In the next following document, we will describe the home assignment, which is a part of the Wiz Backend Engineering recruitment process.

General guidelines

1. The assignment takes around 4 hours, in which you will go over the requirements, design and discuss a high-level design for your solution, and finally implement as much as you can achieve in the given timeframe. We favor higher quality code more than being able to complete more checkpoints of the assignment.
2. While designing and implementing your solution, make sure to take extensibility, readability and testability into consideration. No need to make things over-generic.
3. You can select any programming language you want. Preferably, select Golang. However, if you are unfamiliar with Golang, please select the language in which you are most familiar. Any language that has support for OOP is fine (e.g., Java, C#, Python, NodeJS/TypeScript, C++).

Assignment

Acme Inc. is concerned about the potential risks of it's users using various cloud services. In order to get a better understanding of the scope of the problem, the security team decided to build a database of cloud services, with various fields (DNS name, service type, risk level, country of origin and more) and use it to understand which cloud services are used by its users, based in the organization's firewall logs.

Checkpoints

1. Write a program that takes the Cloud services database and a firewall log, correlates the records, and outputs the list of **distinct** internal IPs (as in, don't count the same IP multiple times) that have used each of the cloud services listed in the database.
2. The security team found out that some of the firewall records only have an IP and don't contain a hostname, which prevents our program from identifying some of the services. Add a reverse-DNS layer to the solution, for records that only have IPs. Make sure to cache your results.
3. Add filtering rules:
    a. **Include IP filters** –These filters will allow to **only** include specific records. The filter should support single IPs and IP ranges (x.x.x.x/y). An example for include rule: 1.1.1.1/24 (which translates to 1.1.1.0-1.1.1.255), only IPs within this range will be evaluated. If no include filter is specified, all IPs should be included.
    b. **Exclude IP filters** –These filters will exclude records by IPs. The filter should support single IPs and IP ranges (x.x.x.x/y). An example for exclude rule: 2.2.2.2/16 (which translates to 2.2.0.0-2.2.255.255), IPs within this range should not be evaluated.
    c. **Users include filter** - This filter should include users based on a regular expression.
    d. **Users exclude filter** - This filter should exclude users based on a regular expression.

    The program should support handling multiple filters of potentially multiple types during its correlation process. As in, for example, include IPs of ranges 1.1.1.0/16, 8.8.8.0/24, and exclude the range 1.1.2.0/24 and the IP 8.8.8.8, all whilst excluding users which their name starts with the string "sys".
4. Add concurrency to the program, such that it will handle multiple firewall records concurrently.
5. Save your results into SQLite to allow different type of queries and aggregations. Write a short explanation regarding your schema design, index design, and which queries you expect to have on your data.

Format samples

Firewall log incoming record with domain

```
Feb  1 00:00:02 bridge kernel: INBOUND TCP: IN=br0 PHYSIN=eth0 OUT=br0 PHYSOUT=eth1
SRC=192.150.249.87 DST=11.11.11.84 LEN=40 TOS=0x00 PREC=0x00 TTL=110 ID=12973
PROTO=TCP SPT=220 DPT=6129 WINDOW=16384 RES=0x00 SYN URGP=0
DOMAIN=www.dropbox.com
```

Firewall log incoming record with user

```
Feb  1 00:00:02 bridge kernel: INBOUND TCP: IN=br0 PHYSIN=eth0 OUT=br0 PHYSOUT=eth1
SRC=192.150.249.87 DST=11.11.11.84 LEN=40 TOS=0x00 PREC=0x00 TTL=110 ID=12973
PROTO=TCP SPT=220 DPT=6129 WINDOW=16384 RES=0x00 SYN URGP=0
USER=dave@acme.com
```

Firewall log incoming record with user and domain

```
Feb  1 00:00:02 bridge kernel: INBOUND TCP: IN=br0 PHYSIN=eth0 OUT=br0 PHYSOUT=eth1
SRC=192.150.249.87 DST=11.11.11.84 LEN=40 TOS=0x00 PREC=0x00 TTL=110 ID=12973
PROTO=TCP SPT=220 DPT=6129 WINDOW=16384 RES=0x00 SYN URGP=0
USER=dave@acme.com DOMAIN=www.dropbox.com
```

Firewall log incoming record without user and domain

```
Feb  1 00:00:02 bridge kernel: INBOUND TCP: IN=br0 PHYSIN=eth0 OUT=br0 PHYSOUT=eth1
SRC=192.150.249.87 DST=11.11.11.84 LEN=40 TOS=0x00 PREC=0x00 TTL=110 ID=12973
PROTO=TCP SPT=220 DPT=6129 WINDOW=16384 RES=0x00 SYN URGP=0
```

Firewall log outgoing record

```
Feb 27 14:40:06 bridge kernel: OUTG CONN TCP: IN=br0 PHYSIN=eth1 OUT=br0 PHYSOUT=eth0
SRC=11.11.11.71 DST=77.88.55.80LEN=40 TOS=0x00 PREC=0x00 TTL=64 ID=50688 DF
PROTO=TCP SPT=80 DPT=1325 WINDOW=6432 RES=0x00 ACK URGP=0
```

Database format sample (CSV)

```
Service name,Service domain,Risk,Country of origin,GDPR Compliant
Google Drive,drive.google.com,Medium,US,Yes
```