles.graphql

```graphql
query {
  me {
    ...A
  }
}

fragment A on Identity {
  ...B
}

fragment B on Identity {
  id
  ...A
}
```

Mitigations

- GraphQL spec validation checks for cycles
- Validate queries before execution/processing

COMMON PITFALL #2

Named fragments can multiply field executions

fragment_multiplication.graphql

```graphql
query {
  me {
    a: bestFriend { ...F1 }
    b: bestFriend { ...F1 }
  }
}

fragment F1 on Identity {
  a: bestFriend { ...F2 }
  b: bestFriend { ...F2 }
}

# ...
fragment F100 on Identity {
  a: bestFriend { id }
  b: bestFriend { id }
}
```

Mitigations

- Query cost analysis (e.g. IBM cost spec)
- Time/Resource limiting

Selection set execution deduplicates fields and named fragments

selection_set_execution.graphql

```graphql
query {
  me {
    friends { id }
    ...A
    ...B
  }
}

fragment A on Identity {
  friends { name }
  ...B
}

fragment B on Identity {
  friends { email }
}
```

Field Collection

- Traverse inline/named fragments of selection set
- Collect fields per response name
- Response name's field executed at most once
- Named fragments visited at most once

Deduplication can lower execution cost for deeply nested queries

linear_execution_cost.graphql

```graphql
query {
  me {
    bestFriend { ...F1 }
    bestFriend { ...F1 }
  }
}

fragment F1 on Identity {
  bestFriend { ...F2 }
  bestFriend { ...F2 }
}

# ...
fragment F100 on Identity {
  bestFriend { id }
  bestFriend { id }
}
```

scalar_execution_cost.graphql

```graphql
query {
  me {
    ... on Human { ...F1 }
    ... on Robot { ...F1 }
  }
}

fragment F1 on Identity {
  ... on Human { ...F2 }
  ... on Robot { ...F2 }
}

# ...
fragment F100 on Identity {
  ... on Human { id }
  ... on Robot { id }
}
```

VULNERABILITY #1

Avoid revisiting named fragments

scalar_execution_cost.graphql

```graphql
query {
  me {
    ... on Human { ...F1 }
    ... on Robot { ...F1 }
  }
}

fragment F1 on Identity {
  ... on Human { ...F2 }
  ... on Robot { ...F2 }
}

# ...
fragment F100 on Identity {
  ... on Human { id }
  ... on Robot { id }
}
```

CVE-2025-31496

- GraphQL spec validation implementation
- Traverses inline/named fragments of selection sets
- Only need to visit named fragments once per set
- Did not track already visited fragments

VULNERABILITY #2

Limit expanded query size

scalar_execution_cost.graphql

```graphql
query {
  me {
    ... on Human { ...F1 }
    ... on Robot { ...F1 }
  }
}

fragment F1 on Identity {
  ... on Human { ...F2 }
  ... on Robot { ...F2 }
}

# ...
fragment F100 on Identity {
  ... on Human { id }
  ... on Robot { id }
}
```

CVE-2025-32030/ CVE-2025-32034

- Federation query planner
- Traverses named fragments necessarily inline
- Effectively expands query size via substitution
- Did not precompute how large this query would be

VULNERABILITY #3

Prevent integer wraparound for limits

scalar_execution_cost.graphql

```graphql
query {
  me {
    ... on Human { ... F1 }
    ... on Robot { ... F1 }
  }
}

fragment F1 on Identity {
  ... on Human { ... F2 }
  ... on Robot { ... F2 }
}

# ...
fragment F100 on Identity {
  ... on Human { id }
  ... on Robot { id }
}
```

CVE-2025-32033

- Plugin limiting various query metrics
- Traverses inline/named fragments of queries
- Computes metric for a fragment at most once
- Used wrapping arithmetic for metric

KEY TAKEAWAYS

Mitigation Strategies

01

Avoid revisiting named fragments when traversing selection sets

Many algorithms and validations can process named fragments without upstream context

02

Compute and limit the query’s size if named fragments were substituted inline

For algorithms that must process named fragments inline or logic outside your control

03

Prevent integer wraparound when computing metrics for limits

Named fragment nesting and reuse can lead to exponential metric growth