BENCHRIFA Mohamed Amine
CADI Hamza
DESTREE Gabriel
NDIAYE El Hadj Pathé
PORQUEZ William

Compte Rendu – Sécurité dans les Réseaux - 6h

API – Serveur de découverte des nœuds

(Blockchain)

SOMMAIRE

SUJET

Implémenter une API pour serveur de découverte de nœuds (et un simulateur de client) en Flask (Python) :

```
GET /nodes/         liste de tous les IDs des noeuds connus
GET /node/<ID>      information sur le noeud ID
POST /node/<ID>     ajouter ou mettre à jour le noeud ID
```

Les données sont toutes envoyées en JSON. A vous de définir les informations que vous mettez (pensez à la vie privée mais aussi à l'authenticité des données.

SCENARIO

On a un ensemble de nœud et on souhaite gérer la découverte de nœuds par un système « Peer to Peer » respectant l'authentification et la vie privée des utilisateurs.

Chaque utilisateur est un nœud : il fait initialement une demande au nœud principal qui lui communique la liste de tous les nœuds.

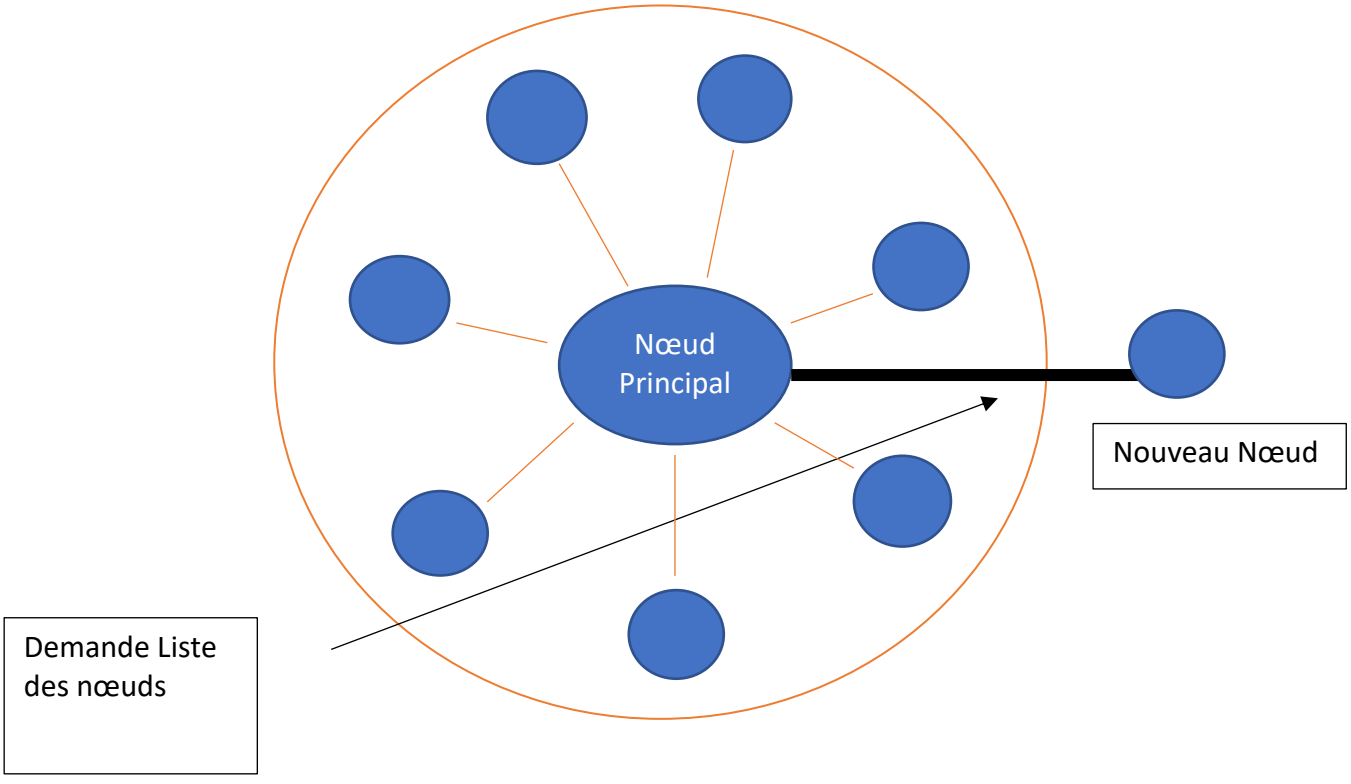
Un client se connecte au réseau, on va lui attribuer un ID, une clé publique.

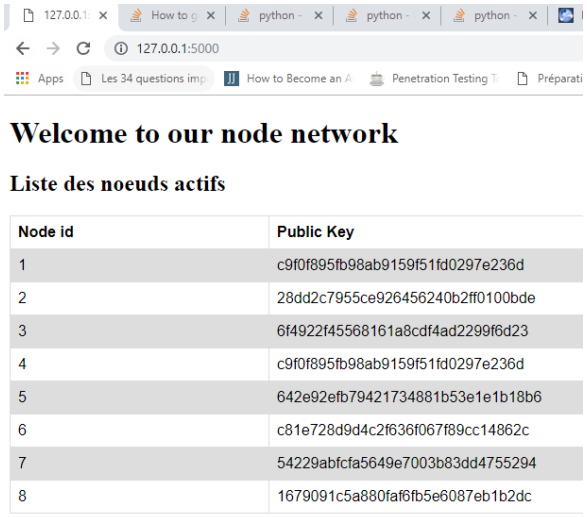
Une fois connecté, il aura accès à une liste de tous les nœuds existants avec lesquelles il pourra communiquer. Une fois qu'un nœud communique sa table de routage à notre client, on se basera sur le principe des Trackers pour sécuriser cet échange.

Utilisation des trackers :

Un tracker (nœud principal) recense les nœuds intéressés par un même contenu de façon à les mettre en relation les uns avec les autres. Les nœuds intéressées par un même contenu vont former ce qu'on appelle une grappe. Le rôle du tracker est alors de transmettre les informations de chaque membre de la grappe aux nouveaux nœuds qui en font la demande.

Ce qui caractérise et identifie le nœud principal qui contient toutes les informations des autres nœuds, c'est l'info hash. Ce info hash est une donnée essentielle que l'on retrouve dans la table qui contient les informations des autres nœuds.





127.0.0.1:5000

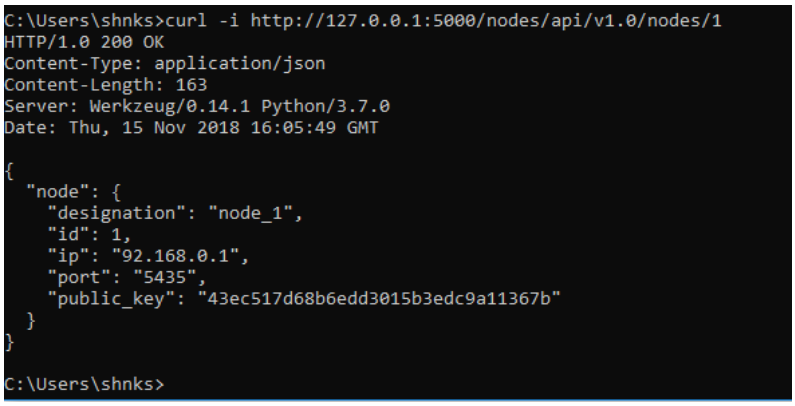
Apps | Les 34 questions imp | How to Become an A | Penetration Testing T | Préparati

Welcome to our node network

Liste des noeuds actifs

Node id	Public Key
1	c9f0f895fb98ab9159f51fd0297e236d
2	28dd2c7955ce926456240b2ff0100bde
3	6f4922f45568161a8cdf4ad2299f6d23
4	c9f0f895fb98ab9159f51fd0297e236d
5	642e92efb79421734881b53e1e1b18b6
6	c81e728d9d4c2f636f067f89cc14862c
7	54229abfcfa5649e7003b83dd4755294
8	1679091c5a880faf6fb5e6087eb1b2dc

Capture d'écran 1



```
C:\Users\shnks>curl -i http://127.0.0.1:5000/nodes/api/v1.0/nodes/1
HTTP/1.0 200 OK
Content-Type: application/json
Content-Length: 163
Server: Werkzeug/0.14.1 Python/3.7.0
Date: Thu, 15 Nov 2018 16:05:49 GMT

{
  "node": {
    "designation": "node_1",
    "id": 1,
    "ip": "92.168.0.1",
    "port": "5435",
    "public_key": "43ec517d68b6edd3015b3edc9a11367b"
  }
}

C:\Users\shnks>
```

Capture d'écran 2

Command Prompt

```
  }
}

C:\Users\shnks>curl -i http://127.0.0.1:5000/nodes/api/v1.0/nodes
HTTP/1.0 200 OK
Content-Type: application/json
Content-Length: 1723
Server: Werkzeug/0.14.1 Python/3.7.0
Date: Thu, 15 Nov 2018 16:01:47 GMT

{
  "nodes": [
    {
      "designation": "node_1",
      "ip": "92.168.0.1",
      "port": "5435",
      "public_key": "aab3238922bcc25a6f606eb525ffdc56",
      "uri": "http://127.0.0.1:5000/nodes/api/v1.0/nodes/1"
    },
    {
      "designation": "node_2",
      "ip": "92.168.0.1",
      "port": "5435",
      "public_key": "19ca14e7ea6328a42e0eb13d585e4c22",
      "uri": "http://127.0.0.1:5000/nodes/api/v1.0/nodes/2"
    },
    {
      "designation": "node_3",
      "ip": "92.168.0.1",
      "port": "5435",
      "public_key": "3295c76acbf4caaed33c36b1b5fc2cb1",
      "uri": "http://127.0.0.1:5000/nodes/api/v1.0/nodes/3"
    },
    {
      "designation": "node_4",
      "ip": "92.168.0.1",
      "port": "5435",
      "public_key": "f033ab37c30201f73f142449d037028d",
      "uri": "http://127.0.0.1:5000/nodes/api/v1.0/nodes/4"
    },
```

Capture d'écran 3

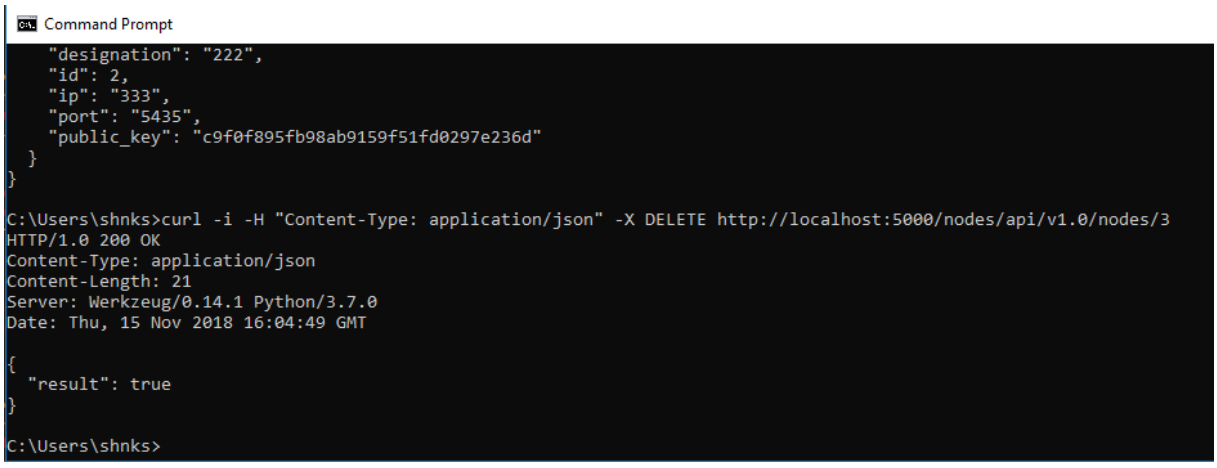
- La première capture est la page réalisé en flask permettant de générer des clés publiques différents pour chaque nœud connecté à notre réseau.
- La deuxième et troisième capture démontre le fonctionnement de la méthode GET
- La quatrième capture démontre le fonctionnement de la méthode POST
- La cinquième capture démontre fonctionnement de la méthode PUT
- La dernière capture démontre le fonctionnement de la méthode `DELETE.

```
C:\Users\shnks>curl -i -H "Content-Type: application/json" -X POST -d "{""designation"":""noeud9"",""ip"":""4.3.2.1"",""port"":""3
244""}" http://localhost:5000/nodes/api/v1.0/nodes
HTTP/1.0 201 CREATED
Content-Type: application/json
Content-Length: 105
Server: Werkzeug/0.14.1 Python/3.7.0
Date: Thu, 15 Nov 2018 16:03:18 GMT

{
  "node": {
    "designation": "noeud9",
    "id": 9,
    "ip": "4.3.2.1",
    "port": "3244"
  }
}
```

Capture d'écran 4

Capture d'écran 5



Capture d'écran 6