GraphQL
For Network and Security Engineers

Damien Garros
Infrastructure Automation Architect

FRNOG 36, September 2022
Agenda

What is GraphQL

GraphQL vs REST

Use Cases for Network and Security Engineers
  - Securing GraphQL applications
  - Infrastructure Automation

Best Practices

Integration with Ansible (demo)
Damien Garros

Freelance Automation Architect

Infrastructure as Code, Automation & Observability for 10+ years,

Previously leading Technical Architecture at

Spent 9 years in the US, recently moved back to France

damiengarros  @dgarros  @damgarros
GraphQL

● Data **query and manipulation language** for APIs
● Designed to make APIs fast, flexible, and developer-friendly.
● Move some complexity out of the client, on the server side.
● Complementary / Alternative to REST API
● Developed at Facebook, released in 2015
GraphQL vs REST

{ REST }

Device
/api/device
/api/interface
/api/ipaddr

3 queries + post processing

IP Address
/graphql

1 query
GraphQL Components

- **Query**: Query data
- **Mutation**: Manipulate Data
  - Create/Update/Delete
  - Action
- **Subscription**: Real Time Update
  - (Over WebSocket)

Graphql Schema

- **Resolvers**
  - SQL Database
  - Graph Database
  - No-SQL Database
Other Features

- Variables
- Fragments
- Multiple queries
- Introspection Query
Use Cases for Network and Security Engineer
Use Cases for Network and Security Engineer

Manage / Secure a GraphQL Application

Network Automation
Applications supporting GraphQL today

- Jira Software
- netbox
- nautobot
- apstra
- tufin
- GitHub

List non exhaustive
Applications that COULD support GraphQL tomorrow??

Anyone
exposing relational data
via a REST API

Some might already support it but I couldn’t find it in my research
Manage / Secure GraphQL Applications

- All requests go to the same endpoint, HTTP status code are not relevant (always returns 200)
- Potential vector of attack
  - DOS / Query are very powerful (too powerful ?)
  - Batch of queries
  - Full schema available,
Network Automation with GraphQL

- Easier to query and consume data.
- Less effort required on the client or in Jinja
- Very easy to pair with Jinja2 to generate files
Best Practices

- Be mindful of the “Cost” of your queries
- Use filter to narrow down what you want to retrieve
- In Ansible / Nornir, don’t query everything in the inventory, query additional information at the beginning of the playbook instead
Integration with Ansible
Leverage GraphQL with Ansible

**DO**
Fast and efficient

**DON'T**
Inefficient and very slow

**Inventory**
- Query minimal device info

**Device A**
- Query Additional Info
- Task2
- Task3

**Device B**
- Query Additional Info
- Task2
- Task3

**Inventory**
- Query all variables
  - devices, interfaces
  - ip etc..

May or may not use GraphQL too
Inventory

---

```yaml
plugin: networktocode.nautobot.gql_inventory
api_endpoint: https://demo.nautobot.com/
token: "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa"
query:
devices:
  device_role: name
  status: slug
  site:
    name:
      filters:
        site: cdg01
group_by:
  - device_role.name
  - status.slug
```
---

- name: "Generate Device Configuration with Data Gathered with GraphQL"
  hosts: all
  gather_facts: false
  connection: local
  vars:
    nautobot_token: "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa"
    query_string: |
      query ($device_name: [String!]) {
        devices (name: $device_name) {
          id
          name
          interfaces {
            name
            ip_addresses {
              address
            }
          }
        }
      }
Playbook 2/2

tasks:
- name: "Gather Interfaces and IP addresses information per device"
  networktocode.nautobot.query_graphql:
    url: "https://demo.nautobot.com/"
    token: "{{ nautobot_token }}"
    query: "{{ query_string }}"
    graph_variables: "{{ gql_variables }}"
    update_hostvars: yes

vars:
  gql_variables:
    device_name: "{{ inventory_hostname }}"

- name: "Generate Configuration"
  ansible.builtin.template:
    src: "config_template.j2"
    dest: "generated_configs/{{ inventory_hostname }}.cfg"
Thank you.
## GraphQL vs REST

<table>
<thead>
<tr>
<th></th>
<th>GraphQL</th>
<th>REST</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Schema</strong></td>
<td>Yes</td>
<td>Recommended</td>
</tr>
<tr>
<td><strong>Endpoint(s)</strong></td>
<td>Only One</td>
<td>1 per Object Type</td>
</tr>
<tr>
<td><strong>Payload</strong></td>
<td>On Demand</td>
<td>Fixed</td>
</tr>
<tr>
<td><strong>Read</strong></td>
<td>Query</td>
<td>GET</td>
</tr>
<tr>
<td><strong>Create</strong></td>
<td>Mutation</td>
<td>POST</td>
</tr>
<tr>
<td><strong>Update</strong></td>
<td>Mutation</td>
<td>PUT</td>
</tr>
<tr>
<td><strong>Delete</strong></td>
<td>Mutation</td>
<td>DELETE</td>
</tr>
<tr>
<td><strong>Notification</strong></td>
<td>Subscription</td>
<td>With HTTP/2</td>
</tr>
</tbody>
</table>