Attack surface service - exercise

Description

At Orca our mission is to secure our customers' cloud environments. We would want to provide customers the ability to understand the potential attack vectors risking their cloud data centers. You should build a service that a customer can query and get the attack surface of a VM - meaning which other virtual machines in the account can access and attack it.

Details

Cloud environment

The input for your service is a JSON document describing the cloud environment of a customer.
A cloud environment is described using 2 types of objects: VMs and firewall rules.
The structure of the cloud environment JSON is:

```
{
      “vms”: [ virtual machines ],
      “fw_rules”: [ firewall rules ]
}
```

Virtual Machine

A virtual machine has the following structure:

```
{
      "vm_id": "vm-xxxxxxx",
      "name": "jira server",
      "tags": ["tag1", ..]
}
```

vm_id - an identifier that uniquely identifies a virtual machines
name - a user-friendly display name
tags - a list of zero or more tag strings

Firewall Rule

By default, a virtual machine has no access from external sources.
If an administrator wants to make a virtual machine accessible to other machines, it defines a firewall rule to allow traffic

Firewall rules have the following structure:

```
{
      "fw_id": "fw-xxxxx",
      "source_tag": "tag1",
      "dest_tag": "tag2"
}
```

fw_id - an identifier that uniquely identifies a firewall rule
source_tag - a string that represents the source tag of a traffic
dest_tag - a string that represents the destination tag of a traffic

In the example above, all traffic from virtual machines that have “tag1” is allowed to virtual machines that have “tag2”.

Your goal:

Build a service that has two REST endpoints:

- /attack - which will get a vm_id as a query parameter and return a JSON list of the virtual machine ids that can potentially attack it
- /stats - which will return service statistics in a JSON format: number of virtual machines in the cloud environment, number of requests to all endpoints & average request processing time (in milliseconds). Statistics should be from process startup.

Example of using the attack endpoint:

```
$ curl 'http://localhost/api/v1/attack?vm_id=vm-a211de'
["vm-c7bac01a07"]
```

Example of using the stats endpoint:

```
$ curl 'http://localhost/api/v1/stats'
{"vm_count":2,"request_count":1120232,"average_request_time":0.0030322
68166772597}
```

We will evaluate the exercise based on:

- Code quality - correct, concise, readable, well-organized implementation
- Performance and Scalability - how well the code performs and scales to many requests and bigger input set
- Completeness of solution - tests (bonus), handling of edge cases and logging

Attached are a few inputs we gathered from cloud environments, the service will accept a JSON file as input at startup.