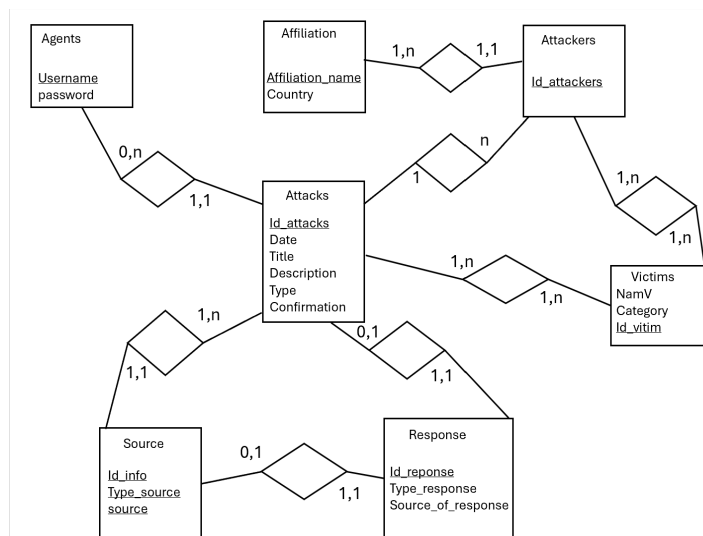


TP SIP Période 1

NOTE:

- Dans test_agents on a rajouté des fonctions pour tester les fonctions dans agents
- De plus, on a réalisé un notebook qui permet de tester toutes les fonctions dans le dossier /db/.
- On a aussi un notebook qui teste toutes les fonctions de /routes/ qui fonctionnent.

1)



Agent (Username, password)

Attacks (Id_attacks, date, title, description, type, confirmation, Id_Attackers, username)

Source (Id_info, type_source, source, Id_attacks, Id_response)

Response (Id_Response, Type_response, source_of_response, Id_attacks)

Attackers (Id_attackers, Affiliation_name)

Victim(Id_victim, NameV, Category)

Attackers_Victim (Id_Attackers, Id_victim)

Attacks_Victims(Id_victim, Id_Attacks)

Affiliation (Affiliation, Country)

Au début, on n'avait pas vu 3NF donc on avait des listes de victimes liées à chaque attaque par cases dans la table Victim.

Ensuite par des modifications de code dans la fonction populate, la table est 3NF en isolant chaque victime dans chaque case et en liant Victim et Attack par une table intermédiaire

2) Cf Code

3) Cf Code : le nouveau fichier s'appelle "nouveau_BDD.csv"

4) Cf Code

5) Cf Code

6) Cf Code

7) Cf code

8) J'ai bien lu le code donné

9) Après la fonction "insert_Hubert()" il manque on ne voit pas de "conn.commit()" dans le code.

En ajoutant "conn.commit()" dans "/test_agents.py/test_insert_agent/" juste après "insert_Hubert()" on voit apparaître Hubert dans la table Agent

10) On a cette erreur qui apparaît : "An integrity error occurred while insert the agent: UNIQUE constraint failed: Agent.username"

11) De la même manière qu'à la question 9) je n'ai réussi à résoudre le problème en modifiant "test_update_password" dans /test_agent.py/

12) Voici l'erreur obtenu: "## TEST: update an agent that does not exist in the database
Aucun agent trouvé avec le nom d'utilisateur 'bond'.
Update successful: None
Number of modified rows in the database: 0"

Aucune modification n'est faite

Toutes les fonctions de db/agents sont testées dans db/test_agents fonctionnent.

13) Cf Code

14) Le code définit une application Flask qui utilise des blueprints pour structurer les routes. Il initialise la base de données et exécute l'application sur un serveur de développement.

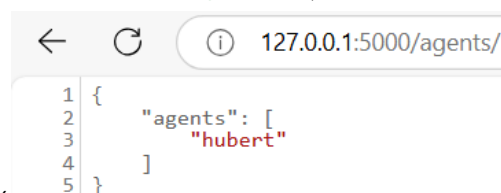
La fonction `init_database()` est appelée pour configurer la base de données avant de démarrer le serveur.

Le serveur de développement est lancé, et l'application est accessible via

<http://127.0.0.1:5000>.

15) The path of the route to get all agents is `/agents/`

Faire une requête GET dans le lien HTTP pour obtenir tous les agents peut être fait avec le lien: (<http://127.0.0.1:5000/agents/>)



```
1 {  
2   "agents": [  
3     "hubert"  
4   ]  
5 }
```

et cela nous donne : ()

16) Cf code

17) Cf code

18) Cf code

19) Cf code

20) Cf code

21) Cf code

22) Cf code

23) Cf code

24)

Dans **routes/data/**: on a mis un contrôle de token sur 2 des 4 fonctions:

`-get_targets()`

`-get_attackers()`

`-get_sources()`, `get_responses()` n'ont pas besoin de token car ce sont des liens vers des sites et articles en open source donc libre d'accès

Toutes les fonctions de **routes/agents/** doivent être protégées car elles touchent des données personnelles pour les requêtes GET et toutes les requêtes PATCH et POST doivent avoir un token car les données et la BDD ne doivent pas être modifiées par n'importe qui.

`-get_all_agents`

`-get_agent`

`-patch_password`

`-add_agent`

Dans les fonctions de **routes/incidents/** seule une n'a pas besoin d'être protégé:

`-get_incident` : les attaques doivent pouvoir être accessibles à tous

Les autres fonctions permettent de modifier la BDD donc doivent être modifiés:

`-assign_incident`

`-update_incident`

`-add_element_to_incident`

`-remove_element_to_incident`